Systèmes de fichiers

Cyril Rabat
cyril.rabat@univ-reims.fr

Licence 3 Informatique - Info0601 - Systèmes d'exploitation - concepts avancés

2021-2022

Cours n°2

Organisation des fichiers

Présentation de différents systèmes de fichiers

Version 30 novembre 2021

Table des matières

1. Fichiers et implantation
   - Implantation des systèmes de fichiers
   - Répertoires et liens
2. Exemples de systèmes de fichiers
   - ISO 9660
   - Systèmes de fichiers pour MS-DOS
   - Ext2
   - NTFS

Fichier logique et enregistrement

- **Fichier logique** : vision que l'utilisateur et les programmes ont des données
- **Fichier physique** : données stockées sur le support
- Chaque fichier est déterminé par un nom
- Ensemble d'opérations possibles :
  - Création/destruction : liaison fichier logique/fichier physique
  - Ouverture/fermeture : rupture de la liaison
- **Enregistrement** :
  - Unité logique de traitement
  - Structure de données compréhensible par le programme
  - Accessible via des fonctions d'accès : lecture et écriture

Les types de fichiers

- **Fichiers ordinaires** : contiennent des informations (données, programmes, bibliothèques)
- **Répertoires** : fichiers systèmes qui conservent la structure du système de fichiers
- **Fichiers spéciaux caractère** :
  - Liés aux entrées/sorties
  - Transfert de données via des périphériques
    $\hookrightarrow$ Terminaux, imprimantes et réseau
  - Pas de temporisation (envoi caractère par caractère)
- **Fichiers spéciaux bloc** :
  - Liés aux entrées/sorties
  - Transfert de données via des périphériques
    $\hookrightarrow$ Disques durs, clés, CD-ROM
  - Utilisation de tampons pour accélérer les transferts

Les fichiers ordinaires

- **Fichiers textes/ASCII** :
  - Contiennent des lignes de texte
  - Possibilité de les lire et de les imprimer directement
- **Fichiers binaires** :
  - Suite d'octets incompréhensibles (sauf pour les applications)
- Type du fichier :
  - Déterminé par une extension (exemple : `.exe` sous MSDOS)
  - Déterminé par un attribut (sous MAC OS)
  - Indéfini : spécifique à une application

Les attributs de fichier

- Informations complémentaires concernant un fichier
- Le nombre d'attributs varie en fonction du système
- Quelques exemples :
  - Indicateurs sur les possibilités de lecture, écriture et exécution
    ↪ `r`, `w` et `x` sous *Unix*
  - Créateur/propriétaire du fichier
  - Fichier ASCII ou binaire
  - L'heure et la date de création ...

Fichier logique : mode d'accès

- **Accès séquentiel** :
  - Traitement des enregistrements les uns après les autres
  - "Pointeur" sur l'enregistrement courant
  - Lecture : donne l'enregistrement courant et se déplace au suivant
  - Écriture : à la fin du fichier
  - Accès soit en lecture seule, soit en écriture seule
- **Accès indexé/aléatoire** :
  - Accès direct à un enregistrement
  - Utilisation d'une clé (champ commun à tous les enregistrements)
  - Index ou structure d'accès maintenu
  - Accès en lecture seule, en écriture seule ou en lecture/écriture
- **Accès direct/relatif** :
  - Réalisé en spécifiant la position relative de l'enregistrement
  - Soit à partir du début, de la fin ou de la position courante

Fichier physique

- Entité allouée sur le support
- Contient les données physiques
- Problématiques :
  ↪ Comment placer un fichier sur le support (blocs ou non) ?
  ↪ Où placer le fichier ?
  ↪ Comment le retrouver ensuite ?

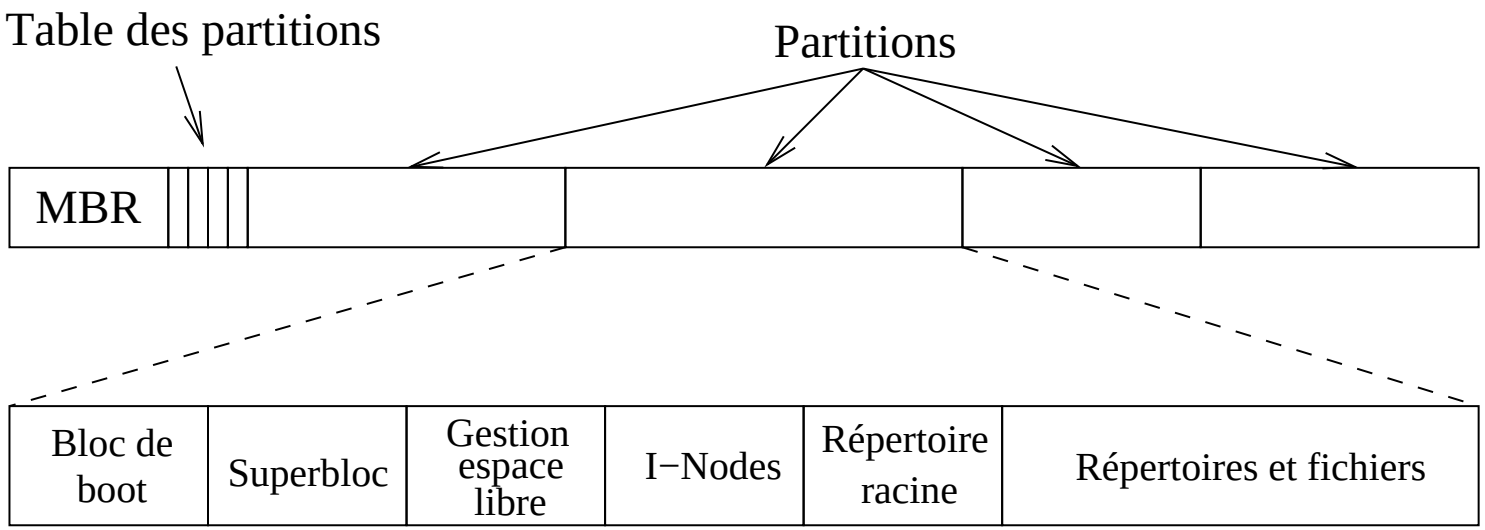
Les blocs

- Objectif : optimisation des opérations de lecture/écriture
- Unité d'échange entre le disque et la mémoire
- Correspond aux données qui peuvent être lues en une opération

Les partitions (1/2)

- Le secteur 0 du disque contient le MBR (*Master Boot Record*)
  ↪ Sert à "booter" la machine
- À la suite du MBR se trouve la table des partitions :
  ↪ Adresses de début et de fin de chaque partition
- Dans chaque partition :
  - Bloc de boot
  - Superbloc contenant les informations sur le système de fichiers

Les partitions (2/2)



- Au démarrage de la machine :
  - Interrogation du MBR pour obtenir la partition active
  - Lecture du premier bloc de cette partition : le *bloc de boot*
  - Le programme dans le bloc de boot charge le système

Implantation des fichiers

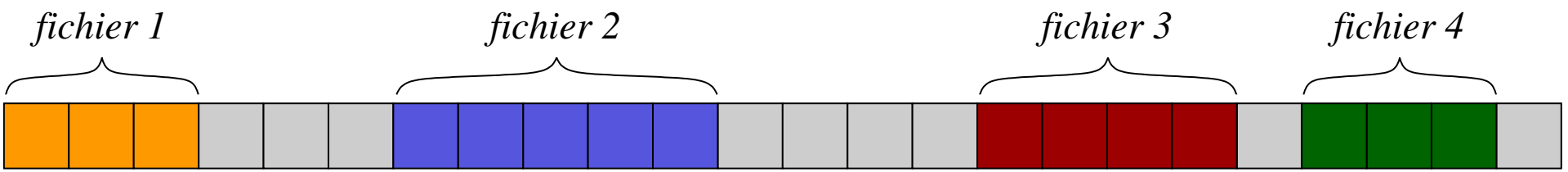
- Le disque est découpé en un ensemble de blocs
- Comment stocker les fichiers ?
- Plusieurs stratégies :
  - Contigüe
  - Par liste chaînée
  - Par liste chaînée et FAT
  - Indexée

Quelle que soit la stratégie,
un bloc est associé aux données d'un seul fichier

Implantation des fichiers : contigüe

- Les données d'un fichier sont stockées dans des blocs consécutifs
- Avantages :
  - Simple à mettre en œuvre
  - Excellentes performances car un seul déplacement au début du fichier.
- Inconvénient :
  - Fragmentation de l'espace libre
- Système utilisé avec les DVD et CD-Rom.

Illustration



Exemple d'implantation contigüe

- Problème : comment choisir la position d'un fichier ?
- Stratégies : *first-fit*, *next-fit*, *best-fit*, *worst-fit*

Stratégie *First-Fit*

- Algorithme de placement pour un fichier :
  1. Commencer la recherche depuis le début
  2. Avancer jusqu'au prochain segment libre
  3. Si la taille du segment n'est pas suffisante, avancer jusqu'au prochain segment libre
  4. Sinon, le segment courant est choisi
- Algorithme rapide car recherches limitées

Stratégie *Next-Fit*

- Idem que *First-Fit* mais la recherche débute à partir de la position trouvée précédemment
- Par simulation, on observe que cette stratégie est plus rapide que le *First-Fit*

Algorithme *Best-Fit*

- Fichier placé à l'endroit optimal :
  $\hookrightarrow$ le plus petit emplacement libre possible
- Stratégie plus lente que les deux précédentes
- Tendance à perdre de la place car les emplacements libres sont très petits

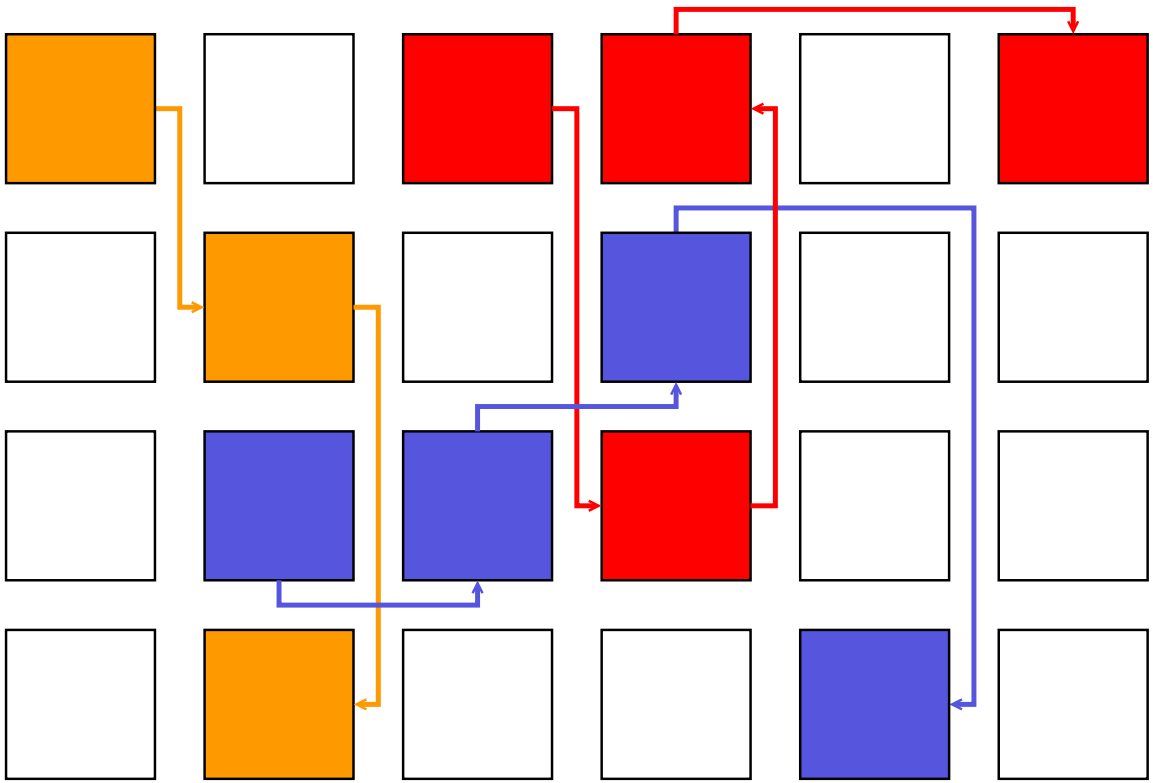
Stratégie *Worst-Fit*

- Idem que *Best-Fit* mais fichier placé dans l'emplacement le plus grand
- Par simulation, on observe que cet algorithme n'est pas performant

Implantation des fichiers : par liste chaînée

- Chaque fichier est considéré comme une liste chaînée de blocs
  $\hookrightarrow$ Les blocs des fichiers sont donc répartis sur le disque
- Avantage :
  - Pas de fragmentation de l'espace libre
- Inconvénients :
  - Lecture du fichier plus lente et plus complexe
  - Utilisation d'octets dans chaque bloc pour indiquer le prochain bloc

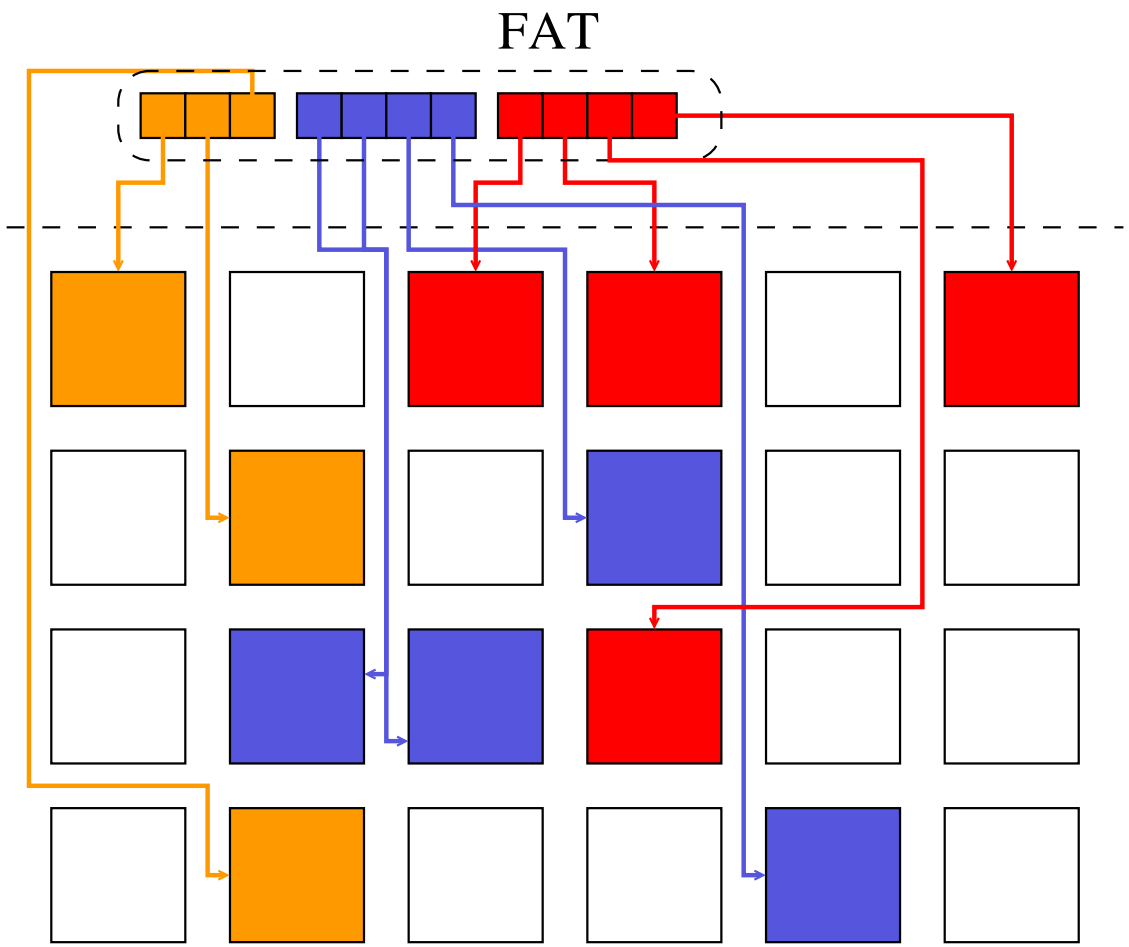
Exemple d'implantation par liste chaînée



Implantation des fichiers : par liste chaînée et FAT

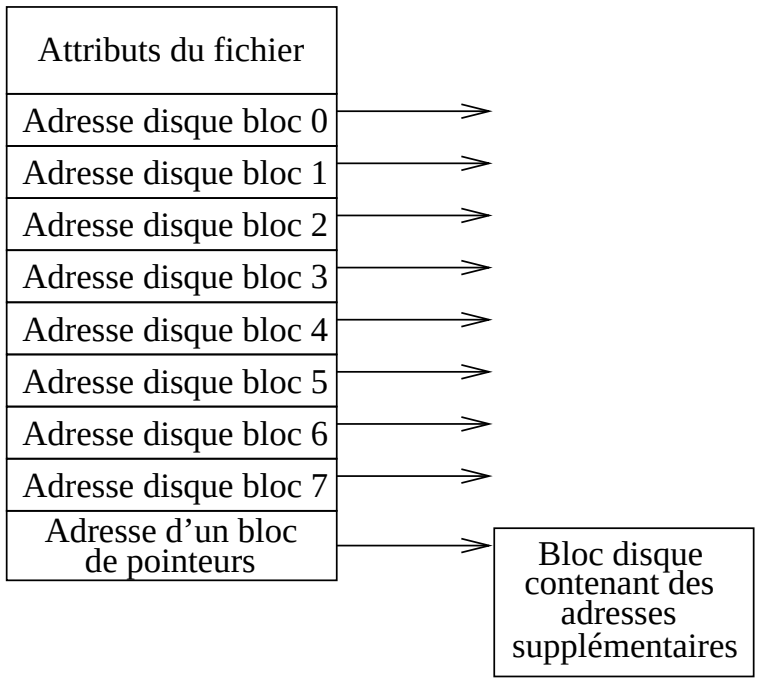
- Les pointeurs vers les blocs sont stockés dans une table :
  $\hookrightarrow$ La FAT (*File Allocation Table*)
- La table est complètement mise en mémoire
- Avantage :
  - Un bloc physique est intégralement disponible pour les données
- Inconvénient :
  - La place occupée par la table peut être considérable

Exemple d'implantation par FAT

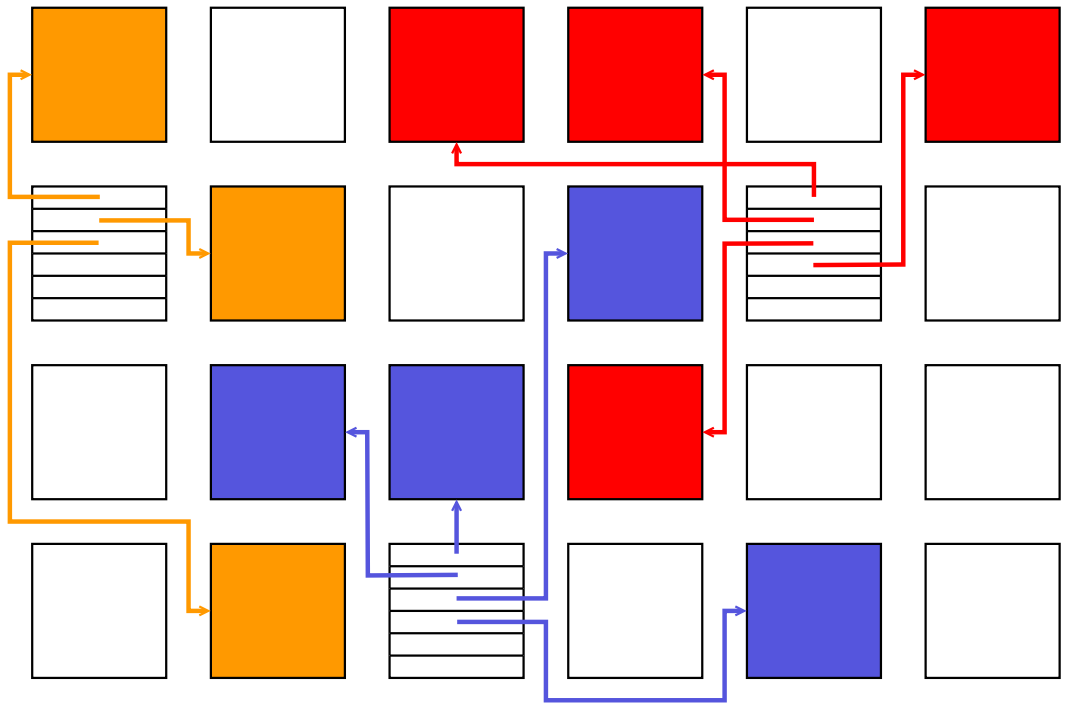


Implantation des fichiers : indexée

- À chaque fichier est associée une structure de données appelée *i-node* contenant :
  - Les attributs
  - Les adresses du disque des blocs du fichier
- Avantage :
  ↪ Seuls les *i-nodes* des fichiers ouverts sont chargés en mémoire



Exemple d'implantation indexée



Gestion de l'espace libre : par tableaux/chaine de bits

- À chaque bloc correspond un bit dans le tableau / la chaîne :
  ↪ 0 si le bloc est libre, 1 sinon
- Problème : taille du tableau / de la chaîne

Exemple

Tableau de bits correspondant :

1 1 1 0 0 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 0 0 0 0

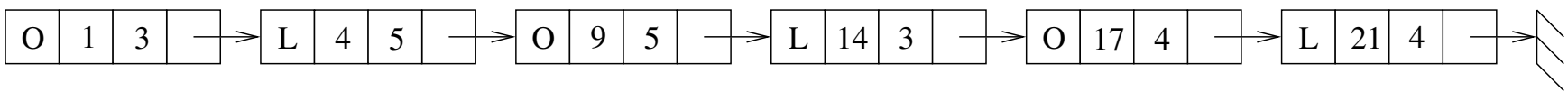
Gestion de l'espace libre : par listes chaînées

- Le tableau est remplacé par une liste chaînée
- Chaque entrée de la liste indique :
  - L'état du segment (L pour libre ou O pour occupé)
  - L'adresse à laquelle le segment débute
  - La taille du segment

Exemple

- Mémoire :

- Liste chaînée :



Les répertoires (1/2)

- Correspondance fichiers logiques / fichiers physiques :
  $\hookrightarrow$ Utilisation d'une table appelée **répertoire**
- Constitué d'un ensemble d'entrées, chacune correspondant à un fichier
- Contenu d'une entrée :
  - Le nom du fichier
  - Le type du fichier
  - La localisation physique
  - La taille
  - Le propriétaire
  - Les protections

Les répertoires (2/2)

- Différents niveaux de répertoires :
  - Systèmes à un seul niveau de répertoire :
    ↪ Un seul répertoire, l'ensemble des fichiers sont dans le répertoire
  - Systèmes à deux niveaux de répertoires :
    ↪ Un répertoire racine contenant un répertoire par utilisateur
    ↪ Chacun contient les fichiers de l'utilisateur
  - Systèmes à répertoires hiérarchiques :
    ↪ Arbre de répertoires (arborescence)

Remarque

- La majorité des systèmes d'exploitation actuels utilisent les répertoires hiérarchiques
- Attention cependant la taille maximum du chemin absolu !

"Lecture" d'un répertoire

- Possible d'ouvrir un répertoire en lecture :
  ↪ En C, utilisation de DIR (au lieu de FILE pour un fichier)
- Permet de parcourir les entrées du répertoire :
  ↪ Possible de faire un parcours récursif

Les liens

- Utilisés pour partager des fichiers entre plusieurs utilisateurs
- Lien **physique** ou **matériel** :
  - Ajout d'un nouveau nom dans le système de fichier qui pointe vers le même *i-node* que le nom original
  - Permet d'avoir plusieurs noms pour un même fichier
  - L'*i-node* contient un compteur de liens (visible grâce à `ls -l`)
- Lien **symbolique** :
  - Fichier texte qui contient le chemin d'accès et le nom du fichier vers lequel il pointe
  - Lien marqué par un type spécial
  - Associé à un raccourci

La norme ISO 9660

- ISO 9660 est un système de fichiers pour CD-ROM (1988)
- C'est un standard :
  ↪ Tous les lecteurs actuels sont compatibles ISO 9660
- Sur un CD, les données sont organisées en une spirale continue
- La spirale est divisée en blocs de 2352 octets
- La partie utile par bloc est de 2048 octets :
  ↪ Les autres octets sont utilisés pour les préambules, à la correction d'erreurs et à la gestion en général

Organisation sur le CD-ROM

- Les 16 premiers blocs n'ont pas d'utilisation définie
  $\hookrightarrow$ Destinés aux fabricants
- Le bloc descripteur primaire de volume composé des identificateurs :
  - Du système (32 octets)
  - Du volume (32 octets)
  - De l'éditeur (128 octets)
  - Du préparateur de données (128 octets)
  - Le nom de 3 fichiers qui contiennent le résumé, la note sur les droits réservés et des informations bibliographiques
  - Le nombre de blocs du CD, la date de création, la position du répertoire racine

Les répertoires

- Les répertoires sont composés d'un nombre variable d'entrées dont la dernière est marquée par un bit spécial
- Chaque entrée est de taille variable :
  ↪ Constituée de 10 à 12 champs
  ↪ Champs codés en ASCII ou en binaire
- Une entrée contient entre autres :
  - La longueur de l'entrée du répertoire (1 octet)
  - La localisation du premier bloc du fichier (8 octets) :
    ↪ Les fichiers sont stockés de manière contigüe
    ↪ La position du premier bloc détermine la position des suivants
  - Taille du fichier, date et heure
  - La taille du nom du fichier, le nom du fichier ...

Les niveaux

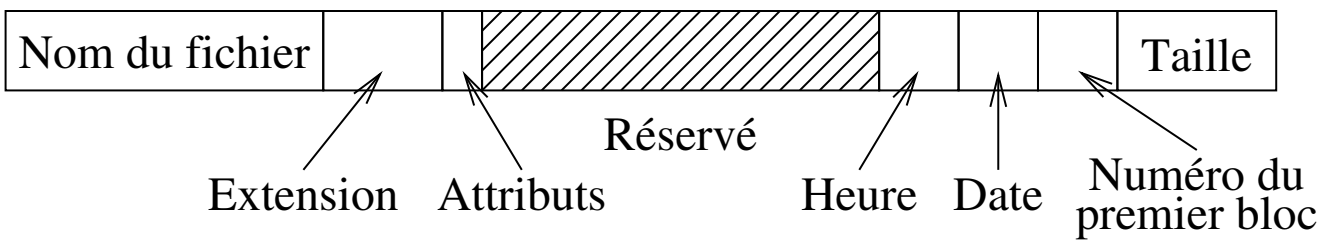
- Il existe 3 niveaux dans cette norme :
- Niveau 1 :
  - Nom de fichier de 8 + 3 caractères
  - Tous les fichiers sont contigus
  - Nom de répertoire de 8 caractères sans extension
- Niveau 2 : noms des fichiers et répertoires allant jusqu'à 31 caractères
- Niveau 3 :
  - Les fichiers ne sont plus contigus
  - Ils peuvent partager des blocs s'ils sont identiques :
    $\hookrightarrow$ Optimisation de l'espace

Les extensions de la norme

- *Rock Bridge* :
  - But : reproduire le système *Unix*
  - Attributs "`rwxrwxrwx`"
  - Liens symboliques
  - Relocalisation des répertoires pour augmenter la profondeur ...
- *Joliet* :
  - But : reproduire le système *Windows*
  - Noms longs jusqu'à 64 caractères
  - Caractères *Unicode* sur 2 octets
  - Profondeur de répertoire supérieure à 8
  - Nom des répertoires avec extension

Systèmes de fichiers pour MS-DOS

- Format d'une entrée d'un répertoire :
  - Nom du fichier (8 octets) + extension (3 octets)
  - Attributs : (1 octet)
    ↪ Lecture seule, archive, caché, système
  - 10 octets inutilisés
  - Heure (2 octets) et date à partir de 1980 (2 octets)
  - Indice du premier bloc du fichier (2 octets)
  - Taille du fichier (4 octets)



FAT pour MS-DOS : FAT 12

- Première version fonctionne avec des blocs de 512 octets
- 12 bits pour une adresse disque : $2^{12} = 4096$ blocs indexés
  $\hookrightarrow$ En fait seulement 4086 car 10 adresses sont réservées
- Taille max. par partition : 4086×512 octets
  $\hookrightarrow$ Partition maximum de 2Mo
- La FAT en mémoire est de 4096 entrées de 2 octets
- Pour accroitre la taille max., augmentation de la taille des blocs de 1Ko, 2Ko et 4Ko :
  $\hookrightarrow$ Partition maximum de 16Mo
- MS-DOS supportait 4 partitions par disque :
  $\hookrightarrow$ Gestion de disque jusqu'à 64Mo

FAT pour MS-DOS : FAT 16

- 16 bits pour une adresse disque
- Des blocs de 8Ko, 16Ko et 32Ko sont autorisés
- La FAT-16 occupe en permanence 128Ko de mémoire
- Taille maximale par partition : 2Go (64Ko d'entrées de 32Ko chacune)
- Taille maximale d'un disque de 8Go (4×2Go)

FAT pour MS-DOS : FAT 32

- 28 bits pour une adresse disque (et non 32 bits !)
- Théoriquement, taille des partitions de $2^{28} \times 2^{15} = 8$To ...
- ... mais bridée à 2To
- Avantage par rapport à la FAT-16 : un disque de 8Go en une seule partition
- Taille des blocs de 4ko à 64ko
- Recherche de blocs libres réalisée en analysant la FAT :
  $\hookrightarrow$ Les blocs libres ont un code particulier
- Taille des fichiers limitée à 4Go :
  $\hookrightarrow$ Limitée par l'entrée "taille de fichier" d'un répertoire
- Sous *Windows* : limitation du FAT32 à des partitions inférieures à 32Go

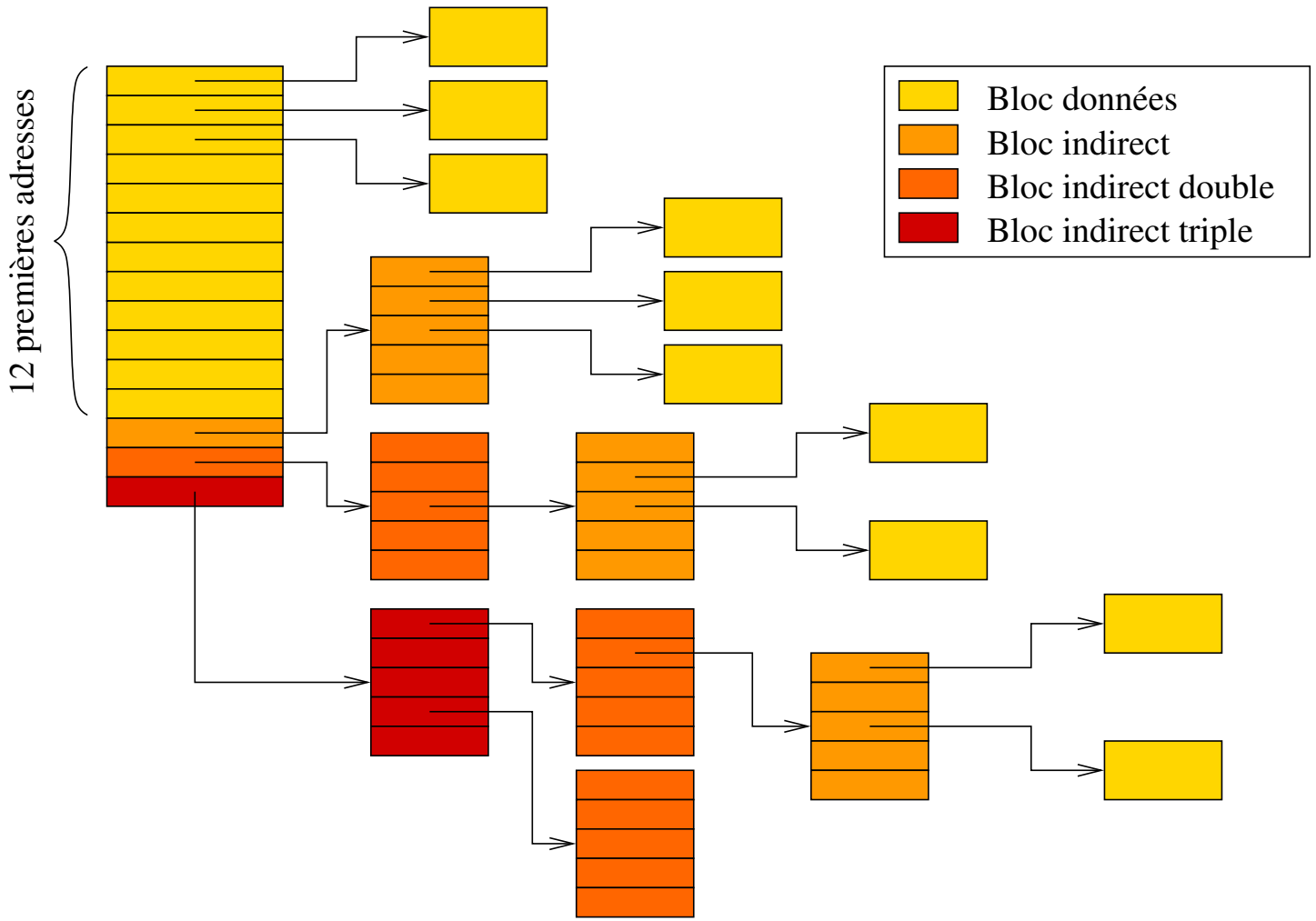
Le système de fichiers Ext2

- Ext2 pour *EXtended File System version 2*
- Premier bloc : le *boot*
- Ensuite, groupes de blocs contenant chacun :
  - Super bloc : nombre de blocs et *i-nodes*, taille des blocs, ...
  - Descripteur de groupe
  - Bitmap des blocs libres puis des *i-nodes* libres
  - Table des *i-nodes*
  - Blocs de données
- Blocs de 512o à 4096o

Les *i-nodes* (1/2)

- Chaque *i-nodes* fait 128 octets
- Contient :
  - Nom du fichier, type et droits
  - Heures (dernier accès, dernière modification, etc.)
  - Nombre de liens vers d'autres fichiers
  - Taille du fichier et nombre de blocs alloués
  - Liste de contrôle d'accès (une pour le fichier, une pour le répertoire)
  - Table d'adresses des blocs de données
- Table d'adresses :
  - Adresses sur 4 octets
  - 15 entrées
  - 12 premières : blocs logiques contenant des données
  - 3 dernières : différents niveaux d'indirection

Les *i-nodes* (2/2) : table d'adresses d'un fichier



La journalisation

- Lors d'un arrêt brutal de la machine, il est possible que la structure de données soit dans un état instable
- Il est nécessaire d'avoir recours à des outils :
  ↪ Exécutés automatiquement au démarrage
- Les outils vérifient alors toute la structure :
  ↪ Coût en temps important
- Pour éviter ça, une nouvelle structure de données est ajoutée :
  ↪ Structure appelée *journal*
- Toutes les actions que le système de fichiers s'apprête à faire y sont stockées
- Lorsqu'un problème survient, il suffit de reprendre le journal :
  ↪ Gain de temps important et intégrité des données garantie

Les évolutions de Ext2

- Ext3 :
  - Proposé pour apporter le principe de la journalisation au système de fichiers Ext2
  - Compatibilité entre les 2 systèmes
  - Suffisant pour un poste utilisateur, mais pas pour l'utilisation sur des serveurs
- Ext4 :
  - Gestion de disques jusqu'à 1024×2^{50} octets
  - Allocation contigüe des fichiers pour minimiser la fragmentation
  - Compatibilité limitée entre Ext3 et Ext4

NT *File System* : NTFS

- Mis au point pour *Windows* NT :
  ↪ Autre traduction : *New Technology File System*
- Adresses disques sur 64 bits
- Noms de fichiers limités à 255 caractères et chemin complet limité à 32767 caractères
- Utilisation possible de caractères *Unicode* avec le respect de la casse (mais pas par l'API *Win32*)
- Blocs de taille comprise entre 512Ko et 64Ko (généralement 4Ko)
- Les fichiers/répertoires correspondent à des enregistrements dans une table :
  ↪ La MFT

La MFT

- MFT pour *Master File Table*
- Chaque fichier et répertoire possède une entrée de 1Ko dans la MFT :
  ↪ Vu comme une suite d'attributs
- Si l'attribut peut être stocké dans la MFT, il est dit résident :
  ↪ Date de création, nom du fichier...
- Les attributs non-résidents stockés ailleurs sur le disque
- Si le fichier est petit, il est intégralement stocké dans la MFT
- La MFT est un fichier qui peut croitre jusqu'à 2^{48} enregistrements :
  ↪ Elle peut être placée n'importe où sur le disque

Fonctionnalités avancées de NTFS

- Support de la compression de fichiers, de répertoires et de volume :
  ↪ Lecture/écriture par n'importe quelle application sans décompression par un autre programme
  ↪ Algorithme de compression supportant des blocs de 4Ko maximum
- EFS (Encrypting File System) : NTFS5
  - Permet de protéger l'accès aux fichiers
  - Utilisation de clé de cryptage symétrique + clés publiques

Pour plus d'informations

Le site officiel : `http://ntfs.com/`