



**POLYTECHNIQUE  
MONTRÉAL**

UNIVERSITÉ  
D'INGÉNIERIE

**INF2610**

# Noyau d'un système d'exploitation

---

Chapitre 10 - Fichiers



# Système de fichiers

Partie du système d'exploitation qui se charge de:

- Manipuler les fichiers
- Gérer une unité de stockage (un disque, partition, etc.)

Afin que l'unité de stockage puisse être en mesure d'être manipulée par le système de fichiers, celle-ci doit être **formatée** selon un format **reconnu par le système d'exploitation**.

# Formatage

- Le formatage efface les données sur le périphérique et crée le répertoire racine du système de fichiers.
- Dans UNIX/LINUX, il y a également création d'un superbloc permettant d'assurer l'intégrité du système de fichiers.



**POLYTECHNIQUE  
MONTRÉAL**

UNIVERSITÉ  
D'INGÉNIERIE



# Fichier

- Un fichier permet de manipuler un **ensemble de données**
- Conteneur de données qu'on peut manipuler avec des opérations comme read, write, open, rename, close, etc.

Deux catégories d'attributs:

1. Contrôle d'accès (bits de protection)
2. État du fichier (taille, dates, indicateur de type de fichier)



# Stockage des fichiers (1)

Plusieurs politiques possibles (Clef USB, Disque pour macOS, etc.)

- On stocke les fichiers en une série de blocs de taille fixe (512, 1024, 2048 octets)
- L'unité de stockage d'un fichier est un bloc.
- On a donc une fragmentation interne (comme la mémoire au niveau des segments, mais pour les *blocs*)

Rappel: On peut utiliser un *superbloc* pour stocker des informations du volume, comme le nombre de blocs, leur taille, etc.



## Stockage des fichiers (2)

Lorsqu'on lit ou qu'on écrit dans un fichier

- On doit déplacer en mémoire tous les blocs du fichier touchant à ce qui est lu/écrit
- Rappelons que le bloc est l'unité d'allocation donc l'unité se **stockage et de manipulation** des données dans un fichier



**POLYTECHNIQUE  
MONTRÉAL**

UNIVERSITÉ  
D'INGÉNIERIE



# Comment savoir si un bloc est libre ou occupé?

1. Une liste chaînée des blocs libres
2. Une table de bits ayant autant d'entrée qu'il y a de blocs (1 pour occupé et 0 pour libre)

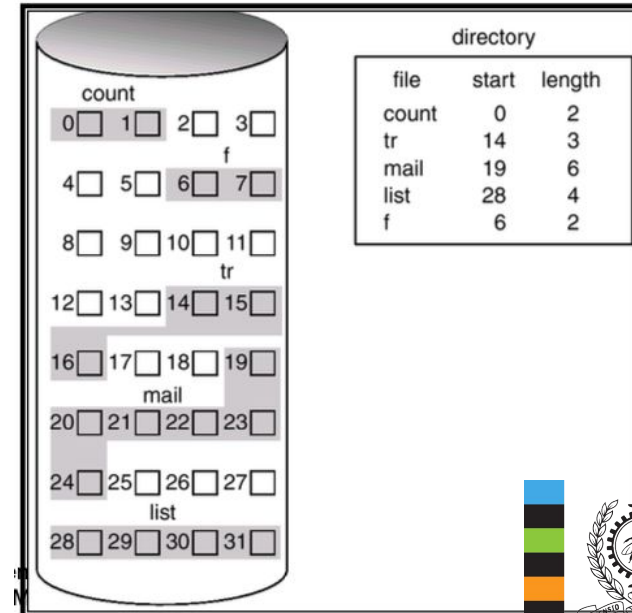
# Allocation : blocs contiguës

Démonstration:

- le fichier *count* commence au bloc 0 et a une longueur de 2 (il termine donc au bloc 1)
- Le fichier *tr* commence au bloc 14 et a une longueur de 3 (il termine donc au bloc 16)
- etc.

Conclusion

- Facile à implémenter
- Fragmentation interne
- Bonne performance de lecture



POLYTECHNIQUE  
MONTRÉAL

UNIVERSITÉ  
D'INGÉNÉRIE



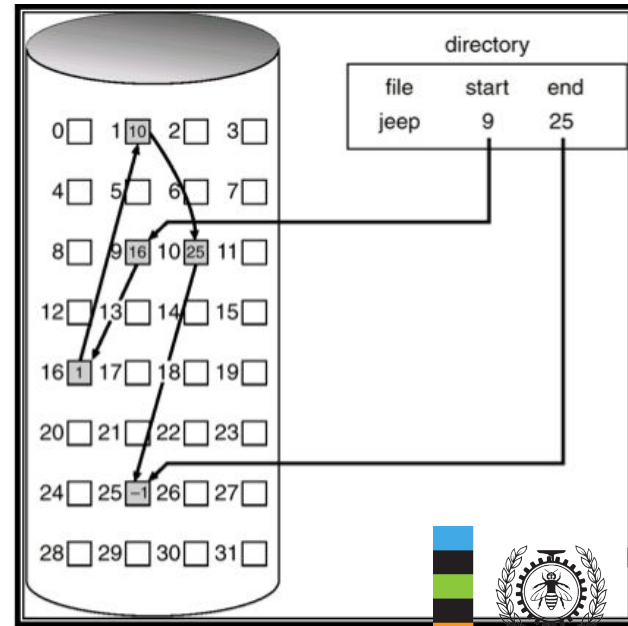
# Allocation : blocs non contiguës (1)

Démonstration:

- Le fichier jeep commence au bloc 9 et termine au bloc 25
- Il faut savoir comment se rendre au bloc suivant

Conclusion

- Fragmentation externe
- Le temps d'accès à un bloc est important



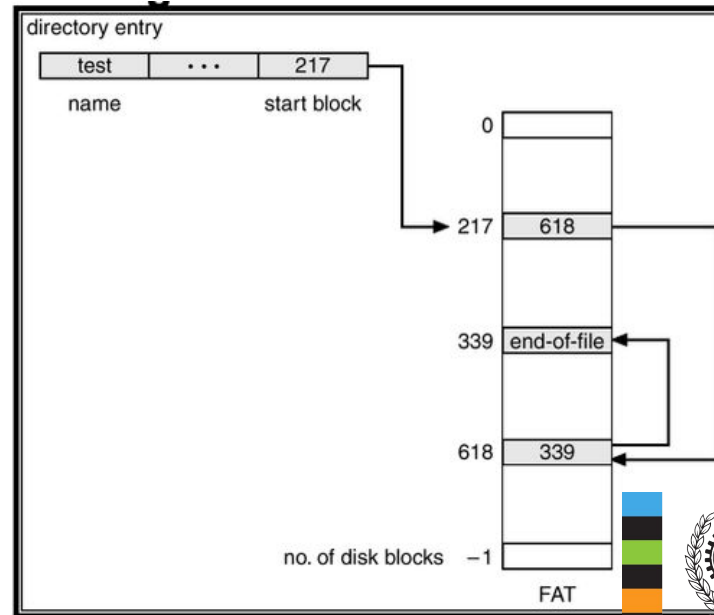
POLYTECHNIQUE  
MONTREAL

UNIVERSITÉ  
D'INGÉNÉRIE

## Allocation : blocs non contiguës (2)

Comment savoir quel est le bloc suivant?

**On utilise une table d'index**



**POLYTECHNIQUE  
MONTRÉAL**

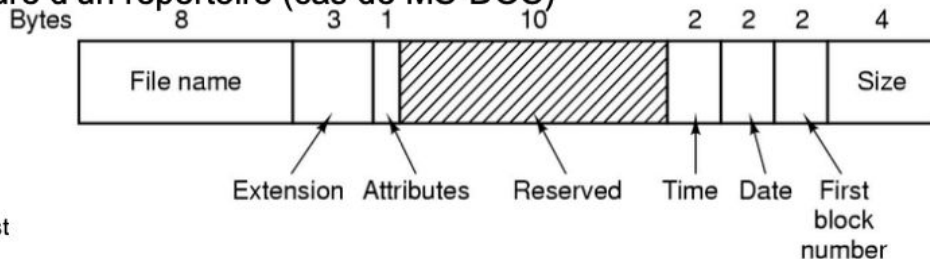
UNIVERSITÉ  
D'INGÉNÉRIE

Boot	Deux copies de la FAT	Répertoire Racine			Données et Répertoires	
------	--------------------------	----------------------	--	--	---------------------------	--

## FAT (File Allocation Table)

- Disquettes, MS-DOS, Clefs USB
- **Blocs non contiguës**
- La *File Allocation Table* est une table dont l'index est le numéro du bloc. L'entrée identifie:
  - le numéro du bloc suivant (valeur impossible ex -1 si EOF)
  - Un bit d'utilisation (1 si le bloc est utilisé)
- On utilise un **répertoire** pour savoir quel est le premier bloc d'un fichier (avec d'autres données sur le fichier)

Structure d'un répertoire (cas de MS-DOS)



au d'un syst

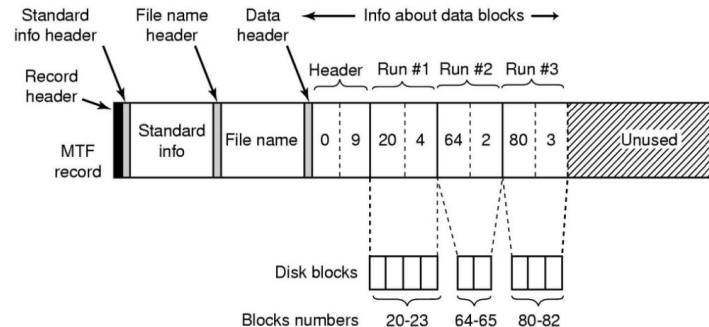


**POLYTECHNIQUE  
MONTRÉAL**

UNIVERSITÉ  
D'INGÉNIERIE

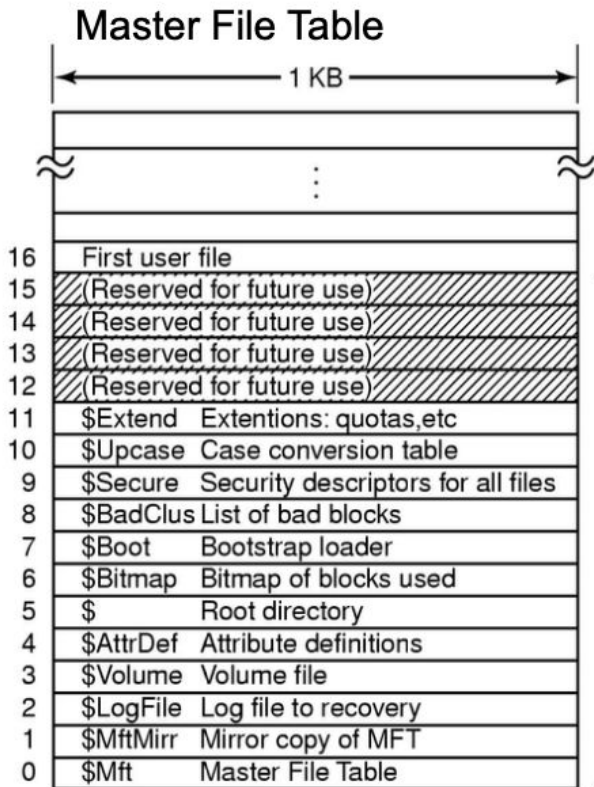
# NTFS (1)

- La partition NTFS contient des **fichiers, répertoires, bitmaps** et autres structures de données.
- La **Master File Table (MFT)** est la structure de données principale.
  - chaque entrée de la MFT contient l'information d'un **fichier ou répertoire**
  - Chaque entrée fait un KO



# NTFS (2)

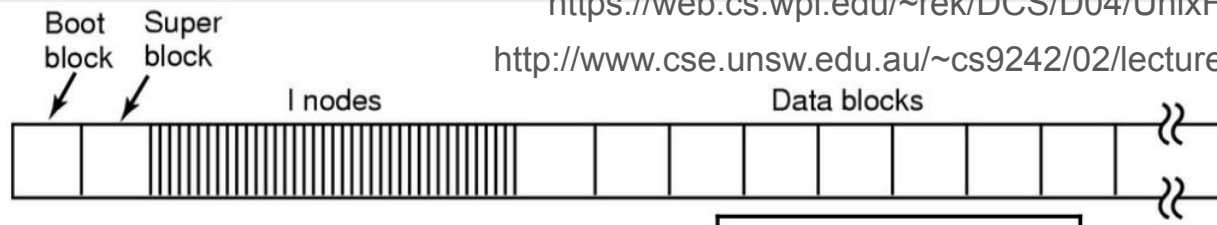
- Si un fichier ou un répertoire est trop gros, il sera associé à plusieurs entrées chaînées de la MFT.
- Une bitmap permet de récupérer les enregistrements libres ou occupés



Attribute	Description
Standard information	Flag bits, timestamps, etc.
File name	File name in Unicode; may be repeated for MS-DOS name
Security descriptor	Obsolete. Security information is now in \$Extend\$Secure
Attribute list	Location of additional MFT records, if needed
Object ID	64-bit file identifier unique to this volume
Reparse point	Used for mounting and symbolic links
Volume name	Name of this volume (used only in \$Volume)
Volume information	Volume version (used only in \$Volume)
Index root	Used for directories
Index allocation	Used for very large directories
Bitmap	Used for very large directories
Logged utility stream	Controls logging to \$LogFile
Data	Stream data; may be repeated

Metadata files





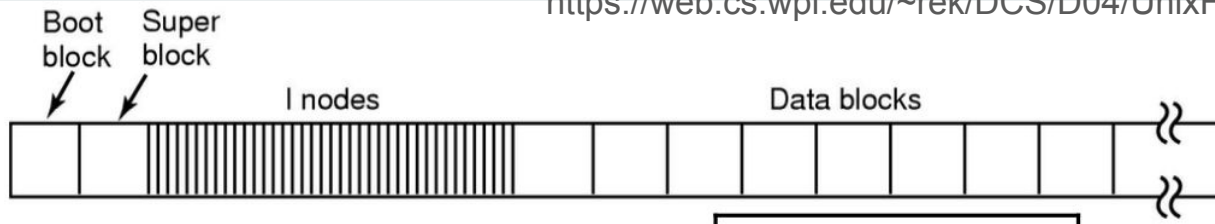
## FFS (Fast File System) / UFC (UNIX File System)

- Boot bloc: Contient le programme pour initialiser le SE
- Super bloc: Même utilité que le super bloc des autres FS
- I-nodes: Structure de données décrivant un fichier **et ses blocs**.
- Blocs de données: blocs des données des fichiers (comme dans les autres FS)



**POLYTECHNIQUE  
MONTRÉAL**

UNIVERSITÉ  
D'INGÉNIÉRIE



# FFS (Fast File System) / UFC (UNIX File System)

## i-node

- Contient l'information d'un fichier
- Les adresses des blocs de données
- La taille du fichier
- L'identifiant d'un fichier
- etc.

Mode
Link count
UID
GID
File size
Times
Address of the first block
Address of the second block
...
Address of the 10 block
Pointeur indirect simple
Pointeur indirect double
Pointeur indirect triple

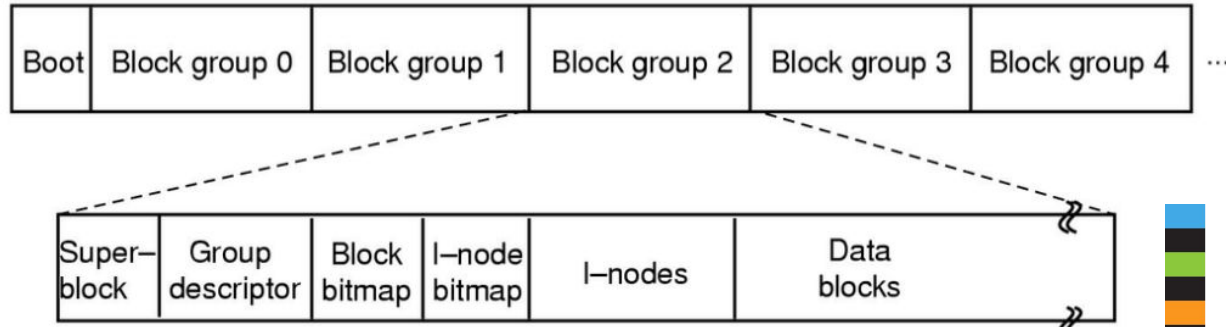
Table d'index



**Structure d'un i-noeud**

# Linux Extended-2 File System (Ext2FS)

- Chaque groupe de blocs a une collection d'i-noeuds
- Chaque groupe de blocs a un super bloc, descripteur de groupe, bloc de bitmap, blocs, etc.



**POLYTECHNIQUE  
MONTRÉAL**

UNIVERSITÉ  
D'INGÉNIERIE





# En résumé

Lors de l'élaboration d'un système de fichiers, il faut avoir en tête:

- Efficacité d'accès aux blocs (en temps)
- Bonne gestion de l'espace
- Intégrité des métadonnées
- Partage des blocs de données avec Copy-On-Write
- Chiffrement
- Clonage efficace des fichiers (uniquement les modifications)



**POLYTECHNIQUE  
MONTRÉAL**

UNIVERSITÉ  
D'INGÉNIERIE



# Liens symboliques

- Pointeur vers un fichier
- Sa destruction n'affecte aucunement le fichier pointé
- Permet de créer un lien vers un fichier qui n'est pas forcément dans le même système de fichiers
- La commande shell d'UNIX qui crée des liens symboliques est `ln` avec l'option `s` pour symbolique. L'appel système est `symlink`.



**POLYTECHNIQUE  
MONTRÉAL**

UNIVERSITÉ  
D'INGÉNIERIE

# Exercice

- On considère un système disposant d'un système de fichiers similaire à celui d'UNIX (FFS) avec une taille de blocs de données de 4KiB (4096 octets) et des pointeurs (numéros de blocs) définies sur 4 octets.
- On suppose que le i-noeud de chaque fichier compte 12 pointeurs directs, 1 pointeur indirect simple, 1 pointeur indirect double et 1 pointeur indirect triple.
- On désire créer un fichier contenant un total de 20.000.000 (vingt millions) de caractères (caractères de fin de ligne et de fin de fichier compris).
- Quelle est la fragmentation interne **totale** sur le disque résultant de la création de ce fichier ?
  - $20\,000\,000 \text{ octets} = 4882 * 4096 + 3328 \text{ octets}$
  - $4883 \text{ blocs} = 12 + 4096 + 775 \text{ blocs}$
  - $4096 - 3328 = 768 \text{ ....}$

