

# Systemes d'exploitation Unix ou Linux

## Chapitre 7-1

Introduction

Système de fichiers

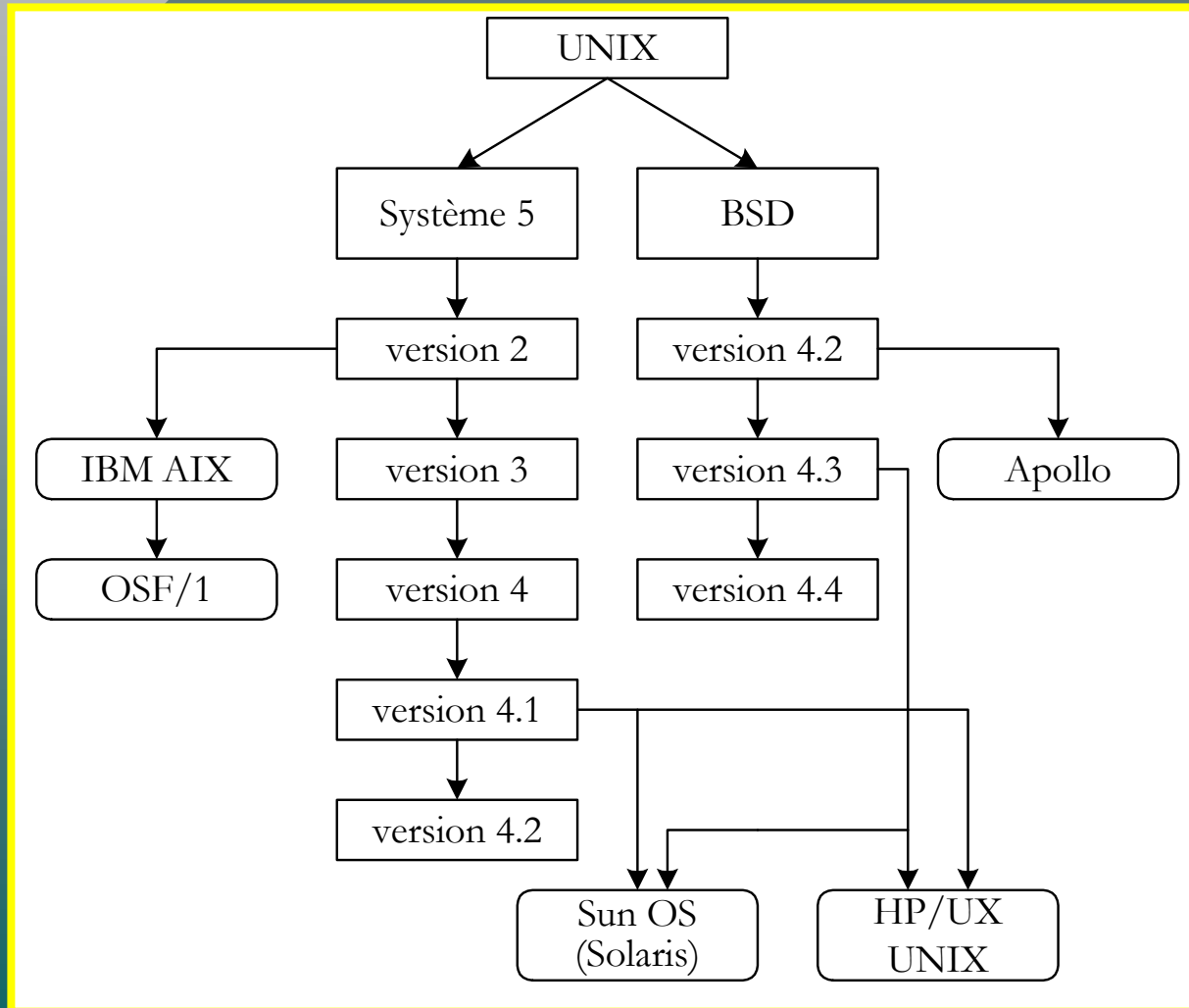
Gestion des processus

Gestion de la mémoire

# Linux ou Unix, Quelles Différences ?

- Qu'est-ce que Unix ?
    - Système créé en 1969 chez AT&T
    - Fin des 70 : réécriture à Berkeley : BSD 4.1
    - Souche AT&T évolue vers System V
    - Souche BSD reprise chez Sun, DEC et HP
    - Début des 90 : multiples combinaisons des 2 souches, arrivée AIX, SunOS (BSD) devient Solaris (System V) ...
- ⇒ La standardisation est indispensable !!

# Linux ou Unix, Quelles Différences ?



# Linux ou Unix, Quelles Différences ?

- Qu'est-ce que Linux ?
  - Noyau Unix-like
    - Mélange/adaptation de diverses technos (Minix, sockets BSD, IPC System V, VFS Sun, ...)
    - Ensemble de commandes GNU
    - G.N.U = Gnu's Not Unix !
  - Nombreuses « contrib » (utions)
    - Contribution = application « offerte » à la communauté

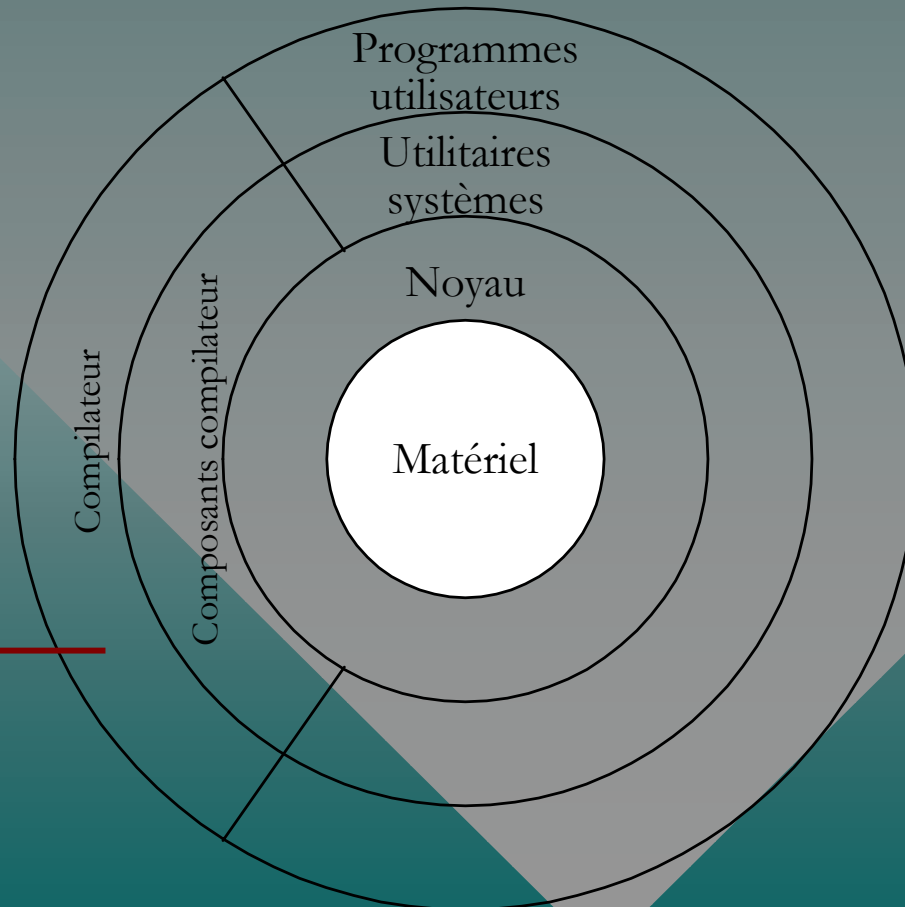
# Architecture interne d'UNIX

---

## ➤ Organisation traditionnelle:

Le compilateur C faisait partie des composants du S.E. parce qu'il fallait recompiler souvent le noyau.

---



# L'Organisation du disque

- Un disque dur est généralement découpé en « tranches », les **partitions** :
  - L'espace de stockage de chaque partition peut être **géré** séparément :
    - Zone réservée au fonctionnement du système
    - Zone réservée aux utilisateurs
    - Zone réservée à la mémoire virtuelle (swap)
  - Cohabitation de technologies différentes
    - Systèmes multi-boot (Linux, NT, Solaris, SCO, ...)

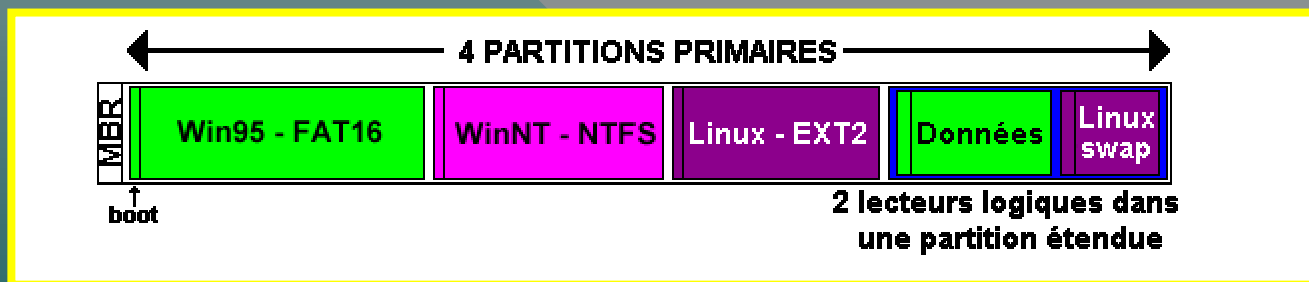
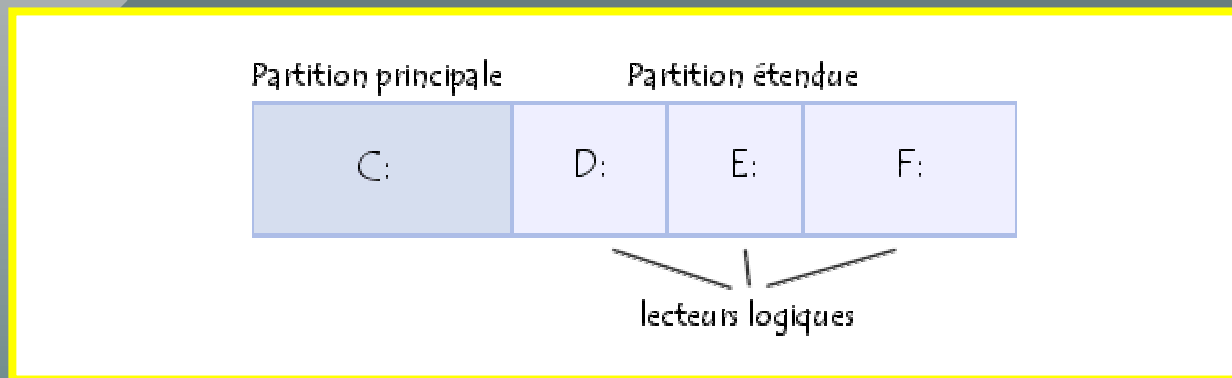
# L'Organisation du Disque

- Disques PC = 3 types de partitions
  - Partitions principale
    - Subdivision primaire du disque
    - Au maximum 4, numérotées de 1 à 4
  - Partition étendue (une seule)
  - Partitions logiques (lecteurs logiques)

Un disque peut contenir jusqu'à quatre partitions principales (dont une seule peut être active), ou trois partitions principales et une partition étendue. Dans la partition étendue l'utilisateur peut créer des lecteurs logiques (c'est-à-dire "simuler" plusieurs disques durs de taille moindre).

# L'Organisation du Disque

## Exemple de découpage en partitions





# Les Systèmes de Fichiers

- Rôle et Caractéristiques
  - Ranger, organiser et retrouver de l'information
    - Catégorisation (types de fichiers)
    - Organisation (hiérarchique)
  - Accéder à des volumes d'information «amovibles»
    - Montage

# Schéma Typique d'Organisation des Données d'une Partition UNIX

L'organisation exacte dépend du type de système de fichiers

L'espace disque est structuré en *blocs* de taille fixe (multiple de la taille d'un secteur)



*Optionnel*

*Information sur le système de fichiers lui-même (structure, dates de mise à jour...)*

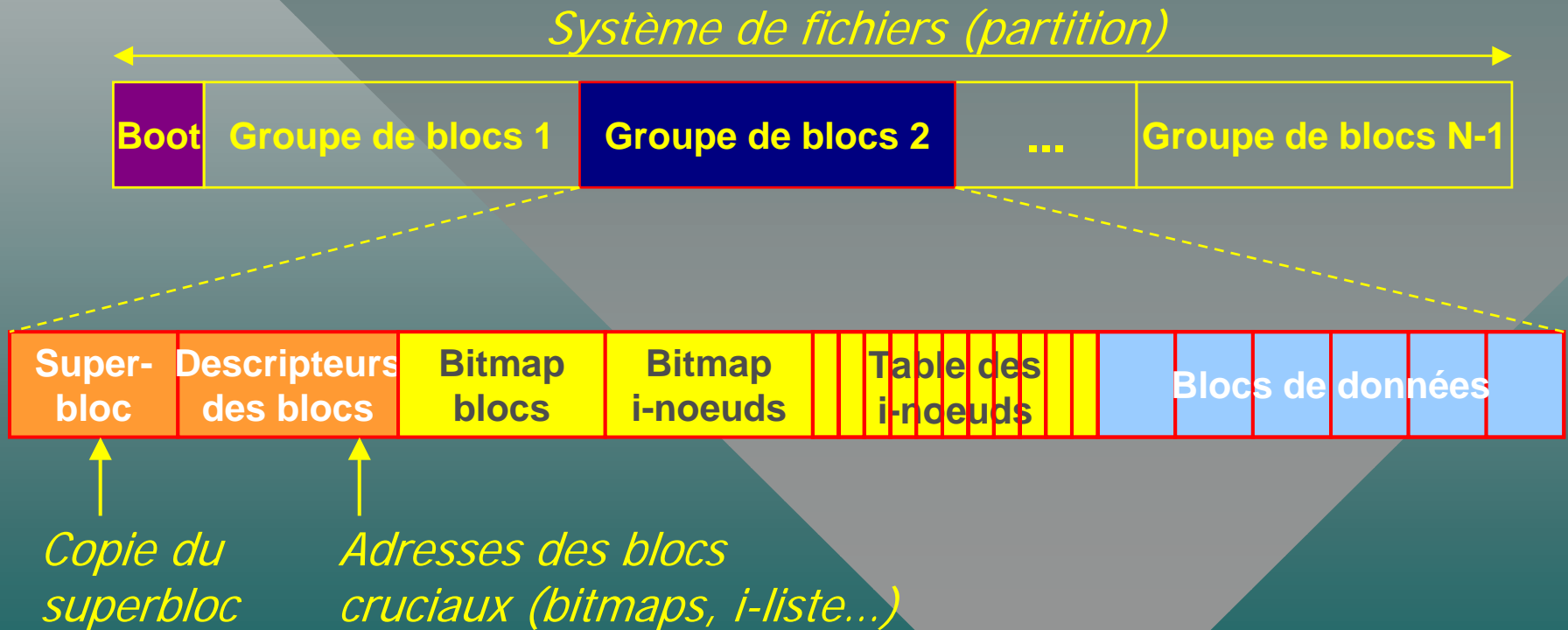
*Liste de **i-nœuds** : descripteurs de fichiers physiques*

*Maps d'allocation des **i-nœuds** et des blocs*

*Blocs contenant les données (répertoires, fichiers ordinaires...)*

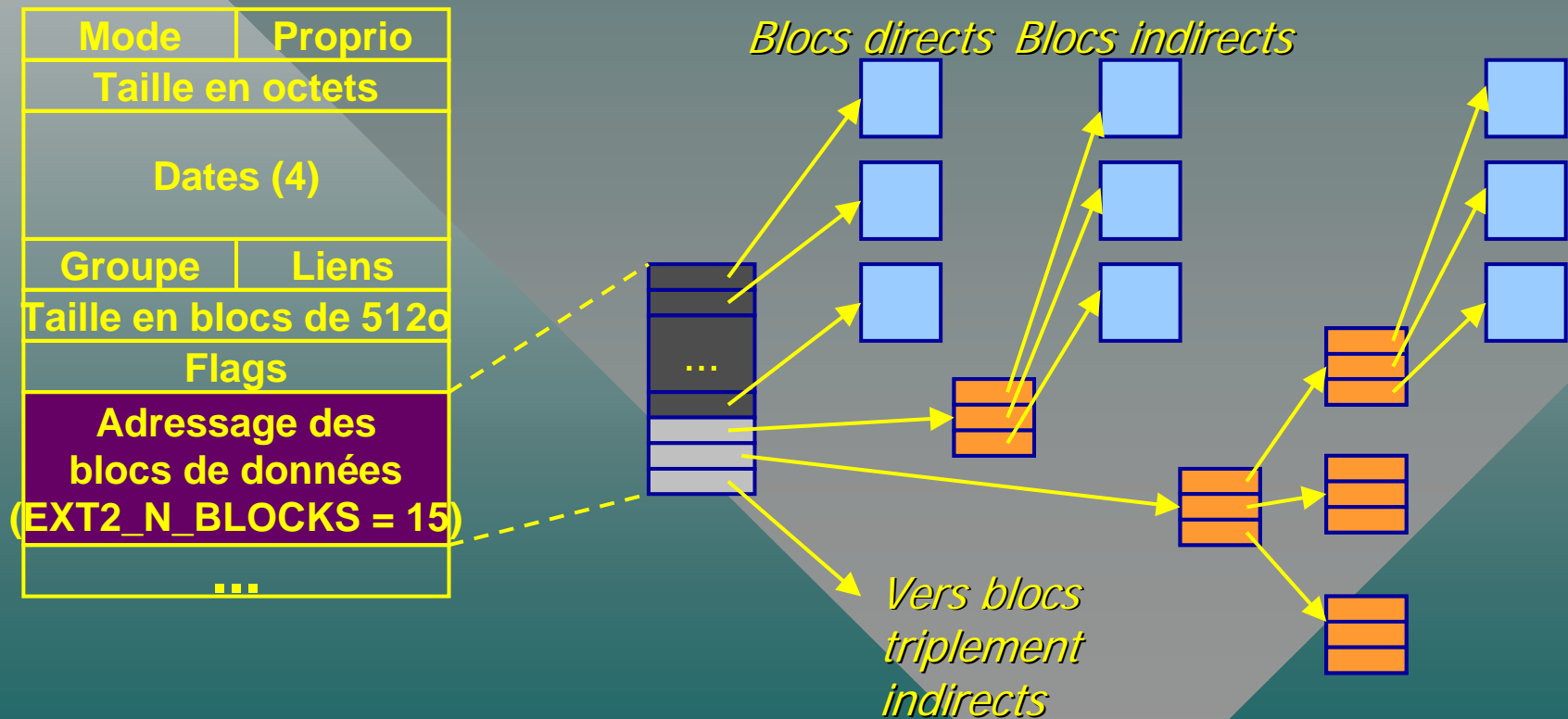
# Le Système de Fichiers Ext2fs

## Schéma d'organisation d'une partition ext2



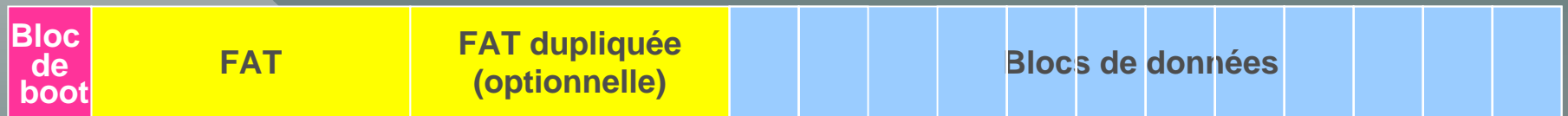
# Le Système de Fichiers Ext2fs

## Structure d'un i-nœud sur disque



# Le Système de Fichiers FAT (MSDOS/Windows)

## Structure d'une partition ms/dos



**File Allocation Table**  
(analogue à la table des i-nœuds)  
Une entrée dans la fat par bloc de  
donnée : taille de l'entrée 16 ou 32 bits

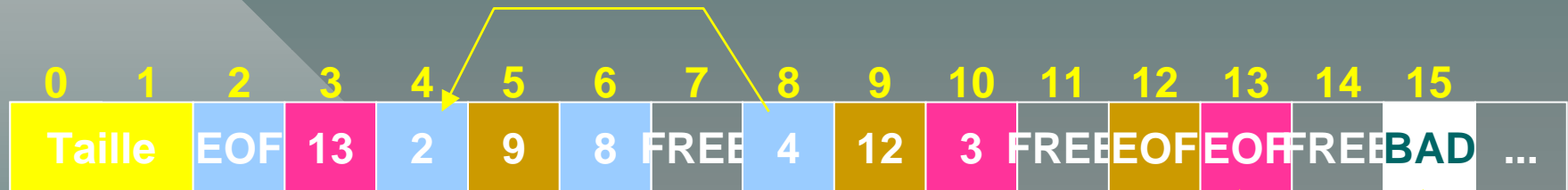
Bloc de taille variant de  
1 à 8 secteurs en fonction  
de la taille totale de la partition

Bloc de boot  
secondaire

# Le Système de Fichiers FAT

## (2)

### Structure de la FAT (principe)



Fichier A :	6	8	4	2
Fichier B :	5	9	12	
Fichier C :	10	3	13	

*Numéro du bloc  
suivant du fichier*

*Dernier bloc  
du fichier*

*Bloc libre*

*Mauvais bloc  
(inutilisable)*

# Méthodes d'allocation

- Il existe principalement 3 types de méthodes d'allocation de l'espace disque
  - L'allocation contiguë
  - L'allocation chaînée
  - L'allocation indexée

# FAT – une variation de l'allocation chaînée

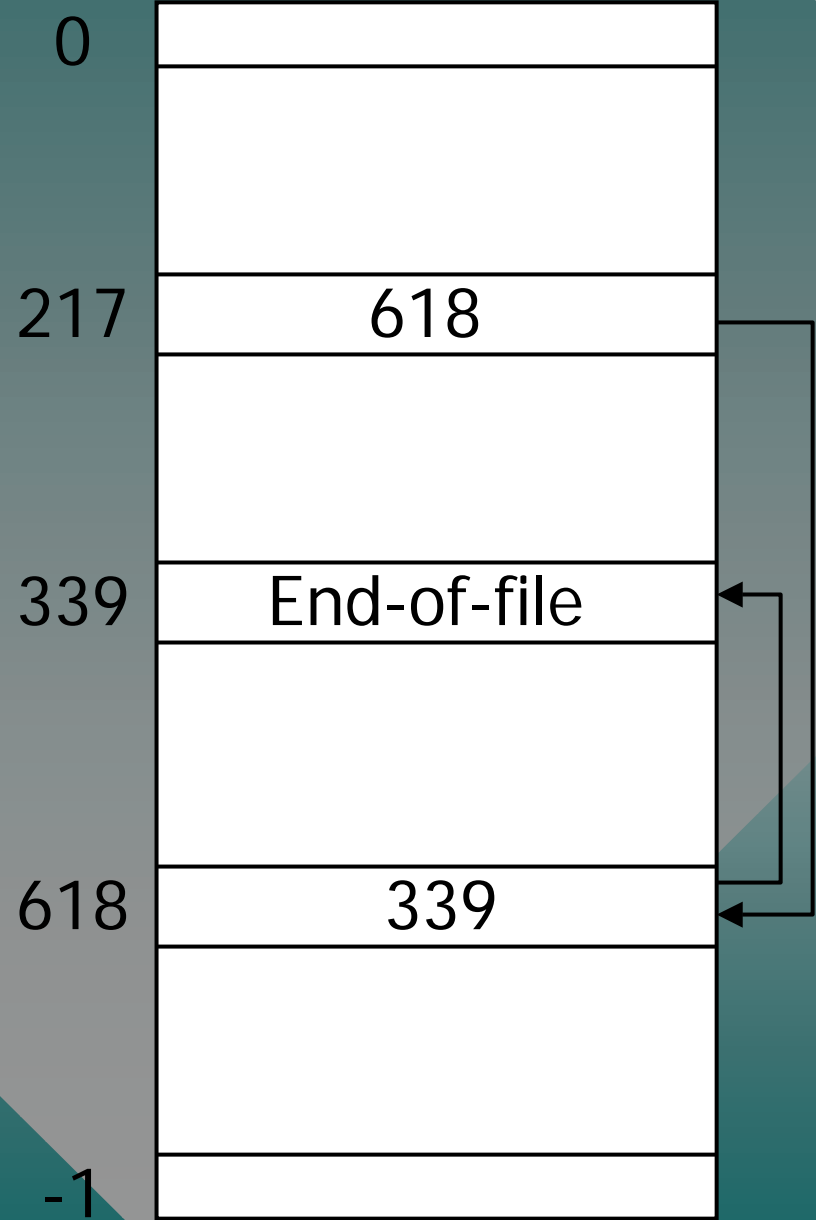
- Une section de disque au début de chaque partition contient une table d'allocation de fichiers (FAT)
- La table contient une entrée pour chacun des blocs disque
- Elle est indexée par le numéro du bloc et le contenu de la table est le numéro du bloc suivant



fichier	Bloc de début
Mail	217

Chacune des entrées fait référence à un bloc sur le disque

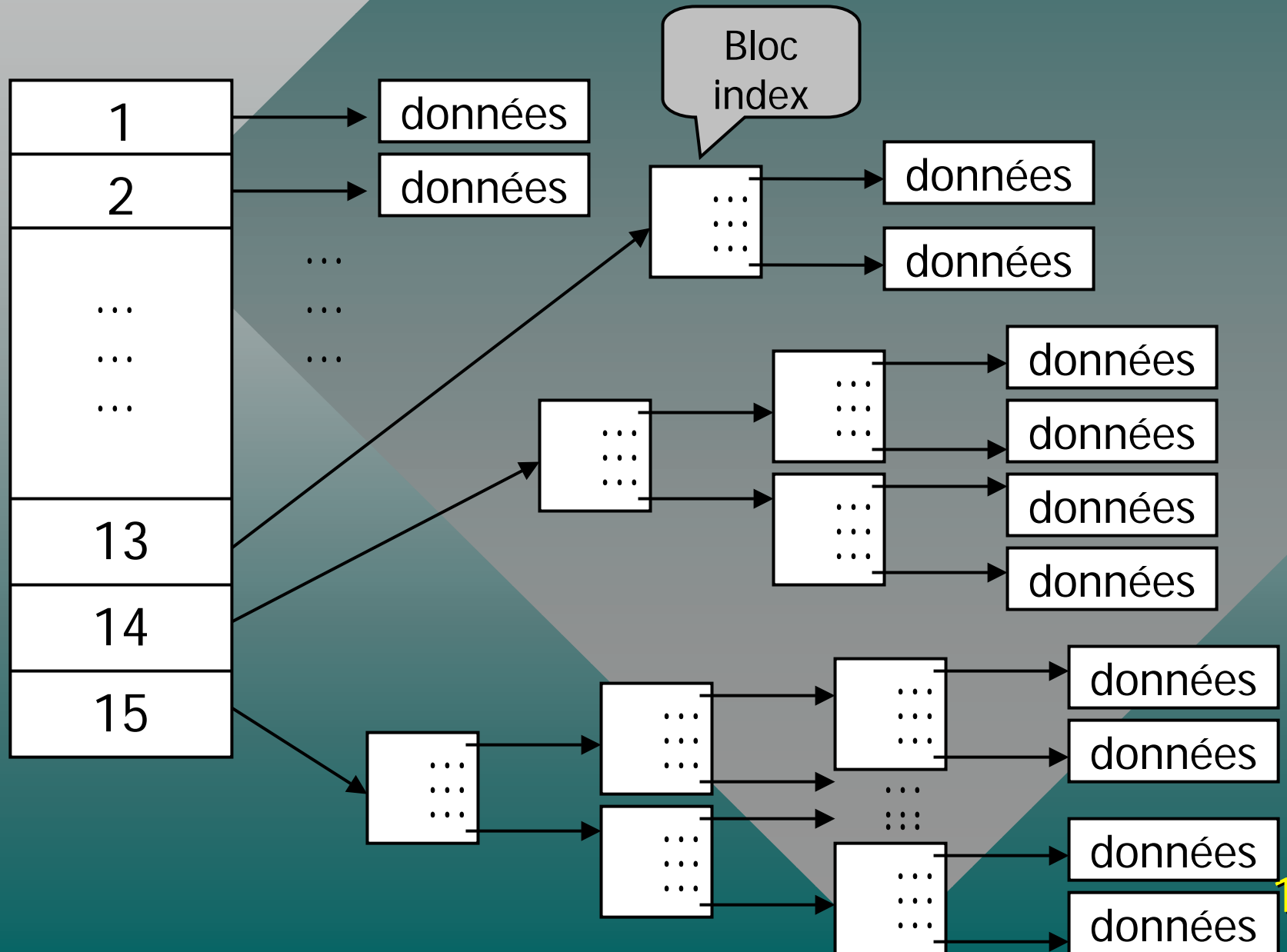
Le contenu du fichier Mail est donc écrit dans les blocs 217, 618 et 339



# Allocation indexée - INODE de UNIX (BSD)

- L'entrée dans le répertoire pointe sur un INODE
- INODE = bloc index composé de 15 pointeurs
- Les 12 premiers pointeurs pointent sur des blocs de données
- Le 13<sup>ième</sup> pointe sur un bloc index dont les pointeurs pointent sur des données (simple indirection)
- Le 14<sup>ième</sup> pointe sur un bloc index dont les pointeurs pointent sur d'autres blocs index dont les pointeurs pointent sur des données (double indirection)
- Le 15<sup>ième</sup> pointe sur un bloc index ... (triple indirection)

# INODE



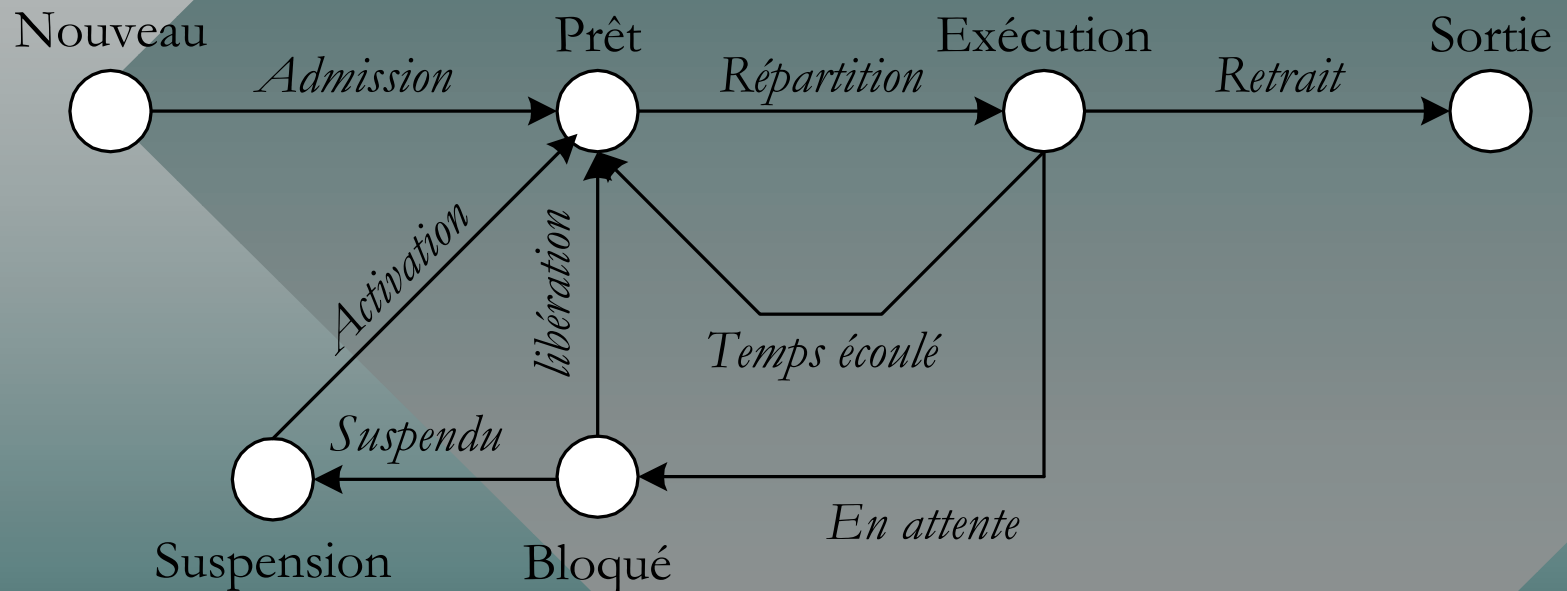
# Rôles fondamentaux

---

Tous les S.E. ont comme premier objectif:

- gestion efficace des processus.
- gestion de la mémoire.
- gestion des périphériques.

# Modèle pratique des processus



Déplacement d'un processus vers/de la mémoire secondaire (*swapping*).

Libère les ressources utilisées afin d'admettre de nouveaux processus dans le système.

# Modèle UNIX

---

## ➤ Zombie:

- processus sorti mais laisse dans le système des informations destinées au processus père qui n'existe plus.

## ➤ Prêt (en mémoire):

- processus prêt pour être exécuté et dont le code est en mémoire.

## ➤ Bloqué (en mémoire):

- processus bloqué en attendant un événement déclencheur (code en mémoire).

# Modèle UNIX

---

## ➤ Créé:

- nouveau processus qui n'est pas encore prêt pour l'exécution.

## ➤ Prêt (suspendu):

- processus prêt pour être exécuté et dont le code est en mémoire secondaire (disque).

## ➤ Bloqué (suspendu):

- processus bloqué en attendant un événement déclencheur (code en mémoire secondaire).

# Modèle UNIX

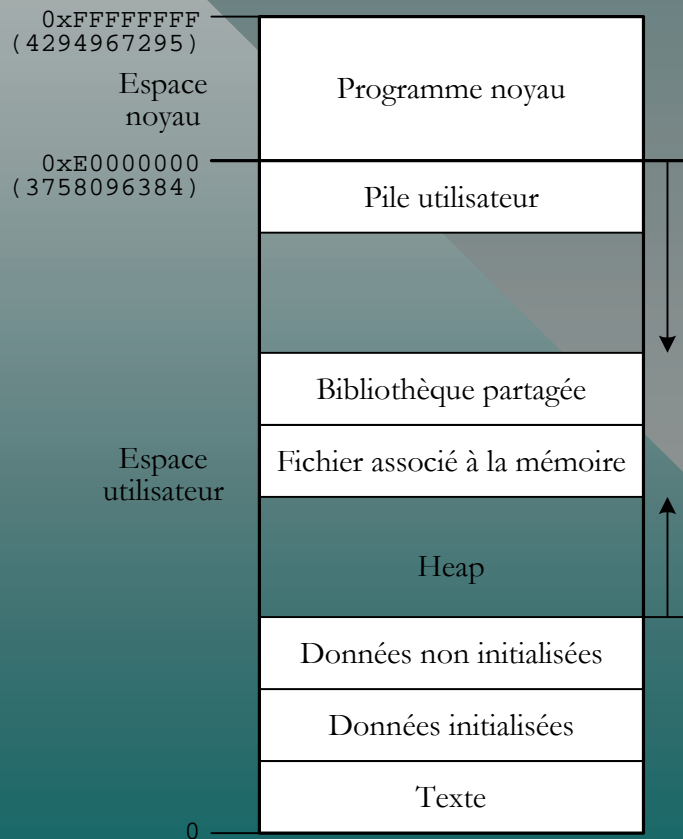
---

- Pour le SVR4, deux processus fondamentaux toujours en exécution:
  - processus qui réalise le déplacement des autres processus de/vers la mémoire centrale (*swapper*);
  - *swapper* → porte le numéro d'identification de processus PID = 0;
  - ce dernier est responsable de la création du processus « init » (PID = 1);
  - tous les autres processus sont descendants de ce processus init.



# Espace d'adresses des processus

## ➤ Dans le SVR4:



- La pile utilisateur « grossit » vers les adresses basses.

- Le heap « grossit » vers les adresses hautes.

# Espace d'adresses des processus

---

- Espace virtuel est divisé en segments.
- Segment « Texte »:
  - code exécutable du programme;
  - entreposé dans le fichier exécutable;
  - à lecture seulement;
  - ce segment est partageable.

Plusieurs processus utilisant le même code exécutable. Les processus partagent alors le même segment « Texte ».

# Espace d'adresses des processus

---

## ➤ Segment « Données »:

- entreposage des symboles constants ou globaux.
- deux types de données → données initialisées et données non initialisées.

Variables globales qui ont une valeur assignées dans le code source du programme. Elles ne sont pas partageables.

Même que les données initialisées mais leur contenu est toujours mise à zéro.

# Espace d'adresses des processus

---

## ➤ Segment « Heap »:

- zone de mémoire pour l'entreposage des données du processus;
- allocation dynamique de la mémoire;
- taille du segment varie en fonction des besoins du processus;
- segment grossit vers les adresses croissantes

# Espace d'adresses des processus

---

- Segment « Fichier associé à la mémoire » (*Mapped File*):
  - facilite l'accès des fichiers par le processus;
  - au lieu de faire appel aux services systèmes;
  - on associe un fichier à la mémoire;
  - accès du fichier est plus rapide;
  - (concept existe également sous Windows NT).

# Espace d'adresses des processus

---

## ➤ Segment « Bibliothèque partagée » (*shared Library*):

- Bibliothèque partagée → code objets utilisables par plus d'un processus à la fois;
- codes objets chargés dans l'espace virtuel du processus par un appel de système;
- espace virtuel réservé pour ces codes objets est le segment « Bibliothèque partagée »;
- par convention, les bibliothèques partagées portent l'extension .so

# Espace d'adresses des processus

---

- Segment « Pile utilisateur »:
  - entreposage des variables locales;
  - adresse de retour des routines;
  - paramètres d'entrée des fonctions;
  - contenu des registres du processeur;
  - valeur de retour des fonctions appels systèmes;
  - segment grossit vers les adresses décroissantes.

# Gestion de la mémoire

---

- Quantité de la mémoire physique est limitée:
  - mémoire virtuelle → illusion de disposer toute la mémoire de la machine pour chacun des processus;
  - mémoire virtuelle → illusion d'une carte de mémoire contiguë pour le programmeur;
  - gestionnaire de mémoire travaille d'une manière transparente au service des processus.



# Gestion de la mémoire

---

- Gestionnaire de la mémoire:
  - intimement lié à l'architecture de l'ordinateur;
- Organisation de la mémoire:
  - pages de taille fixe (8 Ko);
  - adresse physique → {page, décalage};
  - MMU (*Memory Management Unit*) sert à convertir les adresses physiques en adresses virtuelles et vice versa.

# Gestion de la mémoire

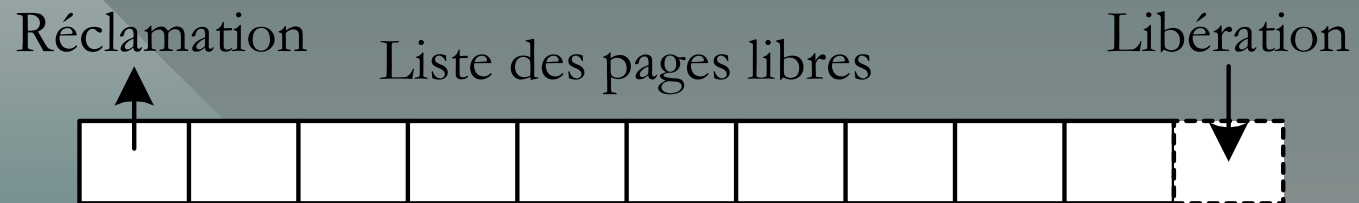
---

## ➤ Techniques de gestion:

- durant la vie d'un processus, des pages sont réclamées et d'autres sont libérées;
- pages réclamées → enlevées de la liste des pages libres;
- pages libérées → retournées dans la liste des pages libres;
- Protocole **LRU** (*Least Recently Used*) pour la réclamation et la libération des pages mémoire.

# Gestion de la mémoire

- Techniques de gestion (suite):
  - Protocole LRU.



Prendre les pages libres à la tête de la liste

Remettre les pages libérées à la fin de la liste des pages libres

Les pages réclamées seront celles qui ont été délaissées depuis longtemps.

# Gestion de la mémoire

---

- Lors de la lecture ou l'écriture (suite):
  - si la page n'est pas en mémoire, le S.E. vérifie la liste des pages libres;
  - si le nombre de pages libres est suffisant à la demande, le S.E. va associer une page libre dans l'espace virtuel;
  - s'il est insuffisant, le S.E. doit déplacer certaines pages utilisées hors de la mémoire puis les récupérer pour le processus demandeur.

# Gestion de la mémoire

---

- Lors de la lecture ou l'écriture (suite):
  - S.E. copie alors le contenu adressé dans la page allouée.
- Déplacement des pages pour faire de la place:
  - *swapper* (PID = 0) responsable des déplacements;
  - déplacement des pages vers le *swap space*;
  - *page daemon* (PID = 2) réalise la procédure de déplacement.

# Gestion de la mémoire

---

- Pour découvrir les pages à déplacer:
  - algorithme **THC** (*Two-handed Clock*).
- Algorithme THC:
  - deux variables;
  - $\text{min} = \text{QMPD} / 64$ ,  $\text{max} = \text{QMPD} / 16$ ;
  - QMPD  $\equiv$  Quantité de Mémoire Physique disponible.  
Comptabilisée lors du démarrage de la machine.
- Si le déplacement des pages n'est pas suffisant ...
- Il faut déplacer les processus au complet.
- C'est le *swapper* qui en est le responsable

# Gestion de la mémoire

---

## ➤ Algorithme THC:

- but → conserver toujours la mémoire disponible à l'intérieur des limites min et max;
- quatre (4) par seconde (250 ms) → algorithme THC est enclenché;
- si  $M_r + M_d < \text{max}$  alors un signal est envoyé au *page daemon* par le S.E. pour déplacer des pages hors de la mémoire.

Quantité de mémoire réclamée

Quantité de mémoire disponible

# Ordonnancement

---

- Classes d'ordonnancement:
  - permet aux programmeurs de déterminer le type d'ordonnancement de leurs applications;
  - mieux répondre aux applications nouvelles;
  - chaque classe possède des caractéristiques différentes;
  - chaque classe → priorité, durée d'exécution;
  - processus peut changer de classe par appels système.



# Ordonnancement

---

## ➤ Stratégie d'ordonnancement:

- priorité assignée initialement par le S.E. selon la classe du fils d'exécution;
- priorité peut varier au cours de l'existence du fils d'exécution;
- ordonnanceur choisit le premier fils le plus prioritaire pour passer le contrôle du processeur;
- ordonnanceur remet le fils exécuté à la fin de la queue de priorité appropriée.

# Ordonnancement

## ➤ Stratégie d'ordonnancement:

Nouvelle priorité = base + utilisation de l'U.C.

Base = 0 normalement ou plus si l'utilisateur décide de ralentir son processus en utilisant la commande nice

