



Système d'exploitation et Unix

Pr. Amal Ourdou



Plan

- **Système d'exploitation**
 - Introduction
 - Structure de système d'exploitation
 - Processus et threads
 - Gestion de la mémoire
 - Système de fichiers
- **Unix**
 - Introduction
 - Système Unix
 - Programmation Shell
 - Les scripts Shell

Systeme d'exploitation

Introduction

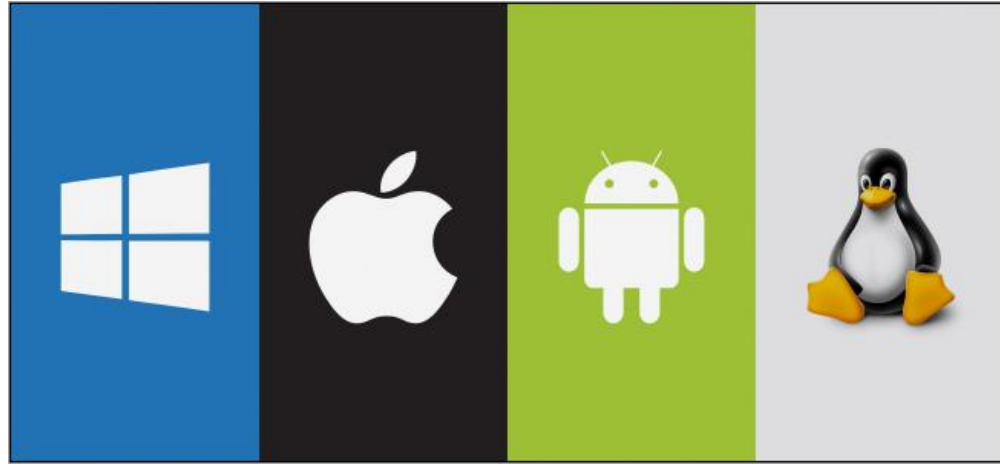


Introduction

- C'est quoi un système d'exploitation?
- Donner un exemple d'un système d'exploitation!

Introduction

- Un système d'exploitation (SE) est un programme qui gère le matériel de l'ordinateur (hardware).
- Il fournit également une base pour les programmes d'application et sert d'intermédiaire entre l'utilisateur et le matériel de l'ordinateur.



Word Spreadsheets Web browser ...

Programmes d'application

Système d'exploitation

Hardware

Ressources: CPU
memoire, I/O devices



Types des SE

- Les systèmes pour mainframes
- Les systèmes serveurs
- Les systèmes multiprocesseurs
- Les systèmes personnels
- Les systèmes temps réel

SE pour Mainframes





SE pour Mainframes

- Les mainframes sont des ordinateurs hautes performances dotés de grandes quantités de mémoire et de processeurs qui traitent des milliards de calculs et de transactions simples en temps réel. Le mainframe est essentiel pour les bases de données commerciales, les serveurs de transactions et les applications qui nécessitent une résilience et une sécurité élevées.
- Les ordinateurs centraux sont conçus pour traiter jusqu'à 1 trillion de transactions web par jour avec les plus hauts niveaux de sécurité et de fiabilité.
- L'accent est mis dans leurs systèmes d'exploitation sur la capacité à gérer de manière optimale plusieurs jobs en même temps, chacun d'eux demandant de grandes ressources d'E/S.



SE pour serveurs

- Un serveur est un ordinateur qui met des ressources, des données, des services ou des logiciels à la disposition d'autres ordinateurs, qualifiés de « clients », sur un réseau.
- En théorie, un ordinateur est considéré comme un serveur à partir du moment où il partage des ressources avec une machine cliente.
- Il existe de nombreux types de serveurs, notamment les serveurs web, serveurs de bases de données, les serveurs de messagerie, etc.



SE pour serveurs –suite–

- Ils servent en parallèle de nombreux utilisateurs à travers le réseau et permettent à ces derniers de partager des ressources matérielles et logicielles.
- Les fournisseurs d'accès à Internet et hébergeurs de sites web utilisent ces machines pour servir leurs clients.



SE pour multiprocesseur

Le principe de multiprocesseur est traduit par le recours à plusieurs CPU sur une même plateforme pour augmenter la puissance de calcul est une technique de plus en plus courante.



SE personnels

Leur rôle est de fournir à l'utilisateur une interface conviviale. Ils sont principalement dédiés à la bureautique, jeux et navigation Internet.



SE Temps réel

- Ces systèmes se caractérisent par le respect de contraintes temporelles.
- Un exemple typique est celui des systèmes d'ordinateurs qui contrôlent un processus de fabrication dans une usine (*comme le montage de véhicule*) : dans le cas d'un robot soudeur, une soudure faite trop tôt ou trop tard peut compromettre un véhicule.



Fonctionnalités d'un SE

- Il s'agit d'une interface entre l'utilisateur et le matériel
- Assurer le contrôle et le fonctionnement de la partie matérielle
- Réaliser la liaison entre l'utilisateur et le logiciel
- Orienter les entrées/sorties
- Allocation des ressources
- Gestion des utilisateurs et les droits d'accès
- Gestion de l'exécution des logiciels Gestion de la mémoire, de la sécurité, etc.



Les bases du système d'exploitation

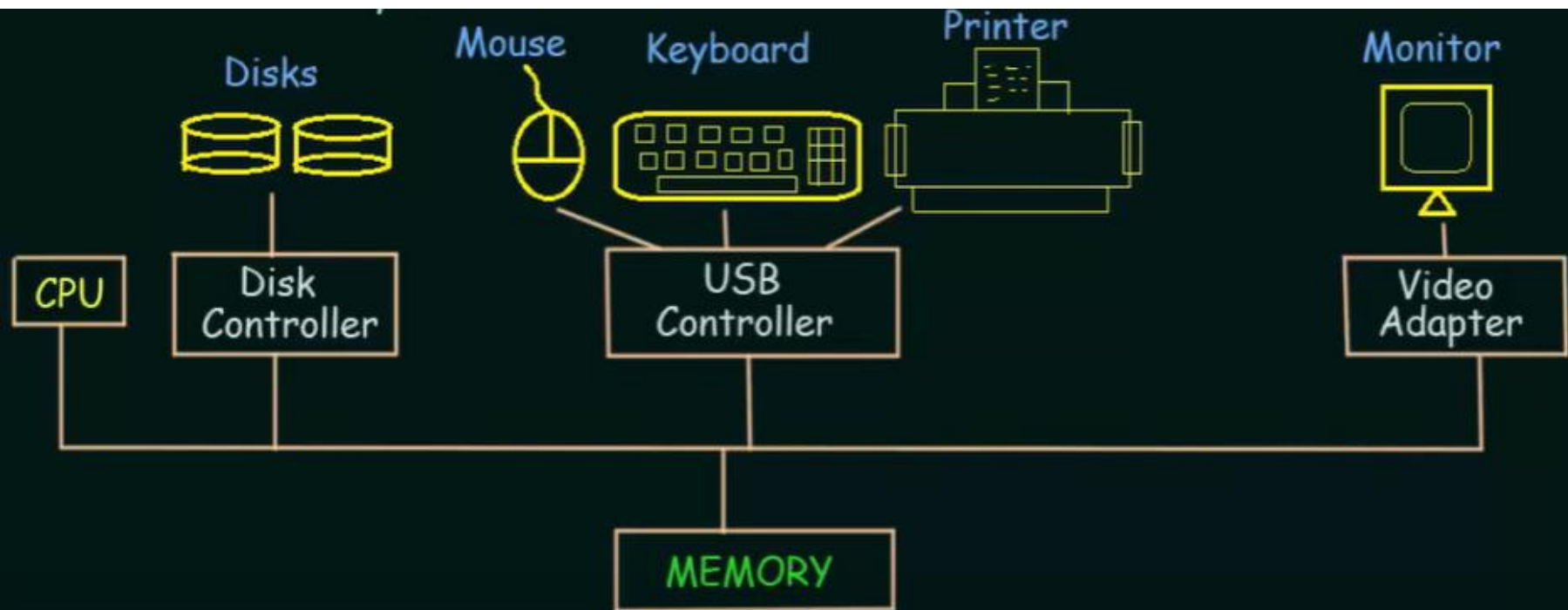
- Fonctionnement du système informatique
- Structure de stockage
- Structure I/O



Fonctionnement du système informatique

Quelques connaissances de base de la structure du système informatique sont nécessaires pour comprendre le fonctionnement du système d'exploitation.

- Un système informatique moderne polyvalent se compose d'une ou plusieurs unités centrales de traitement et d'un certain nombre de contrôleurs de périphériques connectés par un bus commun qui donne accès à une mémoire partagée.





Termes de bases

- **Bootstrap program** (programme d'amorçage –BIOS-):
 - Le programme initial qui s'exécute lorsqu'un ordinateur est mis sous tension ou redémarré.
 - Il est stocké dans la ROM
 - Il doit savoir comment charger le système d'exploitation et commencer à exécuter ce système.
 - Il doit localiser et charger en mémoire le noyau du système d'exploitation.



Termes de bases

- **Interrupt (interruption) :**
 - L'occurrence d'un événement est généralement due à une interruption matérielle ou logicielle.
 - Le matériel peut déclencher une interruption à tout moment en envoyant un signal à l'unité centrale, généralement par le biais du bus système.
- **System Call (appel système):**
 - Un logiciel peut déclencher une interruption en exécutant une opération spéciale appelée appel système.



Structure de stockage

- Registres
- Cache
- Mémoire principale
- Disk électronique
- Disk magnétique
- Disk optique
- Bandes magnétiques



Structure I/O

- Le stockage n'est qu'un des nombreux types de périphériques d'E/S dans un ordinateur.
- Une grande partie du code du système d'exploitation est consacrée à la gestion des E/S, en raison de son importance pour la fiabilité et les performances d'un système et en raison de la nature variable des périphériques.
- Un système informatique polyvalent se compose de cpus et de plusieurs contrôleurs de périphériques reliés par un bus commun.
- Chaque contrôleur de périphérique est responsable d'un type de périphérique spécifique.
- En général, les systèmes d'exploitation ont un pilote de périphérique (drivers) pour chaque contrôleur de périphérique.



Architecture d'un système informatique

- Single processeur systems
- Multiprocessor Systems
- Clustered systems



Single processor system

- Une unité centrale principale capable d'exécuter un jeu d'instructions générales, y compris des instructions provenant de processus utilisateur.
- D'autres processeurs à usage spécial sont également présents et exécutent des tâches spécifiques au périphérique.

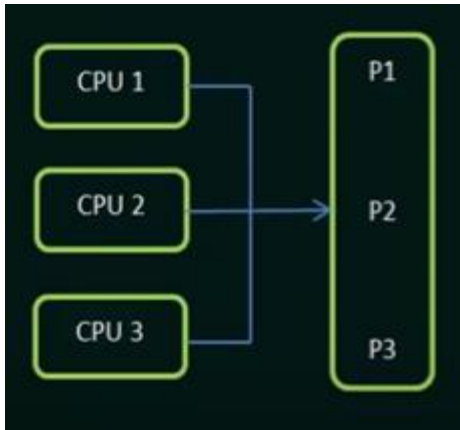


Multiprocessor system

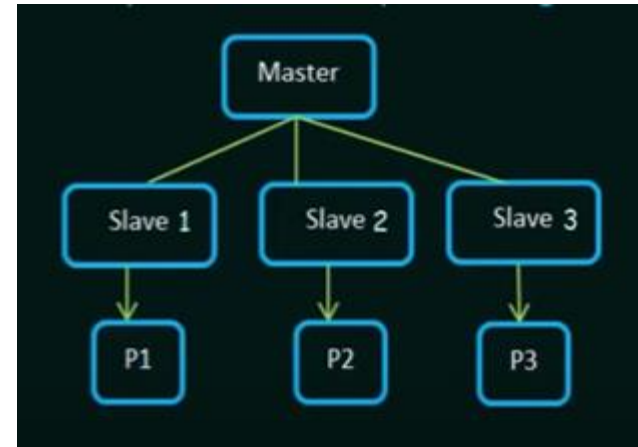
- Également connu sous le nom de parallel system or tightly coupled system
- possède deux processeurs ou plus en étroite communication, partageant le bus de l'ordinateur et parfois l'horloge, la mémoire et les périphériques.

Types des systèmes multiprocesseur

- Symétrique Multiprocessing



- Asymétrique Multiprocessing





Système en grappe (Clustered System)

- Tout comme les systèmes multiprocesseurs, les systèmes en grappe rassemblent plusieurs CPU pour accomplir des tâches de calcul.
- Ils sont composés de deux ou plusieurs systèmes individuels couplés entre eux.
- Ils offrent une haute disponibilité

Ils peuvent être structurés de manière:

- **Asymétrique**
 - Une machine en mode de secours à chaud
 - Les autres exécutent des applications
- **Symétrique**
 - Deux hôtes ou plus exécutent des applications
 - Se surveillent mutuellement

Structure de système d'exploitation

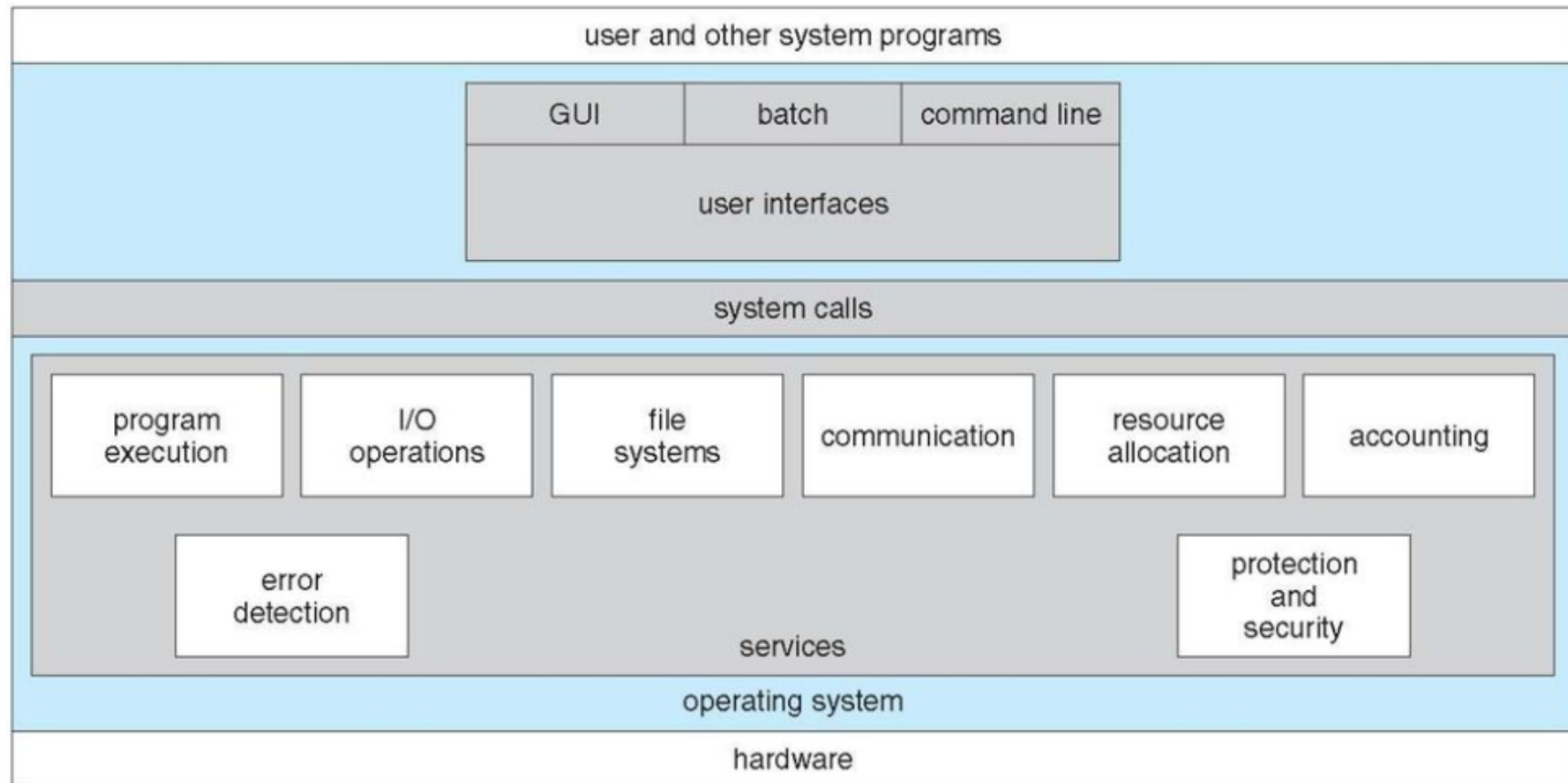


Structure d'un système d'exploitation

- Application
- Utilitaire | Librairies=> Système
- Noyau (kernel) => d'exploitation
- Matériel (CPU/ Mémoires/Périphériques)

Un utilitaire sert à assister l'utilisateur dans une activité. Il permet à l'utilisateur d'effectuer des manipulations basiques telles que démarrer des programmes, copier des fichiers ou modifier des paramètres de configuration

Les librairies regroupent les opérations souvent utilisées, selon les fonctionnalités (E/S, fichier, etc.)





Command Line (CLI)

- L'interface de ligne commande (CLI) ou l'interpréteur de commandes permet une entrée de commande directe
- Ça consiste principalement à récupérer une commande de l'utilisateur et l'exécuter
- Les commandes sont soit intégrées (implémentées dans le noyau), ou sont des noms de programmes systèmes



Graphical user interface (GUI)

- Interface conviviale « desktop »
- Habituellement souris, clavier et moniteur
- Icones représentent les fichiers, les programmes, les actions, etc.
- Divers boutons de la souris sur les objets de l'interface provoquent diverses actions (fournir des informations, des options, exécuter la fonction, ouvrir le répertoire, ...)



Les services

- Les service utilisateur
 - Exécution d'un programme
 - Operations d'entrées/sorties
 - Système de fichier
 - Communication
 - Détection d'erreurs
- Les services système
 - Allocation de ressources
 - Comptabilité
 - Protection et sécurité



Les services utilisateur

- Exécution d'un programme:

Program execution: Le système doit être capable de charger un programme en mémoire, de l'exécuter, terminer son exécution (normalement ou anormalement) et indiquer s'il y avait des erreurs lors de l'exécution

- Operations d'entrées/sorties

I/O operations: Un programme en cours d'exécution peut nécessiter des opérations d'E/S, ce qui peut impliquer un fichier ou un périphérique d'E/S

- Système de fichier

File systems: Le système de fichiers présente un intérêt particulier. Les programmes doivent lire et écrire des fichiers et des répertoires, les créer et les supprimer, les rechercher, lister les informations du fichier, gérer les permissions.



Les services utilisateur

- Communication

Communications: Les processus peuvent échanger des informations, sur le même ordinateur ou entre ordinateurs sur un réseau. Les communications peuvent être effectuées via la mémoire partagée ou via le transfert de messages (paquet déplacés par le système d'exploitation)

- Détection d'erreurs

Error detection: Le système d'exploitation doit être capable de détecter, gérer et anticiper les différentes erreurs susceptibles de se produire dans le cpu, la mémoire, les périphériques d'E/S ou dans les programmes utilisateurs



Les services système

- Allocation de ressources

Resource allocation: Lorsque plusieurs utilisateurs ou plusieurs jobs s'exécutent simultanément, des ressources doivent être allouées à chacun d'entre eux comme: les cycles cpu, RAM, périphériques d'E/S, etc

- Comptabilité

Accounting: Pour garder une trace de qui (utilisateur) utilise quoi (ressource) et quand est-ce qu'il l'utilise et combien de fois, etc.

- Protection et sécurité

Protection and security: Protéger les informations stockées des accès non permis, gérer les processus concurrents, etc.



Appel système

- Un Appel Système a pour rôle d'activer le système d'exploitation en procédant comme suit:
 1. Changer de mode d'exécution pour passer du mode utilisateur au mode noyau
 2. Récupérer les paramètres et vérifier la validité de l'appel
 3. Lancer l'exécution de la fonction demandée,
 4. Récupérer la (les) valeur(s) de retour et retourner au programme appelant avec retour au mode utilisateur.



Appel système

Les types d'appels système :

- Contrôle de processus
- Gestion des appareils
- Communication
- Protection



Les types d'appels système

- Contrôle de processus
 - Créer un processus, mettre fin à un processus
 - Avorter un processus
 - Charger, exécuter
 - Obtenir/définir des attributs de processus
 - Attendre pour un certain temps
 - Attendre/signaler un événement
 - Gestion de la mémoire
 - Débogage
 - Verrous pour gérer l'accès aux données partagées entre les processus
 - etc



Les types d'appels système

- Gestion des fichiers
 - Créés/supprimer les fichiers
 - Ouvrir/fermer les fichiers
 - Lire, écrire, repositionner
 - Obtenir et définir les attributs de fichier



Les types d'appels système

- Gestion des appareils
 - Demande, libération de l'appareil (request / release)
 - Lire, écrire, repositionner
 - Obtenir/définir les attributs de l'appareil
 - Connecter ou détacher des périphériques



Les types d'appels système

- Communication
 - Créer/supprimer les connexions
 - Transférer les informations d'état
 - Envoyer, recevoir des messages (communication par le modèle de passage de message)
 - Créer et accéder aux régions de la mémoire (communication par le modèle à mémoire partagée)



Les types d'appels système

- Protection
 - Contrôler l'accès aux ressources
 - Obtenir et définir des autorisations
 - Autoriser et refuser l'accès des utilisateurs à certaines ressources



Exemples d'appels système

	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()



Noyau (Kernel)

- Le Noyau ou Kernel en anglais, représente les fonctions fondamentales du système d'exploitation telles que:
 - Communication entre processus.
 - Gestion de la mémoire.
 - Traitement des interruptions.
 - Etc.
- Le noyau est le logiciel qui gère l'accès du programme utilisateur aux ressources logicielles et matérielles de l'ordinateur.
- C'est la seule partie du SE résidente toujours en RAM, les autres parties sont amenées en RAM au besoin.



Noyau

En général, les processeurs ont deux modes de fonctionnement :

- **Le mode Noyau** (superviseur , privilégié ou maître) : où toutes les instructions sont autorisées.
 - **Le mode utilisateur** (esclave) : où certaines instructions ne sont pas permises.
-
- Le système d'exploitation utilise cette propriété pour faciliter le contrôle qu'il exerce : il s'exécute en mode noyau alors que tous les autres programmes sont exécutés en mode utilisateur.
 - Ces programmes utilisateur ont ainsi des pouvoirs limités et certaines opérations leurs sont interdites



Mode Noyau VS Mode utilisateur

- Comme un *programme utilisateur* ne peut pas accéder à certaines ressources de la machine, il doit donc faire explicitement **appel aux services** du système d'exploitation pour accéder à ces ressources.
- À ces fins, le SE propose une interface de programmation API (**Application Programming Interface**) qui permet d'accéder à un certain nombre de fonctionnalités qu'il exécutera pour l'utilisateur.
- **Les Appels Système** sont l'interface proposée par le système d'exploitation pour accéder aux différentes ressources de la machine.



Processus



Processus

- Processus
 - Définition
 - Etats d'un processus
 - Ordonnancement



Processus

Définitions

- Un processus est un programme ou une commande en cours d'exécution sur un système.
- Un processus **est l'entité dynamique** représentant l'exécution d'un programme sur un processeur.
- Processus = **unité d'exécution** (unité de partage du temps processeur et de la mémoire)



Processus: Structure de gestion

- Un processus est obtenu à partir de l'exécution d'un fichier exécutable.
- Les informations présentes dans ce fichier sont alors utilisées pour effectuer les différentes étapes nécessaires à l'exécution :
 - l'allocation de la mémoire pour stocker le code et les données,
 - l'allocation et l'initialisation de données utilisées par le système d'exploitation pour gérer les processus.



Processus: Structure de gestion

- Le SE permet d'attribuer des zones mémoire à chaque processus et on dit parfois, par abus de langage, que chaque processus travaille dans son propre espace mémoire.
- Généralement, le programme d'un processus peut être partagé avec d'autres processus, mais les données ne sont en revanche accessibles que par le processus propriétaire.
- Les processus doivent alors utiliser des communications particulières, gérées par le système d'exploitation, pour échanger des données.



Processus: Structure de l'espace mémoire

L'espace mémoire utilisé par un processus est divisé en plusieurs parties .

On trouve en particulier :

- Le code
- Les données
- PCB (process control block)



PCB (process control block)

Le PCB est une structure de données du noyau d'un système d'exploitation représentant l'état d'un processus donné.

Diverses implémentations existent selon les systèmes d'exploitation, mais un PCB contient en général :

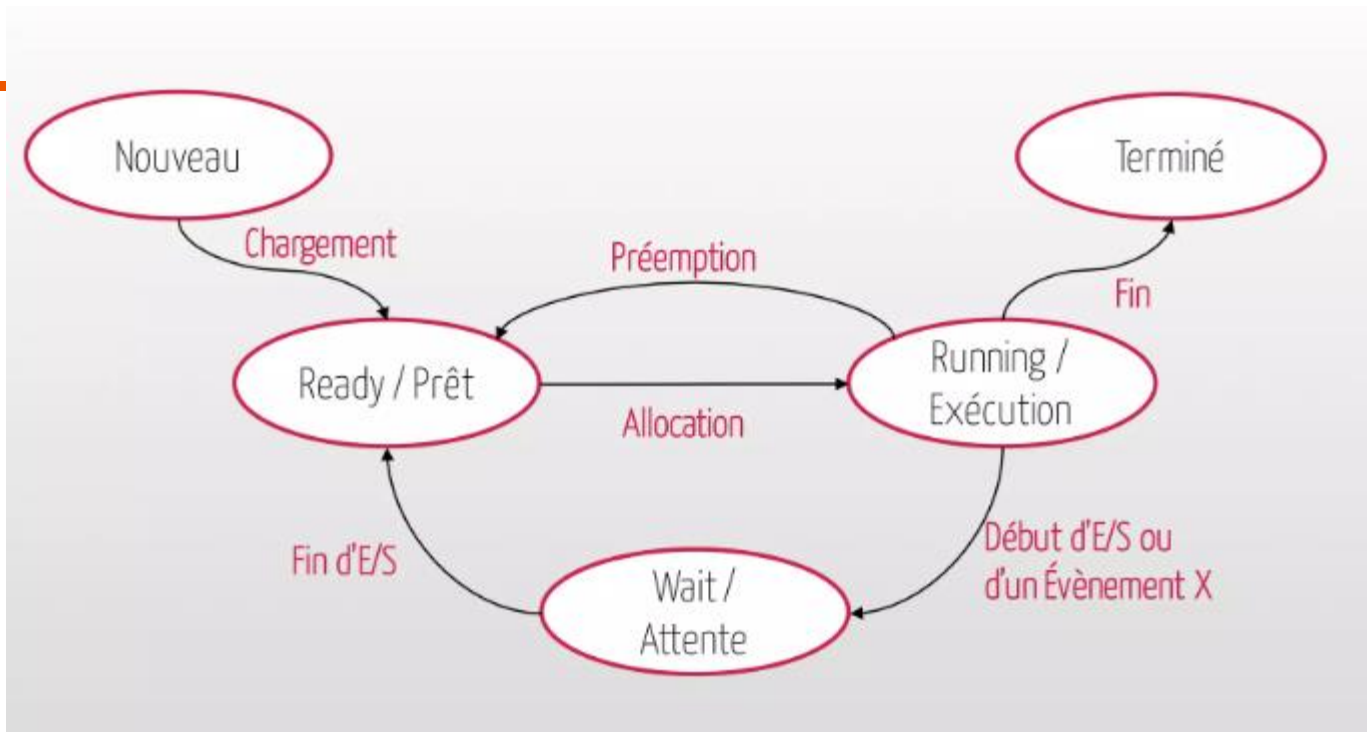
- L'ID du processus (PID), l'ID du processus parent (PPID) et l'ID de l'utilisateur du processus (UID) ;
- Les valeurs des registres correspondant au processus (l'état courant du processus) ;
- Le pointeur de pile : indique la position du prochain emplacement disponible dans la pile mémoire ;
- L'espace d'adressage du processus ;
- D'autres informations telles que le temps CPU accumulé par le processus, etc.



PCB

Lors d'un changement de contexte, le processus en cours est arrêté et un autre processus peut utiliser le CPU. Le noyau doit arrêter l'exécution du processus en cours, copier les valeurs des registres hardware dans le PCB, et mettre à jour les registres avec les valeurs du nouveau processus.

Une commutation de contexte (changement de contexte) consiste à sauvegarder l'état d'un processus pour restaurer à la place celui d'un autre dans le cadre de l'ordonnancement d'un système d'exploitation multitâche.





Processus: Ordonnancement

Ordonnancement \Leftrightarrow la stratégie de planification et d'attribution des ressources aux processus qui en font la demande.

- Plusieurs processus sont prêts à être exécutés, le SE doit faire un choix.
- La partie du SE qui effectue ce choix est l'ordonnanceur (Scheduler):
 - C'est un programme intégré au noyau.
 - Il définit l'ordre dans lequel les processus prêts utilisent l'UC et la durée d'utilisation, en utilisant un algorithme d'ordonnancement.



Processus: Ordonnancement

Un bon algorithme d'ordonnancement doit posséder les qualités suivantes :

- **Rendement maximal** : l'ordinateur doit exécuter le maximum de programmes en un temps donné
- **Temps moyen d'exécution minimal** : chaque programme doit s'exécuter le plus vite possible
- **Temps de réponse minimal** : les programmes interactifs doivent avoir un faible temps de réponse
- **Équité** : chaque programme a droit au temps processeur



Processus: Ordonnancement

- **Temps de réponse:** C'est le temps moyen qu'un processus passe à attendre. Il s'obtient en soustrayant la durée d'exécution du processus de sa durée de rotation. Cette durée est indépendante de la durée d'exécution du processus lui-même.
- **Temps de rotation:** C'est le délai moyen entre l'admission du processus et la fin de son exécution.
- **Temps moyen d'exécution** d'un ensemble de tâches : la moyenne des intervalles de temps séparant la soumission d'une tâche de sa fin d'exécution (moyenne des temps de rotation).



Exemple (temps d'exécution)

Soient 4 jobs A,B,C et D arrivés au même moment avec des temps d'exécution respectifs de 8,4,4,4 minutes, exécutés dans cet ordre.

- Le temps de rotation (soumission → fin) de A est 8, celui de B est 12, celui de C est 16 et celui de D est 20,
- Pour un temps moyen de $14 = (8+12+16+20) / 4$



Algorithme Ordonnancement


Il existe deux familles d'algorithmes :

- **Sans réquisition (ASR) = non préemptif :**
 - Un processus est exécuté jusqu'à la fin
 - Le choix d'un nouveau processus ne se fait que sur blocage ou terminaison du processus courant
 - Inefficace et dangereux (ex: exécution d'une boucle sans fin...)
- **Avec réquisition (AAR) = préemptif :** à intervalle régulier, l'Ordonnanceur reprend la main et élit un nouveau processus actif



Algorithme Ordonnancement

- FCFS: First Come First Served: Premier arrivé premier servi, en gérant une file des processus.
- SJF: Shortest Job First: dans cet algorithme on suppose que les temps d'exécution sont connus à l'avance.
- Shortest Remaining time (SRT): la version préemptive de SJF: Si un processus dont le temps d'exécution est plus court que le reste du temps d'exécution du processus en cours de traitement, alors il prendra sa place.
- Priorité: le choix du processus à élire dépend des propriétés des processus prêts
- Round Robin (Tourniquet): chaque processus dispose d'un quantum de temps pendant lequel il est autorisé à s'exécuter



Ordre d'arrivée	T1	T2	T3	T4	T5	T6	T7
Durée	7	4	6	1	2	4	1
Date d'arrivée	0	0	1	1	1	2	2
Priorité (n)	2	3	1	2	3	1	2



Exercice 1

Sur un CPU, l'ordonnanceur gère l'ordonnancement des processus par un tourniquet avec un quantum de 100 ms. Sachant que le temps nécessaire à une commutation de processus est de 10 ms, calculer le temps de rotation pour :

Ordre d'arrivée des tâches	T1	T2	T3	T4	T5	T6	T7
Durée	700	400	600	100	200	400	100
Date d'arrivée	0	0	100	100	150	200	200



Exercice 2

Considérez un système d'exploitation qui ordonnance les processus selon l'algorithme du tourniquet. Supposez que le système d'exploitation est composé de deux unités de contrôle (deux processeurs CPU1 et CPU2) et d'une unité d'E/S. Chaque processeur exécute l'algorithme du tourniquet avec un quantum de trois unités de temps ($qt = 3$). Tous les processus prêts sont dans une même file d'attente. La commutation de contexte est supposée de durée nulle. Considérez trois processus A, B et C décrits dans le tableau suivant

Processus	Instant d'arrivée	Temps d'exécution
A	0	4 unités de CPU, 2 unités d'E/S, 2 unités de CPU
B	2	3 unités de CPU, 4 unités d'E/S, 2 unités de CPU
C	3.5	5 unités de CPU

Si plusieurs événements surviennent en même temps, vous supposerez les priorités suivantes :

- Le CPU1 a la priorité d'accès à la file des processus prêts par rapport au CPU2.
- A la fin d'un quantum, le processus non terminé en cours est suspendu uniquement si la file des processus prêts n'est pas vide. Le traitement réalisé à la fin d'un quantum est plus prioritaire que celui d'une fin d'E/S qui, à son tour, est plus prioritaire que l'arrivée de nouveaux processus dans le système.



Threads



Thread

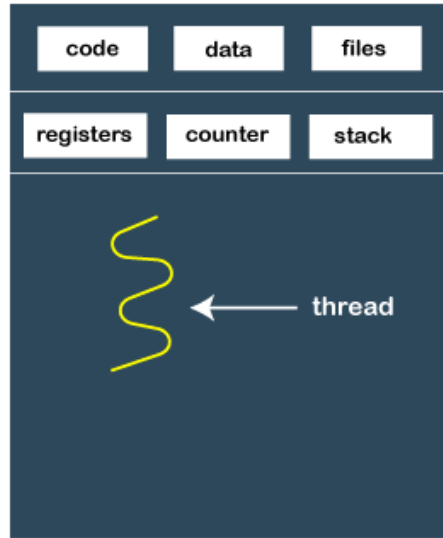
- Un thread est une **unité d'exécution ou de traitement** composée d'un ensemble d'instructions contenues dans le processus.
- Un thread est une unité rattachée à un processus, chargé d'exécuter une partie du programme du processus
- De nombreux systèmes d'exploitation offrent la possibilité d'associer à un même processus plusieurs chemins d'exécution (**multithreading**)



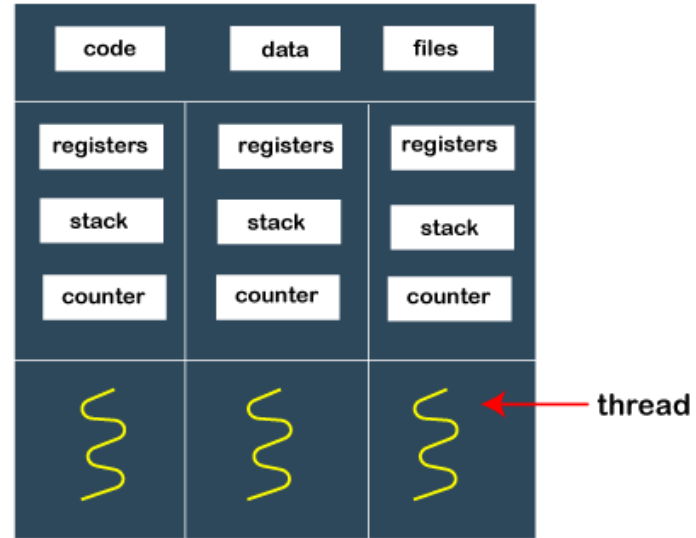
Threads

- Lorsqu'un processus est créé, un seul fils d'exécution (thread) est associé au processus:
- Le thread principal exécutant la fonction main du programme du processus
- Ce fils principal peut en créer d'autres

Threads



Single-threaded process

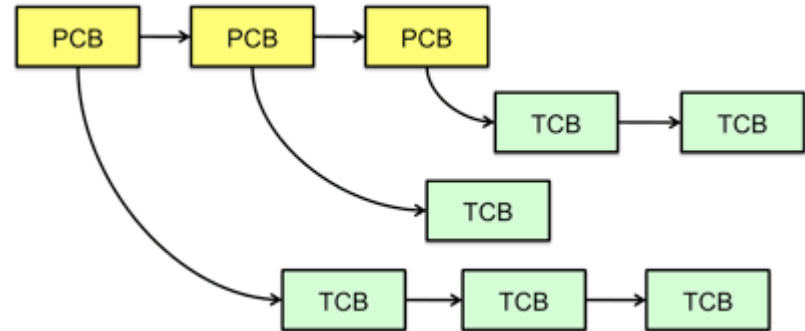


Multi-threaded process

Threads

- états unique pour chaque thread, chaque fil possède un **TCP** (**thread Control Block**)

Thread ID
Thread state
CPU information : Program counter Register contents
Thread priority
Pointer to process that created this thread
Pointer(s) to other thread(s) that were created by this thread

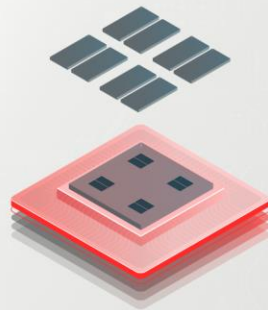


Threads

What Are Logical Processors?



Logical Processors are virtualized processors that the operating system can see



8 Logical Processors
(Typically 2 per core)

4-Core/8-Threads
Physical processor

Multi-Threaded Processor



4 Logical Processors
(1 per core)

4-Core/4-Thread
Physical processor

Single-Threaded Processor

Gestion de la mémoire



Gestion de la mémoire

- Introduction
- Gestion de la mémoire centrale
 - Allocation contiguë
 - Partitions fixes
 - Partitions dynamiques
 - Allocation non contiguë
 - Pagination
 - Segmentation



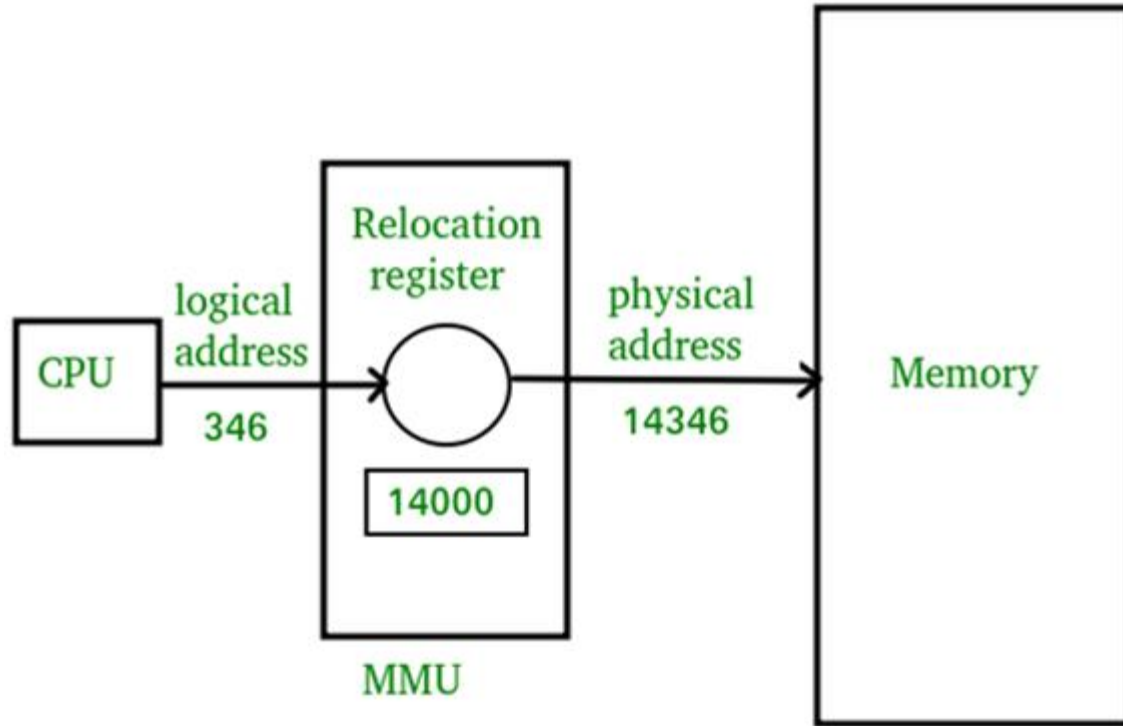
Introduction

- La partie du SE qui s'occupe de la gestion de la mémoire s'appelle: le gestionnaire de mémoire
=> Il se charge de gérer l'allocation de l'espace mémoire aux processus:
 - ☐ Comment organiser la mémoire physique? (Partitions fixes ou variables)
 - ☐ Parmi les parties libres en mémoire, lesquelles allouer au processus ? (politique de placement)
 - ☐ S'il n'y a pas assez d'espace en mémoire, doit on libérer de l'espace en retirant des parties ou des processus entiers ? Si oui lesquels ? (Politique de remplacement)
 - ☐ Les adresses figurant dans les instructions sont elles relatives (logiques) ? Si oui, comment les convertir en adresses physiques.



Mémoire Physique et mémoire Logique

- **Mémoire physique:**
 - La mémoire principale RAM de la machine
 - Adresses physiques: les adresses de cette mémoire
- **Mémoire logique:** l'espace d'adressage d'un programme
 - Adresses logiques: les adresses dans cet espace
- Il faut séparer ces concepts car normalement, les programmes sont chargés de fois en fois dans des positions différents de la mémoire
 - Donc adresse physique != adresse logique





Memory Management Unit

- MMU (Memory Management Unit) est un dispositif matériel qui réalise la conversion des adresses logiques en adresses physiques
- La MMU est généralement située dans le CPU, mais fonctionne parfois dans une puce intégrée (IC) distincte.
- Toutes les entrées de demande de données sont envoyées à la MMU, qui à son tour détermine si les données doivent être récupérées de la mémoire RAM ou ROM.



Gestion de la mémoire centrale

- Lorsque l'utilisateur demande l'exécution d'un programme, le chargeur doit trouver une place libre suffisamment grande dans la zone des programmes utilisateurs.
- Une fois l'exécution de programme terminée, la Ram doit être libérée.
- Il existe plusieurs stratégies d'allocation/libération de la RAM:
 - Allocation contiguë
 - Allocation non contiguë: Le processus est subdivisé dans la RAM, et un mécanisme est défini pour retrouver les différentes briques une fois le programme chargé



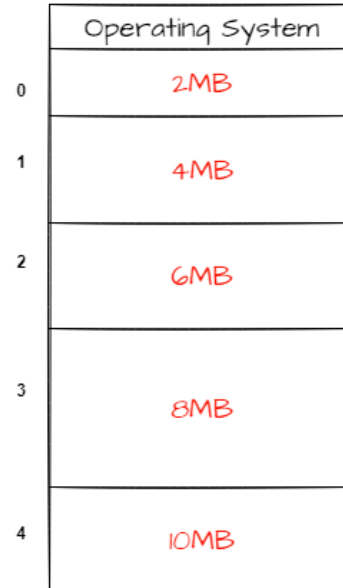
Allocation contiguë

Le processus est chargé en entier dans la mémoire, dans un bloc contiguë:

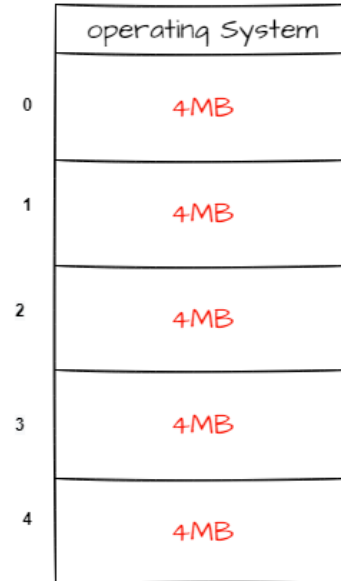
- Partitions fixes
- Partition dynamique

Allocation Contiguë: Partitions Fixes

- La mémoire principale est divisée en zone de taille fixe qui ne se chevauchent pas
- Chaque zone est appelée partition et peut être de taille égale ou non
- La taille des partitions est déterminée au départ et ne change pas durant l'exécution des programmes
- Un algorithme de placement est en charge de trouver une partition libre pour charger les processus en mémoire
- Chaque partition peut contenir un seul processus
- Un programme doit tenir sur une seule partition



RAM





Allocation Contiguë: Partitions Fixes

- Tout processus dont la taille est inférieure ou égale à la taille des partitions peut être chargé en mémoire
- Si toutes les partitions sont occupées, le SE peut déplacer un processus d'une partition vers la mémoire secondaire (swapping)



Allocation Contiguë: Partitions Fixes

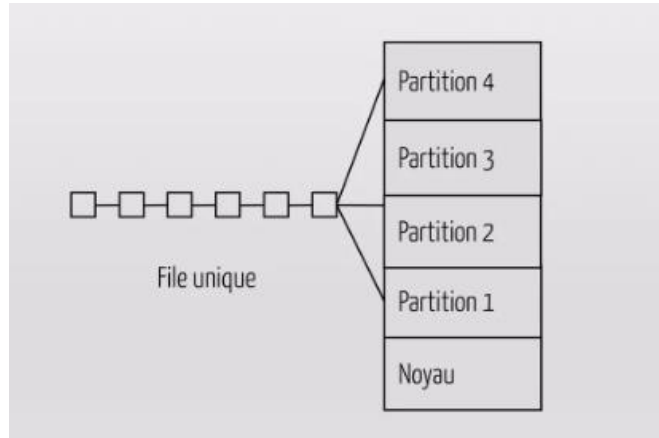
Inconvénient:

L'utilisateur de la mémoire est inefficace avec cette méthode.

- ⇒ Donc on perd tous l'espace inoccupés par ces partitions
- ⇒ Problème de fragmentation: Il y'a assez d'espace pour exécuter un programme, mais cet espace est fragmenté de façon non contiguë

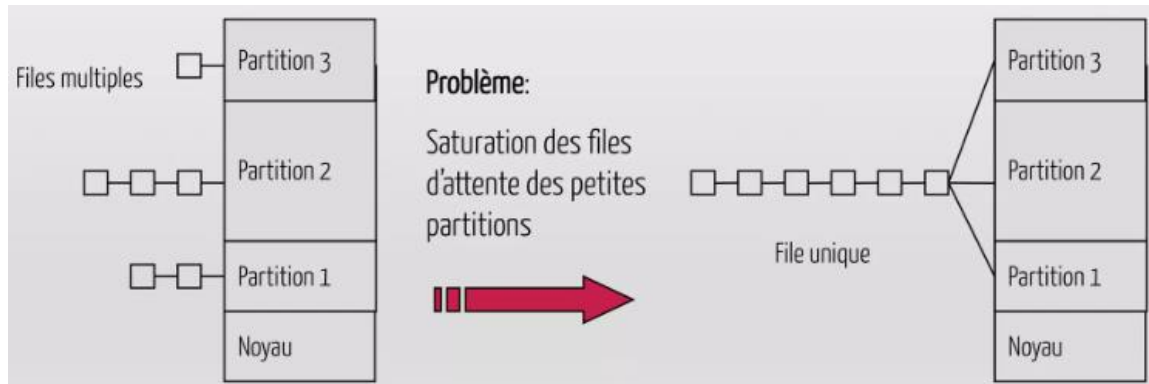
Partitions fixes de tailles égales

- Diviser la mémoire en N partitions de même taille
- Chaque nouvelle tâche est placée dans la première partitions libre
- Problèmes:
 - La taille de partition doit être assez grande pour n'importe quel processus
 - Gros problème de fragmentation



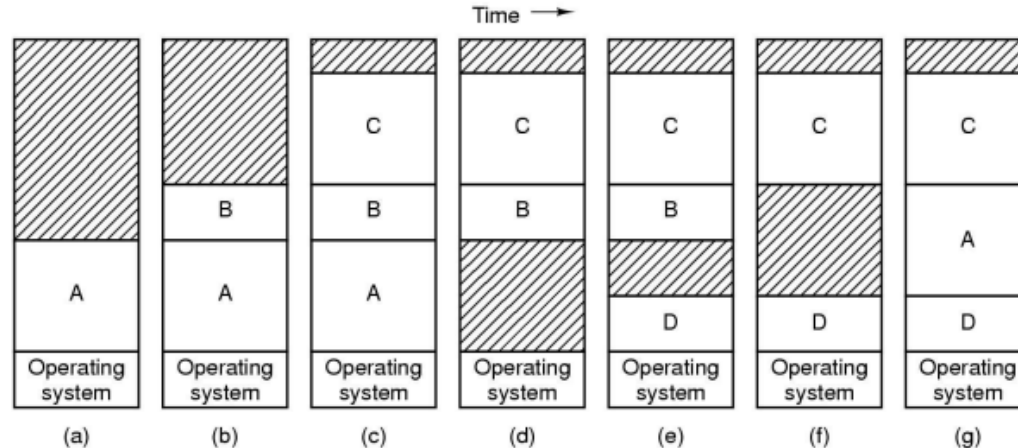
Partitions fixes de tailles inégales

- Diviser la mémoire en N partitions qui peuvent être de tailles inégales et associer à chaque partitions une file d'attente
- Chaque nouvelle tâche est placée dans la file d'attente de la plus petite partitions qui peut la contenir
- L'espace non utilisé dans une partition est perdu (pb de fragmentation)



Allocation contiguë: Partitions Dynamiques

- L'idée est de créer de partitions dynamique selon les tailles des processus
- Les partitions sont créées a l'exécution des processus. Elle sont allouées de la taille correspondant aux besoin du processus

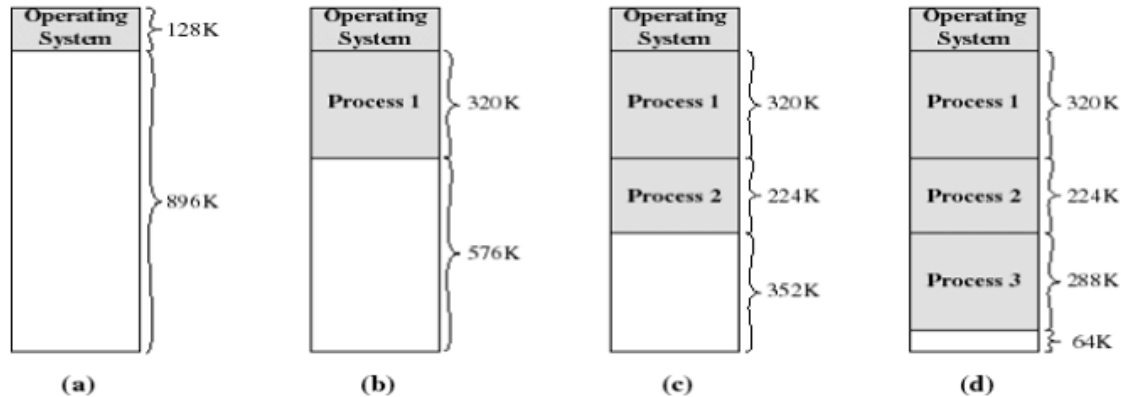




Allocation contiguë: Partitions dynamiques

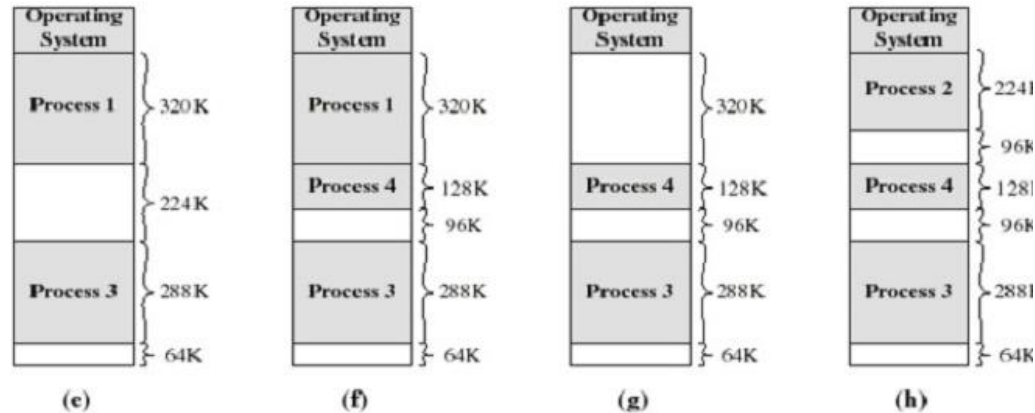
- Cette méthode peut entraîner un problème de fragmentation entre les partitions => Fragmentation externe
- On risque d'avoir plusieurs fragment vide entre les partitions, et s'il y a un processus de grande taille, il risque de ne pas pouvoir y entrer. Dans ce cas on aura besoin de défragmenter la mémoire => cela consomme trop de CPU et risque de ralentir la machine.

Partition dynamique: Exemple



- Mémoire = 1024K, OS= 128K, P1 = 320K, P2= 224K, P3=228K, P4=128K
- Il y a un trou de 64K après avoir chargé 3 processus: pas assez d'espace pour d'autre processus
- Si tous les processus se bloquent (ex: attente d'un événement), P2 peut être permuté et P4 peut être chargé

Partition dynamique: Exemple



- (e-f) P2 est suspendu, P4 est chargé. Un trou de $224 - 128 = 96\text{K}$ est créé (fragmentation externe)
- (g-h) P1 se termine ou il est suspendu, P2 est repris à sa place: produisant un autre trou de $320 - 224 = 96\text{K}$
- Nous avons 3 trous petits et probablement inutiles
- Compression pour en faire un seul trou de 256K

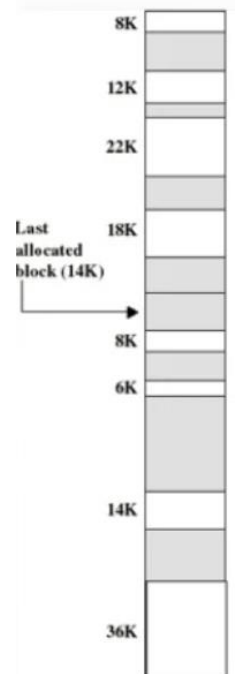


Allocation Contiguë: Algorithmes de placement

- Pour décider l'emplacement du prochain processus
- But: réduire l'utilisation de la compression qui prend du temps
- Choix possible:
 - Best fit: Choisir le plus petit trou
 - Worst fit: le plus grand
 - First fit: choisir 1^{er} trou à partir du début
 - Next fit: choisir le 1^{er} trou à partir du dernier placement

Allocation Contiguë: Algorithmes de placement

- Selon chacun des algorithmes précédent, ou sera placé un nouveau block de 16K?





Exercice 1

- Soit une mémoire centrale composée de 5 partitions P1, P2, P3, P4 et P5; ces partitions ont pour tailles respectives 100, 500, 200, 300 et 600 K (dans cet ordre).
- Soient 4 processus A, B, C et D de tailles respectives 212, 417, 112 et 426 K.
- Donner les différents états de la mémoire centrale (sous forme de schémas) pour charger les processus A, B, C et D (dans cet ordre) en utilisant les algorithmes d'allocation suivants :
 - a) FIRST-FIT
 - b) BEST-FIT
 - c) WORST-FIT
 - d) NEXT-FIT



Exercice 2

- On considère une mémoire centrale de taille 1000, dans laquelle on utilise une allocation contiguë. Soit la suite des demandes suivantes ; (+) signifie une demande d'allocation, alors que (-) signifie une demande de libération.
+A(300), +B(260), +C(200), -B, +D(100), -A, +E(250), +F(300), -F, -D, -C, -E
- Indiquer comment à partir d'une mémoire initialement libre, le système traite ces demandes, en utilisant les stratégies suivantes :
 - a) FIRST-FIT
 - b) BEST-FIT
 - c) WORST-FIT



Allocation non contiguë

- Afin de réduire la fragmentation, tous les systèmes d'aujourd'hui utilisent l'allocation non contiguë
 - Diviser un programme en morceaux et permettre l'allocation séparée de chaque morceau
 - Les morceaux sont beaucoup plus petits que le programme entier et donc permettent une utilisation plus efficace de la mémoire (les petits trous peuvent être utilisés plus facilement)
- Il y a deux techniques de base pour faire ceci: la pagination et la segmentation
 - La segmentation utilise des parties de programme qui ont une valeur logique (des modules)
 - La pagination utilise des parties de programme arbitraire (morcellement du programmes en pages de longueur fixe).
 - Elles peuvent être combinées



Allocation non contiguë: Pagination

- La mémoire est partitionnée en petits morceaux de même taille (pages physiques ou frames)
- Chaque processus est aussi partitionné en petits morceaux de même taille appelées pages logiques (pages)
- Les pages logiques d'un processus peuvent donc être assignées aux frames disponibles n'importe où en mémoire principale
- Conséquences:
 - Un processus peut être éparpillé n'importe où dans la mémoire physique
 - La fragmentation externe est éliminée

Pagination: Exemple

Frame number	Main memory
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	

(a) Fifteen Available Pages

Frame number	Main memory
0	A.0
1	A.1
2	A.2
3	A.3
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	

(b) Load Process A

Frame number	Main memory
0	A.0
1	A.1
2	A.2
3	A.3
4	B.0
5	B.1
6	B.2
7	
8	
9	
10	
11	
12	
13	
14	

(b) Load Process B

Frame number	Main memory
0	A.0
1	A.1
2	A.2
3	A.3
4	B.0
5	B.1
6	B.2
7	C.0
8	C.1
9	C.2
10	C.3
11	
12	
13	
14	

(d) Load Process C

Pagination: Exemple

- Supposons que le processus B se termine ou est suspendu
- Nous pouvons maintenant transférer en mémoire un processus D, qui demande 5 frames (bien qu'il n'y ait pas 5 cadres contigus disponibles)
- La fragmentation externe est limitée au cas que le nombre de pages disponibles n'est pas suffisant pour exécuter un processus en attente
- Seule la dernière pages d'un processus peut souffrir de fragmentation interne

Main memory		Main memory	
0	A.0	0	A.0
1	A.1	1	A.1
2	A.2	2	A.2
3	A.3	3	A.3
4		4	D.0
5		5	D.1
6		6	D.2
7	C.0	7	C.0
8	C.1	8	C.1
9	C.2	9	C.2
10	C.3	10	C.3
11		11	D.3
12		12	D.4
13		13	
14		14	

Pagination: Table de page

- Pour pouvoir convertir l'adresse logique en adresse physique, chaque processus doit conserver sa propre table de pages
- Cette table est placée dans le registre PTBR (Page-Table base Register)

page 0
page 1
page 2
page 3

mémoire
logique

0	1
1	4
2	3
3	7

Table des pages

0	
1	page 0
2	
3	page 2
4	page 1
5	
6	
7	page 3

mémoire
physique

Pagination: Table de page

- Le SE doit maintenir une table de pages pour chaque processus
- Chaque table de page contient le numéro de frame où la page correspondante est physiquement localisée
- Une table de pages est indexée par le numéro de la page afin d'obtenir le numéro du cadre
- Une liste de cadres disponibles est également maintenue (free frame list)

0	0
1	1
2	2
3	3

Process A
page table

0	—
1	—
2	—

Process B
page table

0	7
1	8
2	9
3	10

Process C
page table

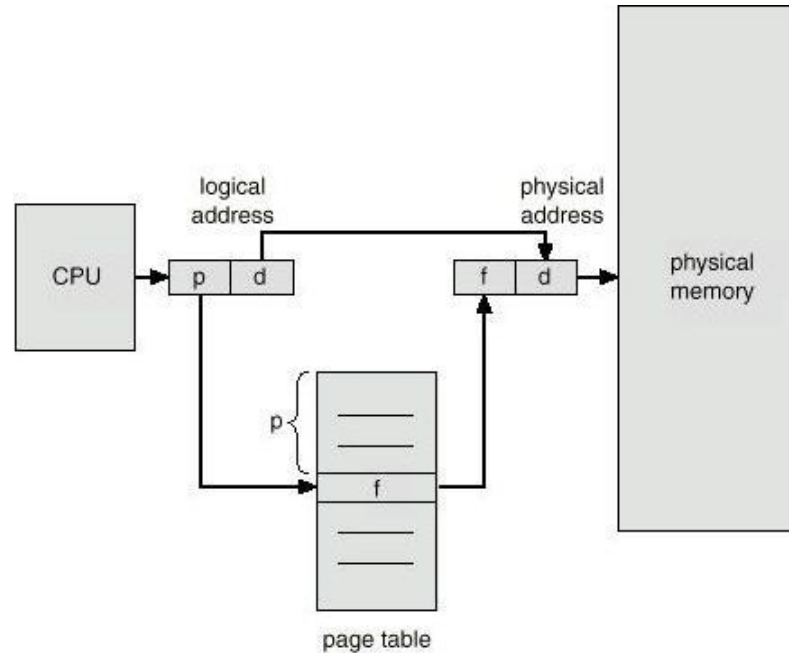
0	4
1	5
2	6
3	11
4	12

Process D
page table

13
14

Free frame
list

Pagination: Traduction d'adresses





Pagination: Validation

- Pour que le cpu puisse savoir si une page est chargée en RAM, un bit de validation V est ajouté aux table de pages (il est à 1 quand la page est chargée en RAM)
- Que se passe-t-il si on veut accéder à une page non chargée à la mémoire?
- Si la mémoire est pleine, quelle page décharger?



Pagination: Défaut de page

- Un défaut de page se produit lorsqu'un processus veut accéder à une page non présente en mémoire centrale
- À l'occurrence d'un défaut de page, un déroulement est réalisé
- Le système d'exploitation charge la page manquante dans une case vide
- Si toutes les cases sont pleines, il doit d'abord vider une case (la mettre en mémoire auxiliaire) puis charger la nouvelle page

Exemple

- Hypothèses:
 - Bus d'adresses de 16 bits
 - Taille d'une page 4ko = 4096 o
- Question
 - Calculer l'adresse physique correspondante à l'adresse logique (virtuelle) 8196 o

15	000
14	000
13	000
12	000
11	111
10	000
9	101
8	000
7	000
6	000
5	011
4	100
3	000
2	110
1	001
0	010



Exemple

- Soit un ordinateur ayant
 - Une mémoire physique de capacité 128 Mo
 - Une architecture 32 bits
 - La taille d'une page égale à 4Ko
- Donner la taille de la table des pages
 - Chaque entrée de la table : référence d'un cadre + bit présence



Exercice 1

- Calculer les adresses physiques qui correspondent aux adresses logiques suivantes: 1035, 4093, 4098, 9000. Le mode de gestion est la pagination et les pages ont une taille de 4KO.

Numéro de la page	Numéro de la case
0	4
1	6
2	15
3	10



Exercice 2

Soit la table de pages ci-après. Sachant que les pages virtuelles et physiques font 1K octets:

1. Quelle est l'adresse mémoire correspondant à l'adresse virtuelle suivante : 142A
2. Quelle est l'adresse logique correspondant à l'adresse physique suivante : 22F1

0	4
1	6
2	8
3	9
4	12
5	1



Algorithmes de remplacement de page

Les algorithmes de remplacement de pages sont (entre autres)

- **L'algorithme First In First Out (FIFO)**
 - Remplacer la page la plus ancienne dans la mémoire
- **L'algorithme Least Recently Used (LRU)**
 - Remplacer la page qui est restée inutilisée le plus de temps
- **L'algorithme optimal**
 - Retirer de la mémoire page qui sera référencée le plus tard. C'est un algorithme théorique car il nécessite la connaissance des requêtes futures



Exercice 3

Nous utiliserons la suite de pages suivante :

7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

et 3 pages en mémoire centrale. Schématiser à chaque étape le contenu de la liste de remplacement et préciser pour chaque algorithme le nombre de défauts de pages et les pages qui en sont responsable. Les algorithmes utilisés sont : l'algorithme optimal, FIFO et LRU (Least Recently Used),



Allocation non contiguë: Segmentation

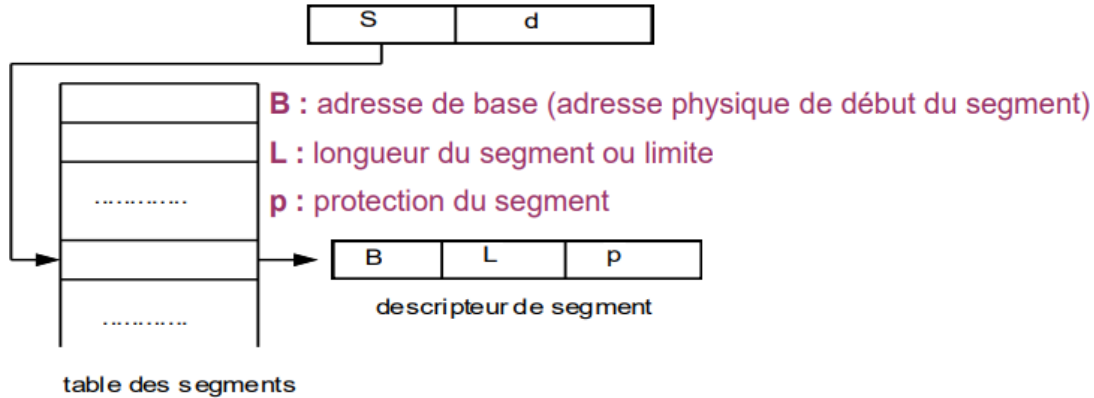
- La segmentation est un découpage de l'espace d'adressage, comme la pagination, mais elle conserve la vue du programmeur de son programme: programme principal, fonctions, données et une pile d'exécution
- Ainsi, lors de la compilation, le compilateur associe un segment à chaque morceau du programme compilé
- Un segment est un ensemble d'adresses virtuelles contiguës
- Chaque segment est repéré par son **numéro S** et sa **longueur variable L**
- Pour calculer l'adresse physique, on utilise une **table des segments**



Table de Segments

- Le tableau des segments contient des descripteurs de segments
 - B: l'adresse de base
 - L: la longueur du segment
 - P: Infos de présence
- Dans le PCB du processus il y aura un pointeur à l'adresse en mémoire de la table de segments

Table de Segments



- L'adresse physique correspondant à l'adresse virtuelle (S,d) sera donc $B+d$ si $d \leq L$



Segmentation paginée

- La segmentation peut être combiné avec la pagination: Chaque segment est composé d'un ensemble de pages (Pour remédier à la fragmentation)
- L'adresses virtuelles sont des triplets: $(s,p,d) \Rightarrow$ (numéro du segment, numéro de page, déplacement dans la page)



Exercice 4

On considère la table des segments suivante pour un processus P1:

1. Calculez les adresses réelles correspondant aux adresses virtuelles suivantes, sous forme de (segment, déplacement). Vous signalerez éventuellement les erreurs d'adressage : (0:128), (1:100), (2:465), (3:888), (4:100), (4:344)
2. L'adresse virtuelle (4,200) est-elle valide ?

Segment	Base	Limite
0	540	234
1	1254	128
2	54	328
3	2048	1024
4	976	200



Exercice 5

On considère un système avec une mémoire virtuelle segmentée paginée où la taille d'une page est de 4Ko et une mémoire physique de 64Ko. L'espace d'adressage d'un processus P est composé de trois segments S1, S2 et S3 de taille, respectivement 16Ko, 8Ko et 4Ko. À un moment donné, pour le processus P, les pages 1 et 2 du segment S1, la page 1 du segment S2 et la page 0 du segment S3 sont chargées en mémoire physique, respectivement dans les cases 2, 0, 9, 12. Pour une donnée située dans l'espace d'adressage du processus P à l'adresse décimale 8212, indiquez :

- 1) le segment
- 2) le numéro de page dans le segment
- 3) le déplacement dans la page
- 4) le numéro de case
- 5) le déplacement dans la case
- 6) l'adresse physique

Gestion des fichiers



Fichiers

- Unité de stockage logique mise à la disposition des utilisateurs pour l'enregistrement de leurs données
- Collection nommée d'information apparentées, enregistrée sur un stockage secondaire (disque)
- Les données qui se trouvent dans un stockage secondaire doivent être dans un fichier
 - SE établit la correspondance entre le fichier et le système binaire utilisé pour le stockage de manière transparente pour les utilisateurs
- Différents types
 - Données (binaire, numérique, caractère, ...)
 - Programmes



Attributs du fichier

- Propriétés du fichier
- Stockées dans un fichier spécial appelé répertoire (directory)
- Exemples
 - Nom: Permet aux personnes d'accéder au fichier
 - Identificateur: Nombre permettant au système d'identifier le fichier de façon unique
 - Type: ex: binaire, texte
 - Position: Indique le disque et l'adresse du fichier sur le disque
 - Taille: en bytes ou en blocs
 - Protection: qui peut écrire, lire, exécuter ...
 - Date: de création, de dernière modification ou de dernière utilisation



Operations sur les fichier

- Création
- Ecriture
- Lecture
- Positionnement dans un fichier
- Suppression (libération d'espace)
- Troncature
- Ajout d'information
- Renommage
- Copie
- Ouverture
- Fermeture



Répertoires

- Appelés également catalogues ou directoires
- Entité créée pour l'organisation des fichiers
- Est lui-même considéré comme un fichier spécial car il est stocké sur le disque et contient d'autres fichiers et/ou répertoires
- De point de vue SGF, un répertoire est un fichier qui dispose d'une structure logique
- Considéré comme un tableau qui contient une entrée par fichier



Répertoires: Structures

Plusieurs structures possibles pour les répertoires:

- **Structure plate à un niveau**
 - Organisée en plusieurs répertoires, mais chacun ne peut contenir que des fichiers
 - Existait à l'époque des premiers systèmes d'exploitation, car le nombre des fichiers était limité
- **Structure à deux niveaux**
 - Chaque utilisateur dispose de son propre répertoire dans lequel il peut conserver ses fichiers et répertoires
- **Structure arborescente**
 - Contient un nombre arbitraire de niveaux et chaque répertoire peut contenir des fichiers et sous-répertoires



Chemin (Path)

- Le nom complet d'un fichier est formé de:
 - La liste de répertoires qu'il faut traverser à partir du haut de la hiérarchie (Répertoire racine: root directory)
 - Le nom du fichier
- Répertoires sont séparés par un caractère qui dépend du SE
 - / pour Unix, \ pour windows
- Deux types de chemins
 - **Absolu**: qui commence à partir du répertoire racine
 - **Relatif**: ne commence pas par la racine, relatif à l'emplacement actuel



Partitionnement

- Permet la cohabitation de plusieurs SE sur le même disque
- Certaines parties du système sont isolées
- L'information sur le partitionnement du disque est stockée dans le premier secteur (secteur 0): le MBR (Master Boot Record)
- Deux types de partitionnement
 - Primaire: On peut avoir jusqu'à 4 partitions primaires sur un même disque
 - Etendue: permet de diviser une partition primaire en sous partitions



Formatage

- Avant qu'un système de fichiers puisse créer des fichiers sur une unité de stockage, son unité doit être formatée selon les spécificités du système de fichiers
- Formatages permet de:
 - Inspecter les secteurs
 - Effacer les données
 - Créer le répertoire racine du système de fichiers
 - Créer un superbloc pour stocker les informations nécessaires pour assurer l'intégrité du SGF, qui contient:
 - Identifiant de la partition (C:..., D:...)
 - Nombre de blocs dans le SGF
 - Liste des blocs libres
 - Emplacement du répertoire racine



Rôles du système de gestion des fichiers (SGF)

- Fournir une interface conviviale pour manipuler les fichiers
 - Simplifier la gestion des fichiers pour l'utilisateur
 - Possibilité d'effectuer plusieurs opérations sur les fichiers
- Gestion de l'organisation des fichiers sur le disque
 - Allocation de l'espace disque aux fichiers
- Gestion de l'espace libre sur le disque
- Gestion des fichiers dans un environnement multi-utilisateurs
- Fourniture d'utilitaires pour le diagnostic
- Récupération en cas d'erreurs



Allocation des blocs sur le disque

- Gestion de l'organisation de l'espace disque:
- Les fichiers et le disque de stockage sont découpés en un ensemble de blocs (appelés aussi clusters)
- Taille d'un bloc varie de 512 octets jusqu'à 32 ou 64 Ko
- Les SGF numérote les blocs physiques de 0 à N-A (avec $N = \text{taille du disque} / \text{taille d'un bloc}$)
- Chaque fichier est stocké sur l'unité de stockage en allouant les blocs logiques (du fichiers) aux blocs physiques (du disque)
- La lecture ou l'écriture d'un élément d'un fichier implique le transfert vers la mémoire du bloc entier qui contient cet élément
- Plusieurs techniques d'allocation existent



Allocation Contiguë

- Chaque fichier occupe un ensemble de blocs contigus sur disque
- Simple: nous n'avons besoin que de l'adresse de début et nombre de blocs pour chaque fichier
- Supporte tant l'accès séquentielle, que l'accès direct
- Application des problèmes et méthodes vus dans le chapitre de l'allocation de mémoire contiguë
- Problèmes:
 - Fragmentation externe
 - Les fichiers ne peuvent pas grandir
 - Impossible d'ajouter au milieu
 - Exécution périodique d'une défragmentation (équivalent de la compression pour la mémoire) pour récupérer l'espace libre



Allocation Non contiguë chaînée

- Allouer des blocs chaînés entre eux aux fichiers
- Un fichier peut être éparpillé sur le disque, car chaque bloc permet de retrouver le bloc suivant
- Aucune limitation de taille (en théorie)
- Elimination du problème de fragmentation externe
- Inconvénients
 - Accès du fichier est totalement séquentiel
 - La perte du chaînage entraîne la perte de tout le reste du fichier
 - La modification même minime de la valeur du pointeur peut nous amener à un autre espace en mémoire
 - Les pointeurs se trouvent sur les blocs, donc le parcours du fichier implique une lecture de tous les blocs sur le disque



Allocation Non contiguë chaînée: FAT

- Une variante de l'allocation chaînée est la table d'allocation de fichiers ou FAT (File Allocation Table)
- Les pointeurs ne sont plus stockés au niveau de chaque bloc physique, mais regroupés ensemble dans une table, qui est chargée en mémoire au démarrage
- FAT:
 - Possède une entrée pour chaque bloc disque
 - Est indexée par un numéro de bloc
- La FAT est utilisée comme s'il s'agit d'une liste chaînée
 - L'entrée du répertoire contient le numéro bloc du fichier
 - L'entrée de la table indexée par ce numéro de bloc contient le numéro du bloc suivant dans le fichier
 - Une valeur spéciale indique la fin du fichier



Allocation Non contiguë chaînée: FAT

- On parle généralement de FAT16 et FAT32
- FAT16
 - Utilisée par MS-DOS
 - Les numéros de blocs sont écrits sur 16 bits Pour des blocs de 32Ko, la taille maximale d'adressage est 2Go ($2^{16} \times 32 \text{ Ko}$)
- FAT32
 - Utilisés par windows 95 et les versions suivantes
 - Les numéros de blocs sont écrits sur 32 bits (en réalité sur 28 bits, 4 étant réservés)
 - Pour les blocs de 32 Ko, la taille maximale adressable est théoriquement 8To ($2^{28} \times 32 \text{ Ko}$)



Allocation Non contiguë indexée

- Suit le même principe que la pagination pour la mémoire
- Chaque fichier possède son propre bloc d'index
 - Tableau d'adresses de bloc disque
 - La i ème entrée dans le bloc index pointe sur le i ème bloc du fichier
- Le répertoire contient l'adresse du bloc index du fichier
- À la création du fichier, tous les pointeurs du bloc index sont à null
- Quand le i ème bloc est écrit pour la première fois, un numéro de bloc est obtenu à partir du gestionnaire d'espace libre, et son adresse est placée dans la i ème entrée du bloc index
- Inconvénient
 - La taille du fichier est limitée par la taille du bloc index



Allocation Non contiguë indexée: iNode

- Utilisé par les systèmes de gestion de fichiers ext3fs (third extended file system) d'UNIX ou GNU/Linux
- iNode est un bloc d'index multi-niveaux
 - L'iNode principal permet de pointer vers d'autres blocs d'index
- Un iNode est constitué de:
 - Attributs décrivant le fichier ou le répertoire (métadonnées)
 - Adresses des blocs contenant les données respectant le format suivant:
 - Dix pointeurs directs vers les premiers blocs de données
 - Un bloc d'indirection simple: pointe sur un bloc d'index contenant des adresses qui dirigent vers les blocs de données
 - Un bloc d'indirection double: pointe sur un bloc d'index, dont chaque entrée pointe vers un autre bloc d'index qui pointe vers des blocs de données
 - Un bloc d'indirection triple: pointe sur un bloc d'index, dont chaque entrée pointe vers un bloc d'index, dont chaque entrée pointe vers un autre bloc d'index qui pointe vers des blocs de données



Gestion de l'espace libre

Les systèmes d'exploitation utilisent essentiellement deux approches pour mémoriser l'espace libre

- Une approche statique: Bitmap
- Une approche dynamique: Liste chaînée



Approche statique: Bitmap

- Solution Mac et Windows 2000
- Pour les systèmes de fichier NTFS et ext2fs
- Utilise une table de bits (vecteurs de bits de N blocs) comportant autant de bits que de blocs sur le disque
- A chaque bloc du disque correspond un bit dans la table
- Sa valeur est 1 si le bloc est occupé
- 0 sinon
- Utile pour trouver n blocs libres contigus



Approche dynamique: Liste chaînée

- Solution MS-DOS
- Utilisation d'une liste chaînée de blocs spéciaux
- Chaque bloc spécial contient les numéros des blocs libres, et un pointeur vers le bloc spécial suivant
- Avantage: La liste ne représente que les blocs libres
- Inconvénient: Parcours de la liste



Liste chaînée avec Comptage

- Variante de la liste chaînée classique
- Permet de représenter les blocs libres contiguës en une seule ligne
- Un élément de la liste stocke l'adresse du premier bloc libre, ainsi que le nombre de blocs libres suivants
- Une entée est donc une adresse + un compteur



Exercice 1

- Avec le système de gestion de fichier UNIX, quelle taille maximale peut avoir un fichier, sachant que les blocs sont de 1Ko et que les adresses des blocs sont sur 4 octets ?
- Refaites le même calcul pour un système de fichier qui gère des blocks 4Ko et des adresses de 4 octets.



Exercice 2

- On considère un fichier texte 2500000 caractères ASCII (y compris les caractères de fin de ligne et de fin de fichier). Suite à un malencontreux accident, l'inode de ce fichier est corrompu et le contenu des pointeurs indirects double et triple est détruit. Calculez le pourcentage de la proportion perdue du fichier suite de cet accident ?