Institut Universitaire et Stratégique de l'Estuaire
Estuary Academic and Strategic Institute(IUEs/Insam)
Sous la tutelle académique des Universités de Buea, de Douala et de Dschang

Intitulé du Cours : Système d'exploitation

Spécialités : GL 2

Enseignant : **KAMWA KOUOGANG Michael Christ**

Master 2 en Génie Industriel et Technologie

Option : Automatique et Informatique Industrielle

Mail : michaelchristk@gmail.com

Tél : 655.50.94.93/ 679.75.75.66

Programme de cours :

1. **Généralités sur les systèmes d'exploitation**
2. **Gestion des processus**
3. **L'ordonnancement des processus**
4. **La Gestion de la mémoire**
5. **La Gestion des périphériques**
6. **La Gestion de fichiers**

Présentation globale :

Ce cours est une introduction aux systèmes d'exploitation. Il permet d'initier et de familiariser les étudiants à la mise en œuvre des équipements informatique, à la maîtrise du fonctionnement de l'ordinateur et de l'utilité de chacun de ces constituants. De plus un ensemble de travaux dirigés, et des travaux pratiques permettent une meilleure compréhension.

Objectifs généraux :

L'objectif du cours de système d'exploitation est de permettre à l'étudiant de comprendre le rôle et le principe de fonctionnement des systèmes d'exploitation et maîtriser les fonctionnalités de gestion des mécanismes de base (fichier, processus, mémoire et périphériques) d'un ordinateur.

Objectifs spécifiques :

Ce cours présente de façon claire et didactique toutes les données qui président au choix et à la décision d'achat et d'installation d'un système d'exploitation. Ensuite après une présentation générale des SE et de ces différents composants, il propose pour les différents constituants un mode de fonctionnement et un moyen de gestion des ressources associées ainsi que leurs caractéristiques.

Prérequis :

Architecture des ordinateurs, Algorithmique et Programmation procédurale

Chapitre 1

Généralités sur les systèmes d'exploitation

Objectif général : À la fin de ce chapitre, l'étudiant doit être capable de choisir un SE pour un ordinateur.

Objectifs spécifiques: ce chapitre permet :

- De connaitre la définition d'un système d'exploitation
- De connaitre le rôle d'un système d'exploitation
- De connaitre les classes des systèmes d'exploitation
- De connaitre les mécanismes d'un système d'exploitation

1.1 Introduction

Les ordinateurs permettent de collecter des données, de réaliser des calculs, de stocker des informations et de communiquer avec d'autres ordinateurs. Un ordinateur est formé d'une partie matérielle et d'une partie logicielle. Cette dernière comporte des logiciels qui sont classés en deux catégories : les programmes d'application des utilisateurs et les programmes système qui permettent le fonctionnement de l'ordinateur. Parmi ceux-ci, le système d'exploitation (SE).

Le système d'exploitation est le logiciel qui prend en charge les fonctionnalités élémentaires du matériel et qui propose une plateforme plus efficace en vue de l'exécution des programmes. Il gère les ressources matérielles, offre des services pour accéder à ces ressources et crée des éléments abstraits de niveau supérieur, tels que des fichiers, des répertoires et des processus.

1.2 Historique

Tout système d'exploitation dépend étroitement de l'architecture de l'ordinateur sur lequel il fonctionne.

✈ **La 1ère génération (1945 -1955) : les tubes à vide et les cartes enfichables**

Il n'existait pas de système d'exploitation. Les utilisateurs travaillaient chacun leur tour sur l'ordinateur qui remplissait une salle entière. Ils étaient d'une très grande lenteur. Ils étaient d'une très grande fragilité.

✈ **La 2ème génération (1955 -1965) : les transistors et le traitement par lots**

Le passage aux transistors rendait les ordinateurs plus fiables. Ils pouvaient être vendus à des utilisateurs (grandes compagnies, université ou administrations). Mais devant les coûts d'équipement élevés on réduisit les temps grâce au traitement par lots.

✈ **La 3ème génération (1965 -1980) : les circuits intégrés et la multiprogrammation.**

Amélioration des coûts et des performances (circuits intégrés). Une famille d'ordinateurs compatibles entre eux. Une seule architecture et un même jeu d'instructions. Des ordinateurs uniques pour les calculs scientifiques et commerciaux. Apparition du spoule (spool, Simultaneous Peripheral Operation On Line) pour le transfert des travaux des cartes vers le disque. Apparition de la multiprogrammation (partitionnement de la mémoire pour des tâches différentes).

Mais, un système d'exploitation énorme et très complexe pour satisfaire tous les besoins (plusieurs millions de lignes d'assembleur). Apparition du partage de temps, une variante de la multiprogrammation (chaque utilisateur possède un terminal en ligne) ; naissance du système MULTICS (MULTiplexed Information and Computing Service) pour ordinateur central. Apparition des mini-ordinateurs (DEC PDP-1 en 1961, 4K mots de 18 bits, pour un prix de 120 000 $).

K. Thompson écrivit une version simplifiée (mono-utilisateur) de MULTICS ; B. Kernighan l'appela avec humour UNICS (Uniplexed Information and Computer Service) ; ce nom allait devenir UNIX1.

D. Ritchie se joignit à K. Thompson pour réécrire UNIX en langage C ; ce système d'exploitation a été le plus porté sur toutes sortes de machine.

✈ **La 4ème génération (1980 -1990) : les ordinateurs personnels.**

Ils sont dus au développement des circuits LSI (Large Scale Integration) contenant des centaines de transistors au cm². Ils ont la même architecture que les mini-ordinateurs mais leur prix est beaucoup moins élevé. Il existe deux systèmes d'exploitation principaux : MS-DOS (Microsoft Inc.) et UNIX. MS-DOS intègre petit à petit des concepts riches d'UNIX et de MULTICS. Dans le milieu des années 80, on voit l'apparition de réseaux d'ordinateurs individuels qui fonctionnent sous des systèmes d'exploitation en réseau ou des systèmes d'exploitation distribués.

✈ **La 5ème génération (1990 -????) : les ordinateurs personnels portables et de poche.**

Apparition des PIC (Personal Intelligent Communicator de chez Sony) et des PDA (Personal Digital Assistant, comme le Newton de chez Apple), grâce à l'intégration des composants et l'arrivée des systèmes d'exploitation de type « micro-noyau ». Ils sont utiles pour les « nomades » et les systèmes de gestion des informations (recherche, navigation, communication). Ils utilisent la reconnaissance de caractère (OCR) et les modes de communication synchrone et asynchrone (mode messagerie). Très bon marché, ils sont capables de se connecter à des ordinateurs distants et performants. Les systèmes d'exploitation de type « micro-noyau » sont modulaires (un module par fonction) ; ils peuvent être réalisés avec plus ou moins de modules et donc adaptables à des très petites machines (PDA et PIC).

1.3 Les systèmes d'exploitation

1.3.1 Rôle et définition

Un système d'exploitation (SE) est présent au cœur de l'ordinateur coordonnant les tâches essentielles à la bonne marche du matériel. C'est du système d'exploitation que dépend la qualité de la gestion des ressources (processeur, mémoire, périphériques) et la convivialité de l'utilisation d'un ordinateur.

Un SE résout les problèmes relatifs à l'exploitation de l'ordinateur en garantissant :

- Une gestion **efficace, fiable et économique** des ressources physiques de l'ordinateur (notamment les ressources critiques telles que processeur, mémoire…) : il ordonne et contrôle l'allocation des processeurs, des mémoires, des icônes et fenêtres, des périphériques, des réseaux entre les programmes qui les utilisent. Il assiste les programmes utilisateurs. Il protège les utilisateurs dans le cas d'usage partagé.
- Il propose à l'utilisateur une abstraction plus simple et plus agréable que le matériel : une **machine virtuelle** permettant l'interaction avec les utilisateurs en leur présentant une machine plus simple à exploiter que la machine réelle

1.3.2 Classes de systèmes d'exploitation

✈ **Mono-tâche (DOS)***:* A tout instant, un seul programme est exécuté; un autre programme ne démarrera, sauf conditions exceptionnelles, que lorsque le premier sera terminé.

✈ **Multi-tâches (Windows, Unix, Linux, VMS)** *:* plusieurs processus (i. e. un «programme» en cours d'exécution) peuvent s'exécuter simultanément (systèmes multiprocesseurs) ou en quasi- parallélisme (systèmes à temps partagé)

✈ **Mono-session (Windows 98,2000)***:* au plus un utilisateur à la fois sur une machine. Les systèmes réseaux permettent de différencier plusieurs utilisateurs, mais chacun d'eux utilise de manière exclusive la machine (multi- utilisateurs, mono- session)

✈ **multi-sessions (Windows XP, Unix, Linux, VMS)***:* Plusieurs utilisateurs peuvent travailler simultanément sur la même machine.

1.3.3 Structure en couches d'un SE moderne

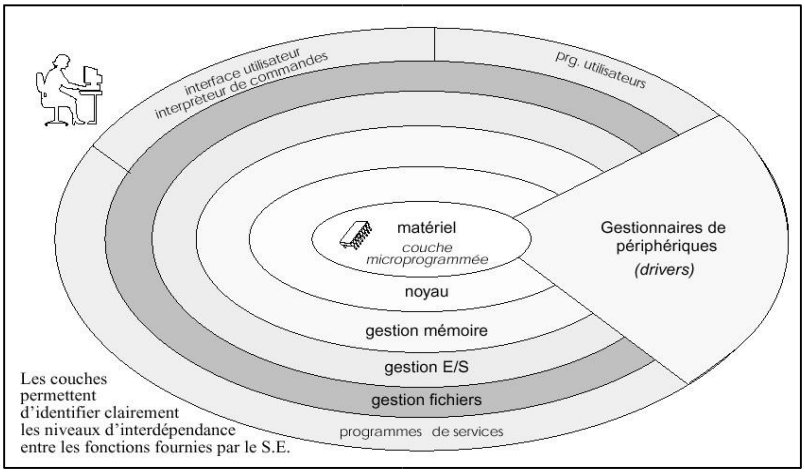


Figure 1: Structure en couche d'un SE

a. Le noyau

- Réside en mémoire (fréquence élevée des interventions)
- Petite taille
- Gestion du processeur : reposant sur un allocateur (dispatcher*)* responsable de la répartition du temps processeur entre les différents processus, et un planificateur (scheduler*)* déterminant les processus à activer, en fonction du contexte.
- Gestion des *interruptions* : les *interruptions* sont des signaux envoyés par le matériel, à destination du logiciel, pour signaler un évènement.
- Gestion du multi- tâches : simuler la simultanéité des processus coopératifs (i. e. les processus devant se synchroniser pour échanger des données) et gérer les accès concurrents aux ressources
  (Fichiers, imprimantes, ...)

b. Le système de gestion de fichiers

Le concept de fichiers est une structure adaptée aux mémoires secondaires et auxiliaires permettant de regrouper des données.

Le rôle d'un système d'exploitation est de donner corps au concept de fichiers (les gérer, c'est- à- dire les créer, les détruire, les écrire (modifier) et les lires, en offrant la possibilité de les désigner par des noms symboliques).

c. Les Entrées/ Sorties

Il s'agit de permettre le dialogue (échange d'informations) avec l'extérieur du système.

La tâche est rendue ardue, par la diversité des périphériques d'entrées- sorties et les multiples méthodes de codage des informations (différentes représentations des nombres, des lettres, etc.)

Concrètement, la gestion des E/S implique que le SE mette à disposition de l'utilisateur des procédures standard pour l'émission et la réception des données, et qu'il offre des traitements appropriés aux multiples conditions d'erreurs susceptibles de se produire (plus de papier, erreur de disque, débit trop différent, ...)

d. L'invite des commandes ou shell

Nécessaire pour interagir avec l'utilisateur, il peut être

- Graphique
- Console interpréteur de commandes (langage de commande interprété).

Il attend les ordres que l'utilisateur transmet par le biais de l'interface, décode et décompose ces ordres en actions élémentaires, et finalement réalise ces actions en utilisant les services des couches plus profondes du système d'exploitation.

Outre l'interaction « directe » (au moyen de terminaux ou de consoles dans le cas d'Unix *ou MS DOS*), les systèmes offrent le *« traitement par lot*s » *(batc*h). Ce mode de traitement non-interactif est obtenu en regroupant les commandes dans un fichier alors appelé *scrip*t.

e. La mémoire virtuelle

La mémoire centrale a toujours été une ressource critique : initialement très coûteuse et peu performante (tores magnétiques), elle était de très faible capacité.

Pour pallier le manque de mémoire centrale, l'idée est venue d'utiliser des mémoires secondaires (de type disque dur), plus lentes, mais de beaucoup plus grandes capacités.

La mémoire virtuelle repose sur une décorellation entre la mémoire physique (centrale ou secondaire), présente sur la machine, et l'espace mémoire mis à disposition des programmes par le système d'exploitation (la mémoire virtuelle, ou logique).

1.4 Eléments de base d'un système d'exploitation :

Les principales fonctions assurées par un SE sont les suivantes :

- Gestion de la mémoire principale et des mémoires secondaires,
- Exécution des E/S (périphériques) à faible débit ou haut débit
- Multiprogrammation, temps partagé, parallélisme
- Interruption, ordonnancement, répartition en mémoire, partage des données,
- Lancement des outils du système (compilateurs, environnement utilisateur,)
- Lancement des outils pour l'administrateur du système
- Protection, sécurité ;
- Réseaux

Chapitre 2

Gestion des processus

Objectif général : À la fin de ce chapitre, l'étudiant doit être capable de maitriser la notion de processus et d'interruption.

Objectifs spécifiques : ce chapitre permet :

- De connaitre la notion de processus
- De connaitre les caractéristiques d'un processus ainsi que son contexte
- De Connaître la notion d'interruptions et de ressources
- De Connaître les étapes du cycle de vie d'un processus.

2.1 Définition d'un processus

Un processus est une entité dynamique correspondant à l'exécution d'une suite d'instructions : un programme qui s'exécute, ainsi que ses données, sa pile, son compteur ordinal, son pointeur de pile et les autres contenus de registres nécessaires à son exécution.

Attention : ne pas confondre un processus (aspect dynamique, exécution qui peut être suspendue, puis reprise), avec le texte d'un programme exécutable (aspect statique).

2.2 Caractéristiques

- Un processus possède un identifiant unique qui est généralement un entier incrémental (le premier processus 1, le second 2, … etc.) et qui désigne de façon unique le processus dans le système (PID : Process Identifier)
- Les instructions à exécuter sont stockées dans une pile de données contenant les valeurs des variables du programme.
- Un contexte d'exécution : contenant entre autre le compteur ordinal qui indique l'adresse de l'instruction qui va être exécuté
- Les adresses des ressources utilisées
- Identifiant du processus parent

Les appels système relatifs aux processus permettent généralement d'effectuer au moins les actions suivantes :

- création d'un processus (fils) par un processus actif (d'où la structure d'arbre de processus gérés par un SE)
- destruction d'un processus
- mise en attente, réveil d'un processus
- suspension et reprise d'un processus, grâce à l'Ordonnanceur de processus (scheduler)
- demande de mémoire supplémentaire ou restitution de mémoire inutilisée
- attendre la fin d'un processus fils
- remplacer son propre code par celui d'un programme différent
- échanges de messages avec d'autres processus
- spécification d'actions à entreprendre en fonction d'événements extérieurs asynchrones
- modifier la priorité d'un processus

Dans une entité logique unique, généralement un mot, le SE regroupe des informations-clés sur le fonctionnement du processeur : c'est le mot d'état du processeur (Processor Status Word, PSW). Il comporte généralement :

- la valeur du compteur ordinal
- des informations sur les interruptions (masquées ou non)
- le privilège du processeur (mode maître ou esclave)
- etc.... (format spécifique à un processeur)

A chaque instant, un processus est caractérisé par son état courant ou contexte : c'est l'ensemble des informations nécessaires à la poursuite de son exécution (valeur du compteur ordinal, contenu des différents registres, informations sur l'utilisation des ressources). A cet effet, à tout processus, on associe un bloc de contrôle de processus (BCP). Il comprend généralement : - une copie du PSW au moment de la dernière interruption du processus - l'état du processus : prêt à être exécuté, en attente, suspendu, ...

- des informations sur les ressources utilisées
- mémoire principale
- temps d'exécution
- périphériques d'E/S en attente

- files d'attente dans lesquelles le processus est inclus, etc...
- et toutes les informations nécessaires pour assurer la reprise du processus en cas d'interruption

Les BCP sont rangés dans une table en mémoire centrale à cause de leur manipulation fréquente.

2.3 Les interruptions

Une interruption est une commutation de l'état (contexte) d'un processus provoquée par un signal généré par le matériel. Ce signal est la conséquence d'un événement interne à un processus, résultant de son exécution, ou bien extérieur et indépendant de son exécution. Le signal va modifier la valeur d'un indicateur qui est consulté par le SE. Celui-ci est ainsi informé de l'arrivée de l'interruption et de son origine. A chaque cause d'interruption est associé un niveau d'interruption.

On distingue au moins 3 niveaux d'interruption :

- les interruptions externes : panne, intervention de l'opérateur, ...
- les déroutements qui proviennent d'une situation exceptionnelle ou d'une erreur liée à l'instruction en cours d'exécution (division par 0, débordement de mémoire, ...)
- les appels système

Le chargement d'un nouveau mot d'état provoque l'exécution d'un autre processus, appelé le traitant de l'interruption. Le traitant réalise la sauvegarde du contexte du processus interrompu (compteur ordinal, registres, indicateurs,....). Puis le traitant accomplit les opérations liées à l'interruption concernée et restaure le contexte et donne un nouveau contenu au mot d'état : c'est l'acquittement de l'interruption. Généralement un numéro de priorité est affecté à un niveau d'interruption pour déterminer l'ordre de traitement lorsque plusieurs interruptions sont positionnées.

2.4 Les ressources

On appelle ressource tout ce qui est nécessaire à l'avancement d'un processus (continuation ou progression de l'exécution) : processeur, mémoire, périphérique, bus, réseau, compilateur, fichier, message d'un autre processus, etc... Un défaut de ressource peut provoquer la mise en attente d'un processus.

Un processus demande au SE l'accès à une ressource. Certaines demandes sont implicites ou permanentes (la ressource processeur). Le SE alloue une ressource à un processus. Une fois une ressource allouée, le processus a le droit de l'utiliser jusqu'à ce qu'il libère la ressource ou jusqu'à ce que le SE reprenne la ressource (on parle en ce cas de ressource préemptible, de préemption). On dit qu'une ressource est en mode d'accès exclusif si elle ne peut être allouée à plus d'un processus à la fois. Sinon, on parle de mode d'accès partagé. Un processus possédant une ressource peut dans certains cas en modifier le mode d'accès.

Exemple : un disque est une ressource à accès exclusif (un seul accès simultané), une zone mémoire peut être à accès partagé.

Le mode d'accès à une ressource dépend largement de ses caractéristiques technologiques. Deux ressources sont dites équivalentes si elles assurent les mêmes fonctions vis à vis du processus demandeur. Les ressources équivalentes sont groupées en classes afin d'en faciliter la gestion par l'Ordonnanceur.

2.5 Cycle de vie d'un processus

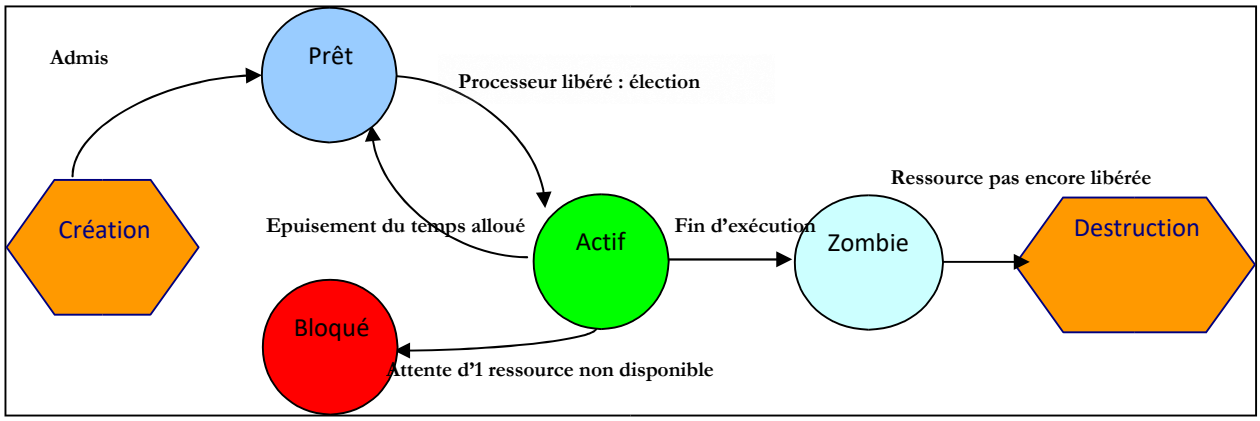


Figure 1: Cycle de vie d'un processus

- **Création:** chargement des instructions, allocation de mémoires et des ressources (statiquement), il passe directement à l'état prêt.

Il existe 4 événements pour créer un processus :

- L'initialisation du système : au chargement du système il y'a création automatique du processus racine père de tous les processus utilisateurs (id=0)
- Un processus peut lancer un autre processus, il en devient le parent, l'autre dernier sera désigné comme processus fils. (Un processus père ne se termine que lorsque tous ses fils sont terminés. On a donc une structure arborescente de processus).
- Une requête de l'utilisateur

- Initiation d’un travail en traitement par lot

- **La destruction d’un processus:** Lors de la destruction le processus libère les ressources allouées. Il y a quatre causes possibles de la destruction d’un processus :
  - Arrêt normal : volontaire, lorsque le processus termine sa tâche.
  - Arrêt pour erreur : volontaire suite à une erreur pour une instruction illégale
  - Arrêt pour erreur fatale : involontaire tel que les mauvais paramètres de l’exécution du processus
  - Arrêt volontaire par un autre processus

– **L’état prêt:** Le processus est prêt à être exécuté. Il est mis en attente jusqu’à ce qu’on lui libère le processeur (dispatch de l’Ordonnanceur), il passera alors à l’état Actif

– **L’état actif ou élu:**
  - Le processus est en cours d’exécution par le processeur.
  - Si le processus épuise le temps qui lui est alloué par le SE, il est remis en file d’attente des Prêts.
  - Si il a besoin d’une ressource non disponible (opérations sur les périphériques), il est mis en attente prolongée (Interruption : état bloqué) jusqu’à la libération de la ressource nécessaire.
  - Si le processus atteint son terme (se termine) il passe à l’état Zombie

– **L’état bloqué:** Le processus est en attente d’une ressource pour terminer. Dès sa libération il repasse à l’état Prêt

– **L’état zombie:** Le processus a terminé son exécution et il ne peut plus évoluer mais les ressources qu’il a allouées ne sont pas encore libérées

3 L'ordonnancement

On appelle ordonnancement la stratégie d'attribution des ressources aux processus qui en font la demande. Différents critères peuvent être pris en compte :

- temps moyen d'exécution minimal
- temps de réponse borné pour les systèmes interactifs
- taux d'utilisation élevé de l'UC
- respect de la date d'exécution au plus tard, pour le temps réel, etc..

Chapitre 3

Ordonnancement des processus

Objectif général : À la fin de ce chapitre, l'étudiant doit être capable de connaitre la notion d'ordonnancement des processus.

Objectifs spécifiques: ce chapitre permet :

- De comprendre la problématique du multitâche
- De connaître les différents critères d'ordonnancement
- De connaître les algorithmes d'ordonnancement préemptifs
- De connaître les algorithmes d'ordonnancement non préemptifs

3.1 Introduction

Un ordinateur possède forcément plusieurs processus en concurrence pour l'obtention du temps processeur, cette situation se produit lorsque 2 ou plusieurs processus sont en état prêt simultanément. L'Ordonnanceur (planificateur, scheduler) est la partie (un programme) du système d'exploitation responsable de régler les états des processus (Prêt, Actif,…etc.) et de gérer les transitions entre ces états ; c'est l'allocateur du processeur aux différent processus, il alloue le processeur au processus en tête de file des Prêts.

3.2 Objectifs d'un Ordonnanceur

Les objectifs d'un Ordonnanceur sont :

- Maximiser l'utilisation du processeur
- Présenter un temps de réponse acceptable
- Respecter l'équité entre les processus selon le critère d'ordonnancement utilisé.

3.3 Critères d'ordonnancement

L'objectif d'un algorithme d'ordonnancement consiste à identifier le processus qui conduira à la meilleure performance possible du système. Certes, il s'agit là d'une évaluation subjective dans laquelle entrent en compte différents critères à l'importance relative variable. La politique d'ordonnancement détermine l'importance de chaque critère. Un certain nombre d'algorithmes ont fait leur preuve dans la mise en œuvre d'une politique d'ordonnancement.

La liste qui suit passe en revue des critères d'ordonnancement fréquemment utilisés.

- **Utilisation de l'UC :** Pourcentage de temps pendant lequel l'UC exécute un processus. L'importance de ce critère varie généralement en fonction du degré de partage du système. **Utilisation répartie :** Pourcentage du temps pendant lequel est utilisé l'ensemble des ressources (outre l'UC, mémoire, périphérique d'E/S…)
- **Débit :** Nombre de processus pouvant être exécutés par le système sur une période de temps donnée.
- **Temps de rotation :** durée moyenne qu'il faut pour qu'un processus s'exécute. Le temps de rotation d'un processus comprend tout le temps que celui-ci passe dans le système. Il est inversement proportionnel au débit.
- **Temps d'attente :** durée moyenne qu'un processus passe à attendre. Mesurer la performance par le temps de rotation présente un inconvénient : Le temps de production du processus accroît le temps de rotation ; Le temps d'attente représente donc une mesure plus précise de la performance.
- **Temps de réponse :** Temps moyen qu'il faut au système pour commencer à répondre aux entrées de l'utilisateur.
- **Equité :** degré auquel tous les processus reçoivent une chance égale de s'exécuter.
- **Priorités :** attribue un traitement préférentiel aux processus dont le niveau de priorité est supérieur.

3.4 Types d'ordonnancement

Il existe 2 types d'ordonnancement

a. **Ordonnancement préemptif:** Avec réquisition où l'Ordonnanceur peut interrompre un processus en cours d'exécution si un nouveau processus de priorité plus élevée est inséré dans la file des Prêts.
b. **Ordonnancement coopératif:** Ordonnancement jusqu'à achèvement : le processus élu garde le contrôle jusqu'à épuisement du temps qui lui a été alloué même si des processus plus prioritaires ont atteint la liste des Prêts

3.5 Les algorithmes d'ordonnancement

a. L'algorithme FIFO (First In First Out)

L'ordonnancement est fait dans l'ordre d'arrivée en gérant une file unique des processus sans priorité ni réquisition : chaque processus s'exécute jusqu'à son terme ; le processus élu est celui qui est en tête de liste des Prêts : le premier arrivé. Cet algorithme est facile à implanter, mais il est loin d'optimiser le temps de traitement moyen

b. L’algorithme SJF (Shortest Job First)

L'ordonnancement par ordre inverse du temps d'exécution (supposé connu à l’avance) : lorsque plusieurs travaux d'égale importance se trouvent dans une file, l'Ordonnanceur élit le plus court d'abord (les travaux les plus cours étant en tête de la file des prêts).

Cet algorithme possède l’inconvénient de la nécessité de connaissance du temps de service à priori et le risque de privation des tâches les plus longues. Afin de résoudre ces problèmes on pourra attribuer aux travaux une priorité croissante avec leur temps de séjour dans la file (temps d’attente). A temps d’attente égale, le travail le plus court est prioritaire. Toutefois, cette solution nécessite le calcul des priorités périodiquement et un réarrangement de la FA.

c. L’algorithme du temps restant le plus court (SRT : Shortest Remaining Time)

L’algorithme du temps restant le plus court, est la version préemptive de l’algorithme SJF. Chaque fois qu’un nouveau processus est introduit dans la file des processus à ordonnancer, l’Ordonnanceur compare la valeur estimée du temps de traitement restant à celle du processus en cours d’ordonnancement. Si le temps de traitement du nouveau processus est inférieur, le processus en cours d’ordonnancement est préempte. Tout comme l’algorithme SJF, l’algorithme SRT favorise les travaux courts : les travaux longs en revanche peuvent être victimes de famine.

d. L’algorithme Round Robin

Il s'agit d'un algorithme ancien, simple et fiable. Le processeur gère une liste circulaire de processus. Chaque processus dispose d'un quantum de temps pendant lequel il est autorisé à s'exécuter. Si le processus actif se bloque ou s'achève avant la fin de son quantum, le processeur est immédiatement alloué à un autre processus. Si le quantum s'achève avant la fin du processus, le processeur est alloué au processus suivant dans la liste et le processus précédent se trouve ainsi en queue de liste.

La commutation de processus (overhead) dure un temps non nul pour la mise à jour des tables, la sauvegarde des registres. Un quantum trop petit provoque trop de commutations de processus et abaisse l'efficacité du processeur. Un quantum trop grand augmente le temps de réponse en mode interactif. On utilise souvent un quantum de l'ordre de 100 ms.

e. L’algorithme HPF (Highest Priority First)

Le modèle du tourniquet suppose tous les processus d'égale importance. C'est irréaliste. D'où l'attribution de priorité à chaque processus. L'Ordonnanceur lance le

processus prêt de priorité la plus élevée. En cas de priorités égales on utilise l'algorithme FIFO. L'ordonnancement des priorités peut être préemptif ou non.

Les mécanismes d'attribution des priorités sont très variables ; la priorité est basée sur la caractéristique du processus (utilisation de la mémoire, fréquence des E/S), sur l'utilisateur qui exécute le processus, les coûts d'utilisation (Le temps de l'UC pour les tâches de priorité supérieure est par exemple plus coûteux) ou sur un paramètre que l'utilisateur peut spécifier. Certains mécanismes produisent des priorités qui varient de manière dynamique : volume du temps d'exécution ; alors que d'autre sont statiques (la priorité associée à un utilisateur).

f. Les files d'attente Rétroactives

Cet algorithme met en œuvre N files (N>1) d'attentes classées de 1 à N, un processus qui vient d'être crée est placée dans la file du niveau le plus élevé, une fois sélectionnés les processus de la file reçoivent une tranche de temps relativement courte. A l'expiration de cette dernière, le processus est déplacé dans la file de niveau inférieur. Les tranches de temps associées aux files s'allongent au fur et à mesure que le niveau décroît. Si un processus est bloqué avant d'avoir utilisé la totalité de sa tranche de temps, il passe à la file de niveau supérieur. Le processus sélectionné par cet algorithme est le prochain processus de la file la plus élevée contenant des processus. La sélection au sein d'une file se fait suivant la stratégie FIFO. Si un processus entre dans une file alors qu'un autre processus d'un niveau inférieur est en cours d'exécution, ce dernier peut être préempté.

3.6 Performance des algorithmes d'Ordonnancement

Les performances d'un algorithme pour un ensemble de processus donné peuvent être analysées si les informations appropriées relatives aux processus sont fournies. Par exemple, des données sur l'arrivée du processus et sur l'heure d'exécution de ce processus sont nécessaires pour évaluer l'algorithme SRT.

Temps de rotation=Temps fin d'exécution - Temps d'arrivée

Temps d'attente=Temps de rotation – Durée d'exécution

$$temps\ \ moyen\ \ d'attente = \frac{\sum temps\ \ attente}{nbre\ \ de\ \ processus}$$

$$\mathrm{Re}ndement = \frac{\sum temps\ \ d'execution}{nbre\ \ de\ \ processus}$$

Chapitre 4

Gestion de la mémoire

Objectif général : À la fin de ce chapitre, l'étudiant doit être capable de connaitre la notion de mémoire et son utilité.

Objectifs spécifiques: ce chapitre permet de :

- Connaître le principe de gestion de mémoire en monoprogrammation
- Connaître le principe de gestion de mémoire en multiprogrammation
- Connaître la notion de mémoire virtuelle et son utilité,
- Connaître le concept de pagination et son principe
- Connaître le concept de segmentation

4.1 Introduction

La mémoire physique sur un système se divise en deux catégories :

- la mémoire vive : composée de circuit intégrés, donc très rapide.
- la mémoire de masse (secondaire) : composée de supports magnétiques (disque dur, bandes magnétiques...), qui est beaucoup plus lente que la mémoire vive.

La mémoire est une ressource rare. Il n'en existe qu'un seul exemplaire et tous les processus actifs doivent la partager. Si les mécanismes de gestion de ce partage sont peu efficaces l'ordinateur sera peu performant, quelque soit la puissance de son processeur. Celui-ci sera souvent interrompu en attente d'un échange qu'il aura réclamé. Si un programme viole les règles de fonctionnement de la mémoire, il en sera de même. Dans ce chapitre on va mettre l'accent sur la gestion de la mémoire pour le stockage des processus concurrents.

4.2 Gestion sans recouvrement ni pagination

4.2.1 La monoprogrammation : Le modèle de base

La représentation de base, pour un système monoprocesseur et mono tâche, est montrée dans figure suivante : il s'agit d'une partition contiguë.

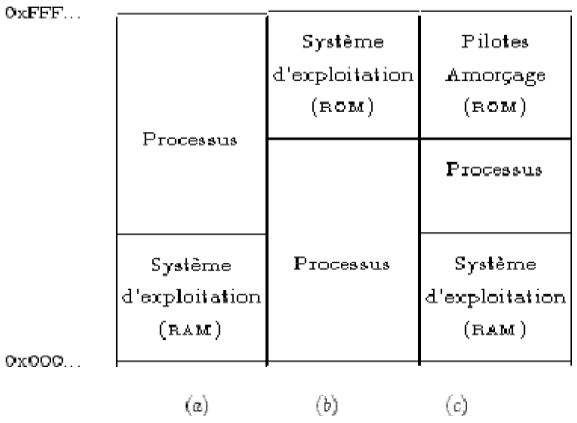


Figure 1 : Les trois organisations de base pour un système mono tâche.

En général, le système d'exploitation se trouve au niveau des premières adresses de la zone mémoire de la RAM. Pour des systèmes avec un SE embarqué (consoles de jeu, téléphones mobiles, *etc*) le système se trouve souvent dans une partie non modifiable (ROM). Afin de garantir de façon transparente mais flexible, l'amorçage d'un système quelconque, on retrouve souvent une combinaison des deux approches (RAM et ROM). La ROM contient alors un système minimal permettant de piloter les périphériques de base (clavier -- disque -- écran) et de charger le code d'amorçage à un endroit bien précis. Ce code est exécuté lors de la mise sous tension de l'ordinateur, et le « vrai » système d'exploitation, se trouvant dans la zone d'amorçage sur le disque, est ensuite chargé, prenant le relais du système minimal.

4.2.2 La Multiprogrammation

a. **Multiprogrammation avec partitions fixes**

i. Partition fixes de tailles égales

Cette partition peut être faite une fois pour toute au démarrage du système par l'opérateur de la machine, qui subdivise la mémoire en partitions fixes de tailles égales. Chaque nouveau processus est placé dans une partition vide. Ceci engendrera le problème de fragmentation interne due au faite que si la taille de partition est supérieure à l'espace recueilli par un certain processus le reste de cette partition restera vide ce qui cause une perte d'espace mémoire.

Le sous-système chargé de la gestion de mémoire met en place une structure des données appelée table des partitions ayant pour rôle l'indication de l'état (libre ou occupée) de chaque partition en identifiant chacune soit par son adresse soit par son numéro.

ii. Partition fixes de tailles inégales

Dans ce cas la mémoire est subdivisée en partitions de tailles inégales. Chaque nouveau processus est placé dans la file d'attente de la plus petite partition qui peut le contenir. Cette façon de faire peut conduire à faire attendre un processus dans une file, alors qu'une autre partition pouvant le contenir est libre.

Il existe deux méthodes de gestion :

- On crée une file d'attente par partition. Chaque nouveau processus est placé dans la file d'attente de la plus petite partition pouvant le contenir. Inconvénients :
  - on perd en général de la place au sein de chaque partition
  - il peut y avoir des partitions inutilisées (leur file d'attente est vide)
- On crée une seule file d'attente globale. Il existe deux stratégies :
  - Dès qu'une partition se libère, on lui affecte la première tâche de la file qui peut y tenir. Inconvénient : on peut ainsi affecter une partition de grande taille à une petite tâche et perdre beaucoup de place
  - Dès qu'une partition se libère, on lui affecte la plus grande tâche de la file qui peut y tenir. Inconvénient : on pénalise les processus de petite taille.

    L'alternative à cette approche consiste à n'utiliser qu'une seule file d'attente : dès qu'une partition se libère, le système y place le premier processus de la file qui peut y tenir. Cette solution réduit la fragmentation interne de la mémoire.

b. Multiprogrammation avec partitions variables

Au lancement du système, on crée une seule zone libre de taille maximale. Lorsqu'on charge un programme, on le place dans une zone libre suffisante, et on lui alloue **exactement** la mémoire nécessaire. Le reste devient une nouvelle zone libre. Lorsqu'un programme s'achève, sa partition redevient libre, et peut éventuellement grossir une zone libre voisine. Il n'y a donc ce qu'on appelle fragmentation externe.

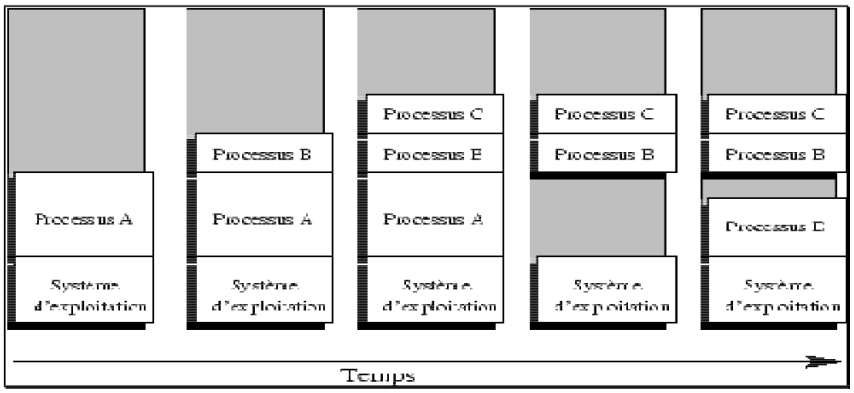


Figure 2: Principe d'allocation de partitions de taille variable.

La solution la plus flexible est d'adopter un système avec des partitions de taille variable et un système de va-et-vient qui permet d'utiliser le disque comme mémoire secondaire et d'y stocker un nombre de processus inactifs ou en attente.

4.3 Gestion avec recouvrement sans pagination

Dès que le nombre de processus devient supérieur au nombre de partitions, il faut pouvoir simuler la présence en mémoire centrale (MC) de tous les processus pour pouvoir satisfaire au principe d'équité et minimiser le temps de réponse des processus. La technique du recouvrement permet de stocker temporairement sur disque des images de processus afin de libérer de la MC pour d'autres processus. On pourrait utiliser des partitions fixes, mais on utilise en pratique des partitions de taille variable, car le nombre, la taille et la position des processus peuvent varier dynamiquement au cours du temps. On n'est plus limité par des partitions trop grandes ou trop petites comme avec les partitions fixes. Cette amélioration de l'usage de la MC nécessite un mécanisme plus complexe d'allocation et de libération.

4.3.1 Le va et vient

Conceptuellement le va-et-vient, ou *swap*, se comporte exactement comme la mémoire vive, à la différence près qu'on ne peut y exécuter des processus (pour exécuter un processus sur le *swap*, il faut le charger en mémoire vive), ainsi que quelques considérations liées au médium de stockage, qui impose un accès et un stockage par blocs, mais que l'on peut négliger.

Un processus qui est inactif (soit bloqué, soit prêt) peut donc être placé dans le swap. Les stratégies de choix des processus à écrire sur disque sont sensiblement identiques à celles mises en œuvre pour l'ordonnancement des processus. Il est à noter qu'il existe deux types de solution : soit on alloue une place fixe dans le swap, pour chaque processus créé, soit on utilise le swap comme une grande zone de stockage dans laquelle les processus sont écrits

selon les besoins et à un endroit déterminé en fonction de l'occupation au moment de l'écriture.

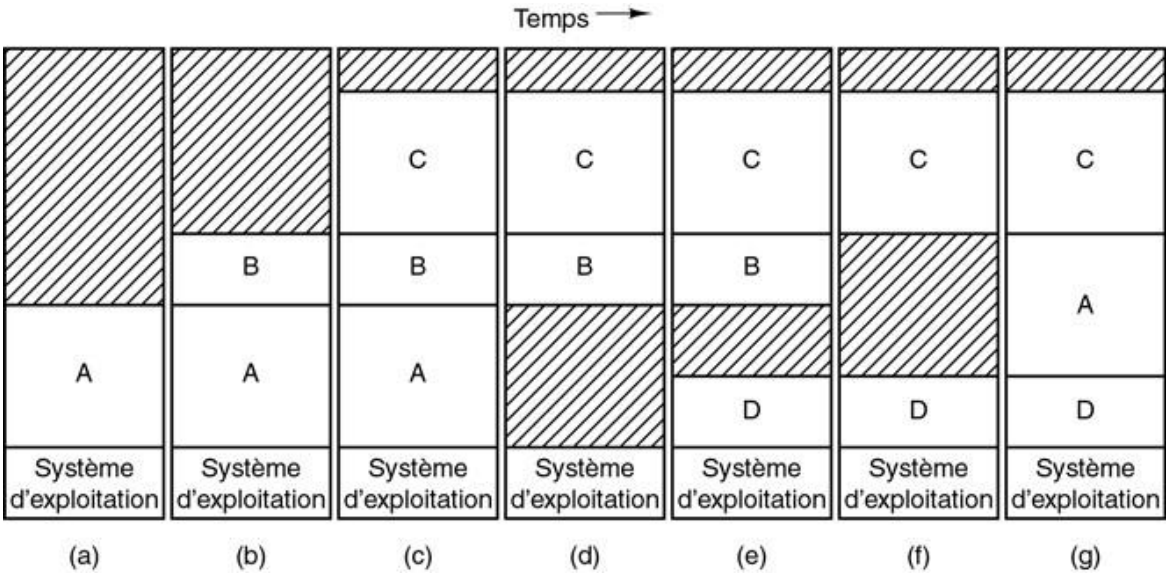


Figure 3 : Gestion de la mémoire avec va et vient

4.3.2 Opérations sur la mémoire

Le compactage de la mémoire permet de regrouper les espaces inutilisés. Très coûteuse en temps UC, cette opération est effectuée le moins souvent possible. S'il y a une requête d'allocation dynamique de mémoire pour un processus, on lui alloue de la place dans le tas (heap) si le SE le permet, ou bien de la mémoire supplémentaire contiguë à la partition du processus (agrandissement de celle-ci). Quand il n'y a plus de place, on déplace un ou plusieurs processus :

- soit pour récupérer par ce moyen des espaces inutilisés,
- soit en allant jusqu'au recouvrement. A chaque retour de recouvrement (swap), on réserve au processus une partition un peu plus grande que nécessaire, utilisable pour l'extension de la partition du processus venant d'être chargé ou du processus voisin. Il existe trois façons de mémoriser l'occupation de la mémoire : les tables de bits (bits maps), les listes chaînées et les subdivisions (buddy).

Comme on le voit sur le schéma suivant on distingue trois méthodes de compactage :

- La première consiste tout simplement à recopier de mémoire à mémoire le programme à déplacer. Elle monopolise le processeur pour effectuer la copie ;
- La seconde effectue une copie du programme à déplacer vers le disque, puis une seconde copie du disque vers la mémoire au nouvel emplacement. Curieuse *a priori*, cette méthode s'explique et se justifie si le transfert de mémoire à disque et

de disque à mémoire est effectué par un canal d'E/S, processeur spécialisé dans les échanges, qui laisse donc le processeur libre pendant les transferts. La recopie coûte toutefois un certain ralentissement du processeur, par vol de cycles.

- La dernière méthode est encore plus curieuse, mais se déduit de la seconde. On utilise deux canaux d'E/S, et on les boucle. L'utilisation de deux canaux est plus coûteuse que l'usage d'un disque, mais la copie est plus rapide. Il faut tout de même noter que les deux canaux vont voler des cycles au processeur de calcul, et l'un à l'autre, ce qui ralentit un peu.

Rq : Quelle que soit la méthode choisie, le compactage est coûteux.

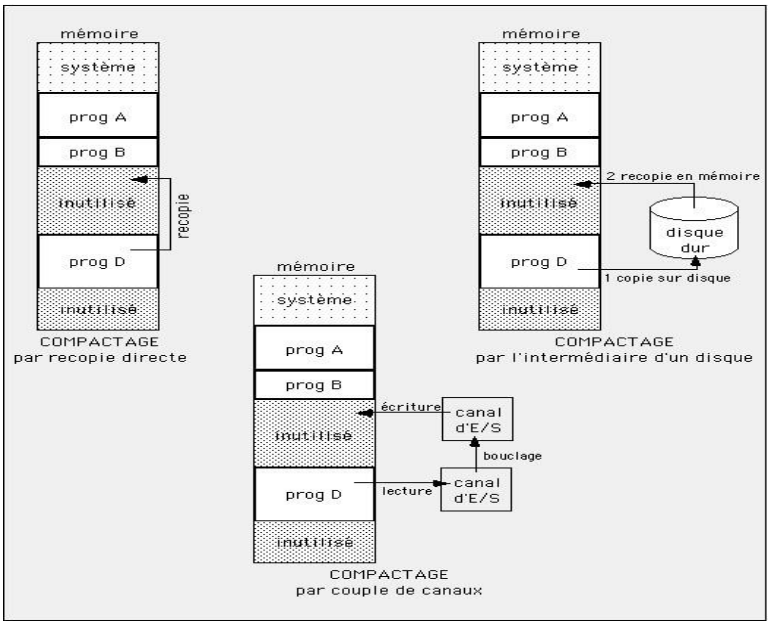


Figure 4 : Différentes stratégies de compactage

4.3.3 Gestion de la mémoire par table de bits

On divise la MC en unités d'allocations de quelques octets à quelques Ko. A chaque unité, correspond un bit de la table de bits : valeur 0 si l'unité est libre, 1 sinon. Cette table est stockée en MC. Plus la taille moyenne des unités est faible, plus la table occupe de place. A un retour de recouvrement (swap), le gestionnaire doit rechercher suffisamment de 0 consécutifs dans la table pour que la taille cumulée de ces unités permette de loger le nouveau processus à implanter en MC, avec le choix entre trois algorithmes d'allocation possibles :

- **First-fit** : la première zone libre : en parcourant la liste libre, on retient la première zone suffisante ; on alloue la partie suffisante pour le programme, et le reste devient une nouvelle zone libre plus petite.

- **Best-fit :** le meilleur ajustement, on recherche dans la liste la plus petite zone possible. Cette méthode est plus coûteuse et peut créer ainsi de nombreuses petites unités résiduelles inutilisables (fragmentation nécessitant un compactage ultérieur), mais évite de couper inutilement une grande zone.
- **worst fit :** le plus grand résidu, on recherche la zone qui laissera le plus petit résidu avec le risque de fractionnement regrettable des grandes unités.

4.3.4 Gestion de la mémoire par liste chaînée

On utilise une liste chaînée des zones libres en MC. On applique :

- soit l'un des algorithmes précédents,
- soit un algorithme de placement rapide (quick fit) : on crée des listes séparées pour chacune des tailles les plus courantes, et la recherche est considérablement accélérée.

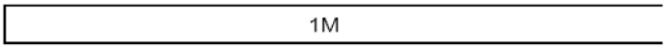
A l'achèvement d'un processus ou à son transfert sur disque, il faut du temps (mise à jour des listes chaînées) pour examiner si un regroupement avec ses voisins est possible pour éviter une fragmentation excessive de la mémoire.

En résumé, les listes chaînées sont une solution plus rapide que la précédente pour l'allocation, mais plus lente pour la libération.

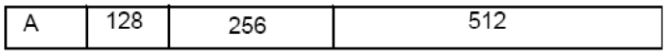
4.3.5 Gestion de la mémoire par subdivisions (ou frères siamois)

Cet algorithme utilise l'existence d'adresses binaires pour accélérer la fusion des zones libres adjacentes lors de la libération d'unités. Le gestionnaire mémorise une liste de blocs libres dont la taille est une puissance de 2 (1, 2, 4, 8 octets, ..., jusqu'à la taille maximale de la mémoire).

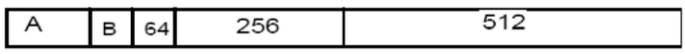
Par exemple, avec une mémoire de 1 Mo, on a ainsi 251 listes. Initialement, la mémoire est vide. Toutes les listes sont vides, sauf la liste 1 Mo qui pointe sur la zone libre de 1 Mo :



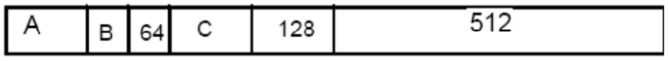
Un processus A demande 70 Ko : la mémoire est fragmentée en deux compagnons (buddies) de 512 Ko; l'un d'eux est fragmenté en deux blocs de 256 Ko; l'un d'eux est fragmenté en deux blocs de 128 Ko et on loge A dans l'un d'eux, puisque 64 < 70 < 128 :



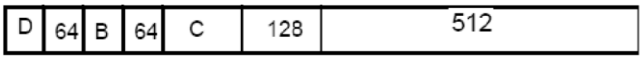
Un processus B demande 35 Ko : l'un des deux blocs de 128 Ko est fragmenté en deux de 64 Ko et on loge B dans l'un d'eux puisque 32 < 35 < 64 :



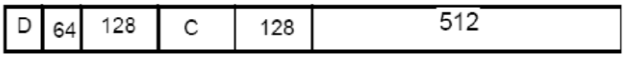
Un processus C demande 80 Ko : le bloc de 256 Ko est fragmenté en deux de 128 Ko et on loge C dans l'un d'eux puisque 64 < 80 < 128 :



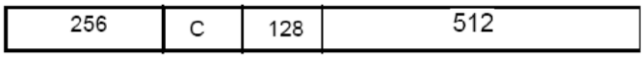
A s'achève et libère son bloc de 128 Ko. Puis un processus D demande 60 Ko : le bloc libéré par A est fragmenté en deux de 64 Ko, dont l'un logera D :



B s'achève, permettant la reconstitution d'un bloc de 128 Ko :



D s'achève, permettant la reconstitution d'un bloc de 256 Ko , etc...



L'allocation et la libération des blocs sont très simples. Mais un processus de taille 2n + 1 octets utilisera un bloc de 2n+1 octets ! Il y a beaucoup de perte de place en mémoire. Dans certains systèmes, on n'alloue pas une place fixe sur disque aux processus qui sont en mémoire.

On les loge dans un espace de va et vient (swap area) du disque. Les algorithmes précédents sont utilisés pour l'affectation.

4.4 Gestion avec recouvrement, avec pagination ou segmentation

4.4.1 Notion de mémoire virtuelle

La taille d'un processus doit pouvoir dépasser la taille de la mémoire physique disponible, même si l'on enlève tous les autres processus. Afin d'assurer une extension de la mémoire il existe 2 manières :

- En découpant un programme en une partie résidente en mémoire vive et une partie chargée uniquement en mémoire lorsque l'accès à ces données est nécessaire.
- En utilisant un mécanisme de **mémoire virtuelle**, consistant à utiliser le disque dur comme mémoire principale et à stocker uniquement dans la RAM les instructions et les données utilisées par le processeur. Le système d'exploitation réalise cette opération en créant un fichier temporaire (appelé fichier **SWAP**, traduit "**fichier d'échange**") dans lequel sont stockées les informations lorsque la quantité de mémoire vive n'est plus suffisante. Cette opération se traduit par une baisse considérable des performances, étant donné que le temps d'accès du disque dur est extrêmement plus faible que celui de la RAM. Lors de l'utilisation de la mémoire

virtuelle, il est courant de constater que la LED du disque dur reste quasiment constamment allumée et dans le cas du système Microsoft Windows qu'un fichier appelé "*win386.swp*" d'une taille conséquente, proportionnelle aux besoins en mémoire vive, fait son apparition.

La **Mémoire Virtuelle** est une mémoire idéale, dont les adresses commencent à 0, et de capacité non limitée ; elle a pour but principal de pouvoir exécuter des processus sans qu'ils soient logés en mémoire en leur totalité ; sans recours à la mémoire virtuelle, un processus est entièrement chargé à des adresses contiguës ; avec le recours à la mémoire virtuelle, un processus peut être chargé dans des pages ou des segments non contigus. On appelle **Espace Virtuel** l'ensemble des adresses possibles ; il est fixé par le nombre de bits qui constitue une adresse virtuelle. On parlera d'adresse réelle et d'adresse virtuelle. Toute la difficulté consiste à traduire une adresse virtuelle (A.V.) en une adresse réelle (A.R.) accessible sur la machine.

4.4.2 La pagination

L'espace d'adressage d'un processus est divisé en petites unités de taille fixe appelées **pages**. La MC est elle aussi découpée en unités physiques de même taille appelées **cadres**. Les échanges entre MC et disques ne portent que sur des pages entières. De ce fait, l'espace d'adressage d'un processus est potentiellement illimité (limité à l'espace mémoire total de la machine). On parle alors d'**adressage virtuel**.

Pour un processus, le système ne chargera que les pages utilisées. Mais la demande de pages à charger peut être plus élevée que le nombre de cadres disponibles. Une gestion de l'allocation des cadres libres est nécessaire.

Dans un SE sans mémoire virtuelle, la machine calcule les adresses physiques en ajoutant le contenu d'un registre de base aux adresses relatives contenues dans les instructions du processus. Dans un SE à pagination, un sous-ensemble inséré entre l'UC et la MC, **la MMU** (Memory Management Unit ou unité de gestion de la mémoire) traduit les adresses virtuelles en adresses physiques.

La MMU mémorise :

- Les cadres physiques alloués à des processus (sous forme d'une table de bits de présence)
- Les cadres mémoire alloués à chaque page d'un processus (sous forme d'une table des pages)

On dira qu'une page est **mappée** ou **chargée** si elle est physiquement présente en mémoire.

adresses virtuelles

0-4 K

4-8K

8-12 K

12-168-12 K K

16-20 K

20-24 K

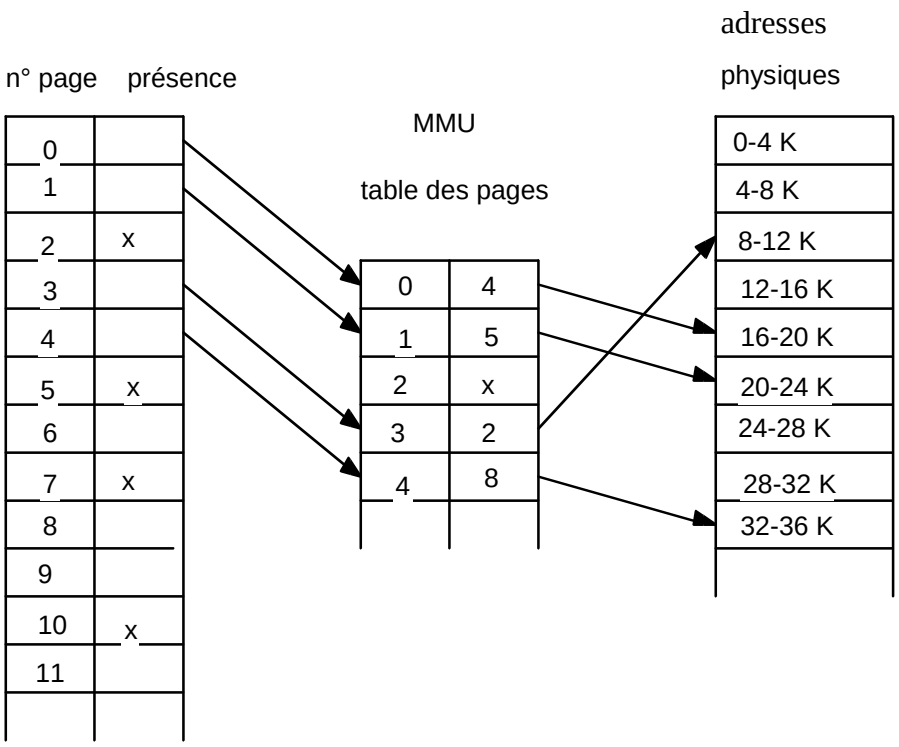
24-28 K

28-32 K

32-36 K 36-40 K

40-44 K

44-48 K



Dans l'exemple précédent, les pages ont une taille de 4 Ko. L'adresse virtuelle 12292 correspond à un déplacement de 4 octets dans la page virtuelle 3 (car 12292 = 12288 + 4 et 12288 = 12*1024). La page virtuelle 3 correspond à la page physique 2. L'adresse physique correspond donc à un déplacement de 4 octets dans la page physique 2, soit : (8*1024) + 4 = 8196.

Par contre, la page virtuelle 2 n'est pas mappée. Une adresse virtuelle comprise entre 8192 et 12287 donnera lieu à **un défaut de page**. Il y a défaut de page quand il y a un accès à une adresse virtuelle correspondant à une page non mappée. En cas de défaut de page, un déroutement se produit (trap) et le processeur est rendu au SE. Le système doit alors effectuer les opérations suivantes :

- Déterminer la page à charger (page victime)
- Déterminer la page à décharger sur le disque pour libérer un cadre
- Lire sur le disque la page à charger
- Modifier la table de bits et la table de pages

La sélection de page est réalisée par les algorithmes de remplacement. Le principe de fonctionnement de la majorité de ces algorithmes repose sur le principe de localité. Ces algorithmes sont en général divisés en deux grandes catégories :

- les algorithmes dépendants de l'utilisation des données: LRU, LFU, etc…
- les algorithmes indépendants de l'utilisation des données : aléatoire, FIFO.

Ci-dessous sont listés quelques d'algorithmes.

a. Algorithme optimal

Cet algorithme, utilise la mémoire cache de manière optimale : il retire la page qui sera référencée le plus tard possible. Cet algorithme doit connaître pour chaque page le nmbre d'instructions à exécuter avant qu'elle ne soit référencée cause pour laquelle set algorithme est irréalisable. Cet algorithme constitue néanmoins un excellent moyen afin de mesurer l'efficacité d'un algorithme de remplacement en fournissant une référence.

b. NRU (not recently used)

Le SE référence chaque page par deux bits R (le plus à gauche) et M initialisés à 0. A chaque accès en lecture à une page, R est mis à 1. A chaque accès en écriture, M est mis à 1. A chaque interruption d'horloge, le SE remet R à 0.

En cas de défaut de page, on retire une page au hasard dans la catégorie non vide de plus petit index (RM).

c. FIFO (first in, first out)

Le SE indexe chaque page par sa date de chargement et constitue une liste chaînée, la première page de la liste étant la plus anciennement chargée et la dernière la plus récemment chargée. Le SE replacera en cas de nécessité la page en tête de la liste et chargera la nouvelle page en fin de liste.

Deux critiques à cet algorithme :

- ce n'est pas parce qu'une page est la plus ancienne en mémoire qu'elle est celle dont on se sert le moins
- l'algorithme n'est pas stable : quand le nombre de cadres augmente, le nombre de défauts de pages ne diminue pas nécessairement.

d. LRU (least recently used)

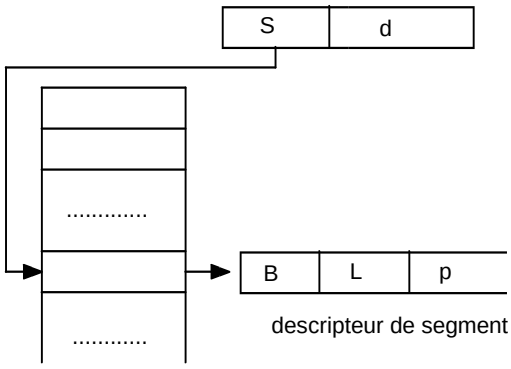
En cas de nécessité, le SE retire la page la moins récemment référencée. Pour cela, il indexe chaque page par le temps écoulé depuis sa dernière référence et il constitue une liste chaînée des pages par ordre décroissant de temps depuis la dernière référence. L'algorithme est stable. Mais il nécessite une gestion coûteuse de la liste qui est modifiée à chaque accès à une page.

4.4.3 La segmentation

Dans cette solution, l'espace d'adressage d'un processus est divisé en segments, générés à la compilation. Chaque segment est repéré par son numéro S et sa longueur variable L. Un segment est un ensemble d'adresses virtuelles contiguës.

Contrairement à la pagination, la segmentation est "connue" du processus : une adresse n'est plus donnée de façon absolue par rapport au début de l'adressage virtuel; une adresse est donnée par un couple (S , d), où S est le n° du segment et d le déplacement dans le segment, $d \in [0 , L [$.

Pour calculer l'adresse physique, on utilise une **table des segments** :



B : adresse de base (adresse physique de début du segment)

L : longueur du segment ou limite

p : protection du segment

L'adresse physique correspondant à l'adresse virtuelle (S , d) sera donc B + d, si d <= L

La segmentation simplifie la gestion des objets communs (rangés chacun dans un segment), notamment si leur taille évolue dynamiquement.

Chapitre 5

Gestion des périphériques

Objectif général : À la fin de ce chapitre, l'étudiant doit être capable de connaître l'utilité et l'organisation des périphériques d'entrée/sortie.

Objectifs spécifiques: ce chapitre permet :

- De connaître le principe de fonctionnement des périphériques d'entrée/sortie
- De comprendre le principe de communication entre UC et E/S
- De comprendre et maîtriser les algorithmes d'ordonnancement des disques de stockage

5.1 Introduction

La gestion des périphériques représente peut-être le défi le plus considérable d'un système d'exploitation. Ce dernier doit contrôler tout un ensemble de périphériques avec des différences multidimensionnelles. Rapidité du périphérique, volume des informations, service proposé, direction du flux d'informations et protocoles de communications sont autant de grandeurs aux éventails très larges. Outre cette diversité, le système d'exploitation doit pouvoir traiter un grand nombre de périphériques, ce traitement doit se dérouler dans un environnement parallélisé. Les périphériques agissent en général indépendamment de l'UC, en fonction de leur propre fréquence et synchronisation. Le système d'exploitation, qui la plupart du temps s'exécute sur une seule UC, doit donc gérer des requêtes simultanées en provenance d'un grand nombre de périphérique.

5.2 Organisation des dispositifs d'E/S

Même si certains ordinateurs sont différents dans les détails, ils sont conçus autour de la même philosophie. Les dispositifs d'E/S, le mémoire et l'UC communiquent par le biais d'un bus de communication.

Les machines les plus simples présentent un seul bus de communication. Mais les communications ne peuvent avoir lieu qu'entre deux éléments à la fois. Un dispositif appelé « Arbitre de bus » décide quel périphérique est autorisé à communiquer au prochain cycle. Celui-là peut communiquer avec n'importe quel autre de son choix.

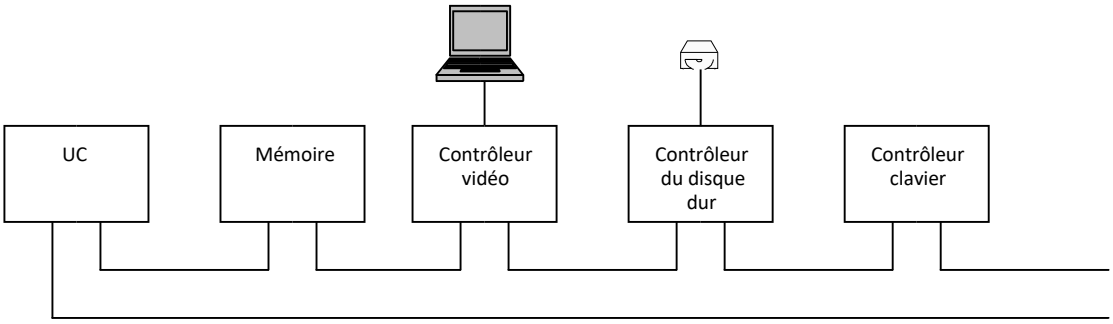


Figure 1 : Architecture à bus unique

En principe, le bus est attribué à l'UC afin qu'elle puisse communiquer avec la mémoire. Des accès fréquents à la mémoire et une vitesse relativement rapide de l'UC conduisent à une utilisation élevée du bus par cette dernière. Bien que le bus leur soit fréquemment nécessaire, les E/S ont des besoins en communication généralement plus urgents que les requêtes de l'UC. C'est pourquoi les requêtes des périphériques d'E/S reçoivent souvent une priorité plus élevée. Le processus consistant à retirer le bus de l'UC pour l'attribuer à un périphérique est appelé vol de cycle.

On peut trouver des bus multiples sur des machines pour des raisons de parallélisme et d'ajustement des performances. Les bus multiples permettent à plusieurs communications de se dérouler simultanément. L'UC peut par exemple communiquer avec un port série sur un bus alors qu'un disque communique avec la mémoire sur un autre. Cependant l'avantage des bus multiples est assez limité. La plupart des communications impliquent soit la mémoire, soit l'UC. Sans matériel multi-accès particulier, ils ne peuvent communiquer qu'avec un seul dispositif à la fois.

Les architectures PC les plus récentes ont souvent recours à 3 types de bus outre celui du processeur de l'UC : le bus standard de connexion des périphérique, **bus PCI**. Par ailleurs, **un bus mémoire** spécial permet des communications optimisées entre l'UC et la mémoire ; **un bus ISA** (Industry standard architecture) est relié au bus PCI pour offrir une compatibilité descendante pour les anciens périphériques.

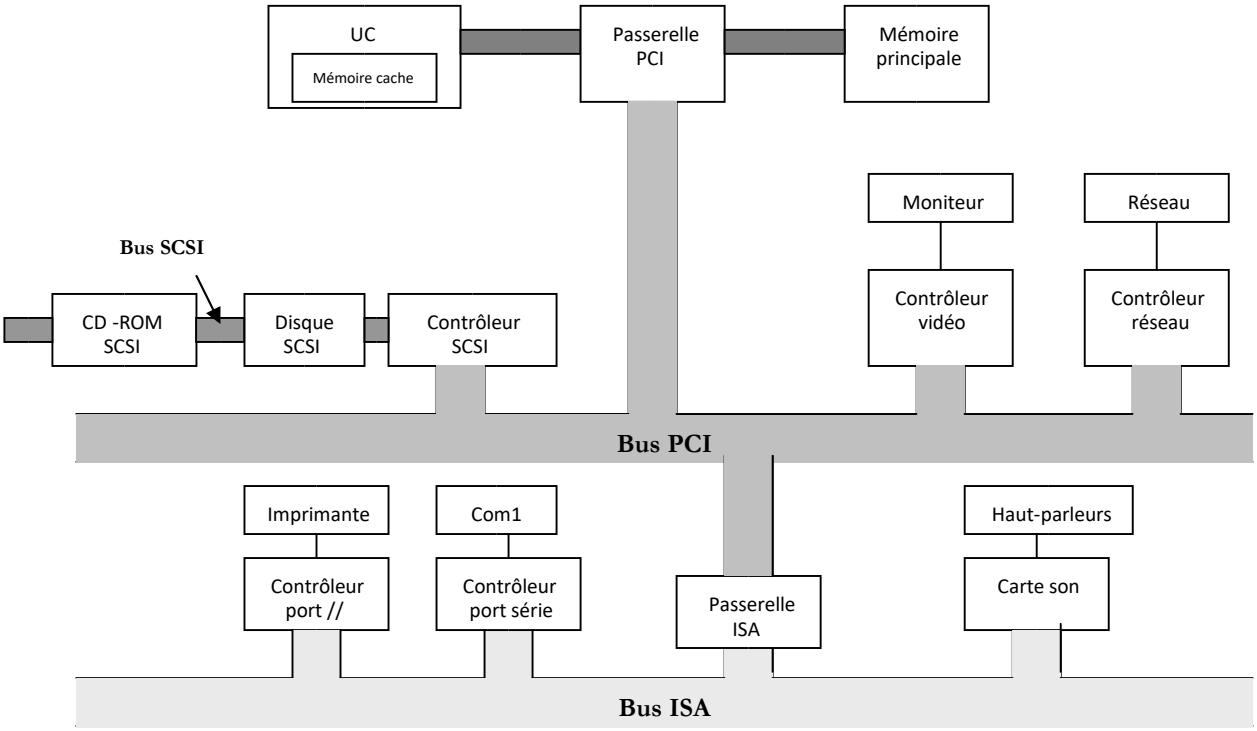


Figure 2: Architecture à bus multiple

5.3 Contrôle des E/S

Dans le modèle le plus simple l'UC communique directement avec les périphériques dE/S et prend en charge le contrôle des moindres détails de l'opération du périphérique. Ce type de communication est de plus en plus rare (encore dans les systèmes embarqués).

Les nouveaux systèmes incorporent la notion de « contrôleur de périphériques ». Une commande classique de l'UC au contrôleur peut être le lancement d'une opération de lecture pour un octet d'informations depuis un appareil en série ou d'un secteur d'informations depuis un disque. Le contrôleur de périphérique transmet au périphérique les commandes plus détaillées nécessaires à la réalisation de l'opération requise. En déchargeant cette responsabilité sur le contrôleur, l'UC est libre d'accomplir simultanément d'autres tâches. Chaque dispositif dE/S possède un contrôleur spécifique.

La plupart des contrôleurs peuvent servir plusieurs périphériques à la fois.

5.4 Ports d'E/S

Pour réaliser les E/S, l'UC doit communiquer avec les modules d'E/S, qu'il s'agisse d'un périphérique ou d'un contrôleur ou d'un canal. Chaque module d'E/S contient un ou plusieurs registres servant à la communication avec le processeur.

En écrivant dans ces registres, le SE ordonne au périphérique de délivrer des données, d'en accepter, de s'activer, désactiver ou effectuer une opération donnée (commande de périphérique).

En lisant les registres, le SE connaît l'état du périphérique.

De nombreux périphériques sont équipés d'un tampon de données que le SE peut écrire ou lire. Par exemple, la RAM vidéo contient les pixels affichés à l'écran. Cette RAM vidéo est le tampon de données relatif au périphérique vidéo (carte graphique).

5.5 Communication entre UC et E/S

La communication entre les modules d'E/S et l'UC suivent l'un des quatre protocoles suivant :

- **Attente active:** L'UC émet une commande au module d'E/S pour lancer une opération d'E/S. L'UC entre dans une boucle pour vérifier si l'opération est achevée ou non et ensuite tester si l'opération est finie avec succès ou non. Les processus sont bloqués jusqu'à ce que l'E/S finisse.
- **La scrutation:** L'UC lance les opérations d'E/S puis retourne pour exécuter les processus du système. Après des délais périodiques, l4UC fait une scrutation des périphériques d'E/S pour voir si l'un d'eux a fini sa tâche ou non. Ce protocole est plus efficace que l'attente active, mais il est plus couteux.
- **Interruption:** Dans ce cas, le périphérique d'E/S se charge d'informer l'UC de l'achèvement de l'E/S qu'il prépare. L'interruption utilisée par un module est généralement configurable et unique ce qui offre le parallélisme.
- **Accès direct à la mémoire:** Les composants d'accès direct à la mémoire (DMA : Direct Memory Access) permettent aux modules d'E/S de lire ou écrire des données directement depuis le mémoire. Avec cette technique, les données ne doivent pas transiter par l'UC. Cet accès est utile pour des dispositifs tels que les disques.

5.6 Les pilotes des périphériques

Les pilotes constituent la partie logicielles qui contrôle et interagit directement avec le périphérique d'E/S.

Le SE peut exiger que pour chaque périphérique, il existe un ensemble de fonctions mises en œuvre, telle que :

- **Open** : réalise les tâches de démarrage avant l'accès au périphérique.
- **Close** : fermer le périphérique après l'utilisation
- **Schedule** : ordonnance une requête d'E/S ou plusieurs avant d'être passés au périphérique (pilote…)
- **Startio** : vérifie si le périphérique est actif ; si ce n'est pas le cas, lance la prochaine opération d'E/S sur la file d'ordonnancement du périphérique.

- **Interrupt** : routine exécutée lorsque le périphérique envoie une interruption à l'UC.

Compte tenu des différents types de périphériques pouvant être connectés à certains ordinateurs, il ne serait pas pratique d'inclure les pilotes de tous les périphériques éventuels dans le SE.

Les SE doivent être configurés en fonctions des périphériques. Ceci se fait par ajout ou suppression de périphériques (modifier l'image du SE).

5.7 Les périphériques

5.7.1 Les périphériques graphiques

L'un des plus grands problèmes est le volume d'informations qui doit être transmis pour décrire l'affichage sur les moniteurs.

L'image sur les moniteurs est affichée sous formes de points nommés pixels. Les moniteurs diffèrent selon le nombre de pixels et les couleurs qu'ils peuvent adopter.

Les premiers moniteurs étaient composés de 200 lignes de 320 pixels noir et blanc. De nos jours, les résolutions de 800×600, 1024×768 et 1280×960 sont les plus répondues.

L'illumination de chaque pixel est contrôlée par des valeurs stockées en mémoire vidéo :

- Les affichages monochromes de base n'ont besoin que d'un bit par pixel.
- Les affichages par niveaux de gris (255 niveaux) ont besoin de 8 bits.

Les affichages couleurs réelles ont besoin de 24 bits : chaque pixels est généré par la combinaison des trois couleurs primaires (rouge, vert et bleu : RGB ou RVB) ; 8 bits pour chacune de ces couleurs. Avec un affichage couleurs réelles et pour un moniteur vidéo 1024×768, il nous faut 2,3 Mo pour stocker une valeur de 24 bits pour chaque pixel.

Pour modifier une image à l'écran, de nouvelles données doivent être écrites en mémoire vidéo. Compte tenu du grand volume de données multimédia cela peut représenter une charge importante sur le système. Par exemple, une animation vidéo qui nécessite 25 image/seconde et 2,3 Mo par image réécrit 25 fois dans la mémoire vidéo par seconde donc un total de 58 Mo/seconde.

Cette exigence de transfert a poussé l'évolution des conceptions matérielles.

- Bus PCI : capacités de transfert de 132 Mo/S
- Le nouveau bus graphique AGP (Accelerated graphics Port) possède un taux de transfert de 528 Mo/S.

5.7.2 Les disques de stockage

Le disque peut être considéré comme le seul périphérique d'E/S commun à tous les ordinateurs.

Même les moniteurs et les claviers ne sont pas indispensables sur des systèmes tels que les serveurs.

Il existe de nombreuses tailles et vitesses de disques et l'information peut être stockée de manière optique ou magnétique. L'unité élémentaire de stockage d'informations est le secteur.

- Les DVD et CD-ROM : les secteurs forment une longue spirale qui s'éloigne en tournant du centre du disque.
- Sur les disquettes et disques dans le support tourne à vitesse constante. Les secteurs sont organisés en pistes. Les pistes sont des cercles concentriques autour du centre.
- Certains disques stockent le même nombre de secteurs pour chaque piste. D'autres disques placent plus de secteurs sur des pistes externes.
- Les disques contiennent un ou plusieurs plateaux de support. Certain proposent d'utiliser les deux faces du plateau et propose deux têtes de lecture/écriture par plateau.

5.7.3 Ordonnancement du disque dur

Si un disque est appelé à répondre à plusieurs E/S, il doit ordonnancer ces requêtes suivant certains algorithmes d'ordonnancement. La performance de ces algorithmes se mesure par le total des mouvements par tête.

- **FIFO :** premier entré premier servi.
- **Priorité :** la requête venant du processus ayant la priorité la plus élevée est servi la première.
- **SSTF (Shortest Seek Time First):** traduit plus court positionnement d'abord, répond à la requête dont la position de la piste est la plus proche de celle en cours.
- **SCAN :** avance et recule la tête de Lecture/écriture entre la piste la plus interne et la plus externe et satisfait en route toutes les requêtes de la piste en suspens.
- **LOOK :** le même que SCAN, mais s'il n'y a pas de requêtes dans un sens la tête ne fait pas le déplacement de façon inutile.
- **C-SCAN et C-LOOK :** une fois la dernière piste est atteinte, les algorithmes retournent sur la piste de départ.
- **N-step SCAN :** file de requête divisée en sous files de longueur N. Ces files sont ordonnancées en FIFO. Au sein des files les requêtes sont ordonnancées en SCAN.
- **FSCAN :** tel que le précédent, mais seulement deux sous files illimitées : Une en cours de traitement et l'autre pour les nouvelles requêtes (celles qui viennent lors du traitement).

Chapitre 6

Gestion de fichiers

Objectif général : À la fin de ce chapitre, l'étudiant doit être capable de Connaître la notion de fichier et ses caractéristiques.

Objectifs spécifiques: ce chapitre permet :

- De Connaître la notion de répertoires et partitions
- De Connaître les différentes stratégies d'allocation de blocs pour fichiers et de gestion d'espace disque.

6.1 Introduction

Le volume des données traitées par les applications informatiques atteignant plusieurs mégas et giga octets, ces données ne peuvent pas être stockées dans la mémoire centrale. On souhaite également disposer d'un stockage à long terme qui ne disparaisse pas lorsqu'on éteint la machine. Le principe consiste à stocker ces données dans des mémoires secondaires sous forme de fichiers, c'est-à-dire de suites de blocs (la plus petite unité que le périphérique de stockage est capable de gérer). Le contenu de ces blocs, simple suite de chiffres binaires, peut être interprété selon le format de fichier comme des caractères, des nombres entiers ou flottants, des codes d'opérations machines, des adresses mémoires, etc... L'échange entre les deux types de mémoires se fait ensuite par transfert de blocs.

L'objectif du système de fichier est de permettre l'accès au contenu du fichier (l'ouverture du fichier, sa recopie à un second emplacement ou sa suppression) à partir de son chemin d'accès, formé d'un nom précédé d'une liste de répertoires imbriqués.

6.2 Les fichiers

6.2.1 Définition :

Un fichier est une collection logique d'information. Un système de fichiers est une collection de fichiers.

6.2.2 Le système de gestion de fichiers

Une des fonctions d'un SE est de masquer les spécificités des disques et des autres périphériques d'E/S et d'offrir au programmeur un modèle de manipulation des fichiers agréable et indépendant du matériel utilisé. Les appels système permettent de créer des fichiers, de les supprimer, de lire et d'écrire dans un fichier. Il faut également ouvrir un fichier avant de

l'utiliser, le fermer ultérieurement. Les fichiers sont regroupés en répertoires arborescents; ils sont accessibles en énonçant leur chemin d'accès (chemin d'accès absolu à partir de la racine ou bien chemin d'accès relatif dans le cadre du répertoire de travail courant).
Le SE gère également la protection des fichiers.

6.3 Répertoires, noms de fichiers et partitions

6.3.1 Répertoire

Les systèmes de fichiers permettent aux utilisateurs d'organiser des fichiers et d'autres objets de systèmes de fichiers au moyen de répertoires. Un répertoire (dossier) est généralement défini comme un objet du système de fichier contenant d'autres objets de systèmes de fichiers. Les entrées des répertoires déterminent le chemin d'accès absolu ou nom associé à un objet du système de fichiers. En commençant par le répertoire racine, le chemin d'accès absolu est construit en concaténant la séquence des noms parcourus séparé par un « *backslah* » : \ sous windows ou Dos et un « *slash* » / sous linux ou Unix.
La plupart des systèmes prennent en charge la notion de répertoire courant ; au lieu de recourir à un chemin d'accès complet, s'avérant très long, il est possible de spécifier un chemin d'accès relatif.

6.3.2 Noms de fichiers

Les noms de fichiers représentent des mécanismes d'abstraction qui permettent d'écrire des données et de les trouver plus tard : chaque fichier est référencé par son nom. Le nom d'un fichier possède une longueur maximale qu'il ne peut pas dépasser relativement au système d'exploitation (Dos : 8, Win_XP : 255). Le nom est généralement composé de la partie nom et la partie extension séparés par un point.

6.3.3 Partitions

Les fichiers sont généralement stockés sur des unités de mémoire secondaire ; la mémoire vive peut néanmoins être utilisée pour stocker des fichiers, tels que les fichiers temporaires, pour lesquels un accès rapide est souhaité. La notion de partition peut également entrer en compte pour déterminer sur quelle unité doit être stockée un fichier. Sur certains systèmes, comme Dos et Windows, la partition est spécifiée dans le chemin d'accès. Le nom c:\rules\II2.txt (sous Dos ou Windows) indique que l'objet du système de fichiers nommé \rules\II2.txt se trouve sur la partition c :.

6.4 Type des objets du système de fichiers

Dans de nombreux systèmes d'exploitation, les noms des systèmes de fichiers peuvent faire référence à des objets qui ne sont ni des fichiers ni des répertoires. Les objets qui peuvent se trouver dans un système de fichiers comprennent les éléments suivants :

- **Un raccourci :** un raccourci est un pointeur vers un autre nom dans le système de fichiers. Dans la plupart des cas, le fait de se référer à un raccourci revient à se référer au nom pointé par le raccourci. La suppression du nom vers lequel pointe le raccourci laisse généralement le raccourci en suspens.
- **Un périphérique :** un élément matériel, comme un port parallèle.
- **Un tube :** un canal de communication entre 2 processus. Un processus envoie des données dans le tube l'autre processus lit ces données depuis le tube. Le tube met en mémoire tampon les données écrites à l'intérieur, jusqu'à ce qu'un autre processus lise les données. Le tampon est généralement de taille limitée et un processus qui écrit sur le tube doit être suspendu lorsque le tampon est plein. Les tubes peuvent être nommés ou non. Comme ile n'ont pas de nom les identifiant, les tubes sans nom ne sont généralement accessibles que par le processus les ayant créés ou les processus qui sont des descendant du processus.
- **De la mémoire partagée :** une allocation d'emplacement de mémoire utilisable par un ou plusieurs processus. Comparable à un fichier sur un disque virtuel.

6.5 Fonctions des systèmes de fichiers

Le système de fichier doit offrir aux utilisateurs la possibilité d'accomplir des opérations abstraites sur les objets au sein du système de fichiers. Il doit, au minimum, proposer les fonctions suivantes : créations, suppression, lecture et écriture.

6.6 Architecture du système de fichiers

6.6.1 Structure de fichiers

Les fichiers peuvent être structurés de deux manières sous formes de suites d'octets non structurés ou d'une suite d'enregistrements. Un fichier est une séquence d'enregistrements de longueur fixe qui ont la même structure interne. Un fichier prend la forme d'un arbre d'enregistrements qui ne sont pas nécessairement de même longueur. Un enregistrement est une collection logique d'informations (par exemple, une ligne de texte, des informations relatives à une personne). Les opérations d'E/S s'effectuent généralement en termes d'enregistrements. Le système d'exploitation peut gérer des structures d'enregistrements fixes et/ou variables. Les enregistrements peuvent à leur tour être divisés en champs, un champ

représentant une donnée élémentaire (telle que le nom et l'âge). La figure suivante décrit la strla structure logique d'un fichier

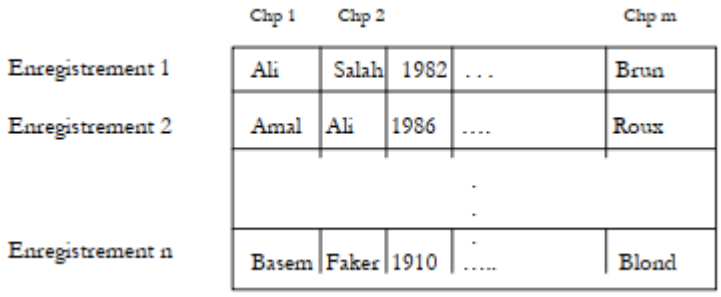


Figure 1 : Structure logique d'un fichier

Transparent à l'utilisateur, le système d'exploitation considère un fichier comme une collection de blocs logique à taille fixe. Un bloc est l'unité de base d'une opération dE/S entre le disque et la mémoire tampon du système de fichiers. Le disque en tant que tel est un ensemble de blocs physiques. Chacun d'entre eux stocke un bloc logique et éventuellement d'autres données administratives. La taille du bloc est un multiple de l'unité d'E/S de base fournie par le pilote du disque.

6.6.2 Méthodes d'accès

Il existe deux méthodes fondamentales pour accéder à des informations au sein d'un fichier : séquentielles et directes. Dans l'accès séquentiel, il faut accéder aux informations du fichier dans l'ordre dans lequel elles ont été stockées dans le fichier. L'accès se déroule de manière séquentielle depuis le début jusqu'à la fin. Les opérations de lecture ou d'écriture sur le fichier n'ont pas besoin de spécifier l'emplacement logique au sein du fichier, car le système d'exploitation maintient un pointeur de fichier qui détermine l'emplacement du prochain accès. Avec l'accès direct, il est possible d'accéder à tout emplacement logique à l'intérieur du fichier. Généralement, l'accès direct peut être réalisé de 2 manières : Soit en spécifiant l'emplacement logique auquel accéder comme un paramètre à l'opération de lecture ou d'écriture, soit en spécifiant l'emplacement d'une opération de positionnement pour qu'il soit appelé avant la lecture ou l'écriture. Les systèmes de base de données utilisent deux opérations d'accès au système d'exploitation élémentaire pour mettre en œuvre un large éventail de méthodes d'accès de haut niveau. Certains systèmes d'exploitation mettent en œuvre des méthodes d'accès de haut niveau par eux-même. Parmi toutes ces méthodes, l'accès indexé est peut-être le plus significatif. Avec ce dernier, chaque enregistrement de fichiers dispose d'un ou plusieurs

champs. Un champ sert de champ d'indexe. Les opérations de lecture et d'écriture comprennent un paramètre d'index. L'enregistrement avec la valeur d'index correspondante est l'enregistrement sur lequel est effectuée l'opération.

Pour toute méthode d'accès, les opérations de lecture et d'écriture peuvent être synchrones ou asynchrones. Les blocs d'E/S synchrones bloquent le processus jusqu'à la fin de l'opération d'E/S. Les E/S asynchrones renvoient immédiatement le contrôle au processus, laissant le processus libre de continuer à s'exécuter pendant lE/S. Si les opérations d'entrées sont asynchrones, il faut faire appel à certains mécanismes pour avertir le processus que l'opération est terminée. Pour cela, il est possible d'envoyer un signal au processus, d'attribuer une valeur particulière aux variables du processus ou de lancer un appel système spécial pour tester l'état de l'opération dE/S. Les opérations de sortie asynchrones peuvent recourir aux mêmes techniques de notifications ou n'offrir aucune notification.

L'E/S synchrone représente la norme qui simplifie considérablement la programmation d'application. La grande vitesse des opérations d'E/S de fichier n'incite en rien à l'utilisation des E/S asynchrones.

Cependant ces dernières peuvent parfois être employées avec un périphérique d'E/S.

6.6.3 Contrôle des droits d'accès

Le contrôle des droits d'accès établit une limite quant aux personnes pouvant accéder aux fichiers et à la manière dont elles peuvent y accéder. Le mécanisme de contrôle des droits d'accès le plus simple attribut un accès illimité à tous les utilisateurs. Il s'agit là du modèle de contrôle d'accès choisi par DOS. Sur un tel système, les utilisateurs qui souhaitent contrôler l'accès à leurs fichiers doivent mettre en place une limite d'accès physique (et sur le réseau) de leur machine.

Un aspect important du contrôle des droits d'accès est le type d'opérations à réaliser sur le fichier. Les opérations contrôlées comprennent entre autre :

- **La lecture :** lecture des informations contenues dans le fichier.
- **L'écriture :** écriture de nouvelles informations dans un fichier ou écrasement des informations d'un fichier.
- **L'adjonction :** écriture de nouvelles informations à la fin du fichier seulement.
- **La suppression :** suppression d'un fichier et libération de son espace de stockage en vue d'une utilisation dans d'autres fichiers.
- **La liste :** lecture des noms contenus dans un répertoire.

- **L'exécution :** chargement du contenu d'un fichier dans la mémoire principale et création d'un processus pour l'exécuter.
- **Le changement des droits d'accès :** modifications de certains droits d'accès d'utilisateur en vue d'une opération de contrôle.

L'autre caractéristique majeure du contrôle des droits d'accès est la manière dont il détermine ou non d'octroyer l'accès. Le mécanisme le plus commun consiste à baser la décision sur l'identité de l'utilisateur. Sur un système qui utilise une liste des droits d'accès, le système d'exploitation associe à chaque fichier le type d'opérations autorisé à chaque utilisateur. Dans un modèle de droits d'accès illimité, un ensemble indépendant de permissions est conservé pour chaque utilisateur, ce qui représente un volume important de données.

Un mécanisme de contrôle des droits d'accès limité réduit ce volume en regroupant les permissions d'accès pour un certain nombre d'utilisateurs ou de fichiers. Ainsi, de nombreux systèmes d'exploitation mettent en œuvre la notion de groupes d'utilisateurs. Chaque utilisateur et chaque fichier sont associés à un ou plusieurs groupes d'utilisateur. Au lieu d'avoir un ensemble de permissions d'accès pour chaque utilisateur, le fichier possède uniquement un ensemble de permissions d'accès pour son propriétaire et pour chaque groupe auquel il est associé. Tous les utilisateurs d'un groupe partagent les mêmes permissions d'accès. Un autre groupement fréquemment utilisé consiste à demander à ce que tous les fichiers d'un répertoire partagent les mêmes permissions.

L'autre base communément utilisée pour contrôler l'accès est le mot de passe. Pour réaliser une opération sur un fichier, un utilisateur doit spécifier le mot de passe de fichier associé à cette opération (un mot de passe de fichier est distinct de tout éventuel mot de passe d'ouverture de session). Comme avec les listes de droits d'accès, le groupement peut réduire le volume des données maintenues par le système d'exploitation (et réduire le nombre de mots de passe que doit retenir un utilisateur). Ainsi, tous les fichiers d'un répertoire peuvent partager le même mot de passe.

Les capacités constituent une variante intéressante des listes de droits d'accès, mais cependant moins répandue. Sur les systèmes basés sur les capacités, les droits d'accès, au lieu d'être associés aux fichiers, sont associés aux processus. Lorsqu'un accès au fichier est tenté, le système d'exploitation vérifie le droit correspondant au fichier dans les droits d'accès associés au processus.

6.6.4 Verrouillage des fichiers

Le verrouillage des fichiers offre aux processus la possibilité de mettre en œuvre un accès exclusif à un fichier. Trois grandes options existent dans la mise en œuvre du verrouillage :

✈ Le verrouillage peut se limiter à l'ensemble des fichiers ou la mise en œuvre peut autoriser de verrouiller certaines parties d'un fichier.

✈ Le verrouillage peut s'appliquer à tout accès ou il peut exister différents niveaux. Sur certains systèmes, il y à la fois des blocs de lecture et d'écriture. Un fichier verrouillé pour la lecture n'empêche pas un autre accès en lecture, mais l'accès en écriture par d'autre processus est refusé.

✈ Le verrouillage peut être soit obligatoire soit consultatif. Avec le verrouillage obligatoire, le système d'exploitation refuse l'accès pendant que le fichier est verrouillé. Avec le verrouillage consultatif, les primitives de verrouillage fournissent uniquement des informations sur l'état de verrouillage du fichier. L'accès n'est restreint que si un processus vérifie l'état de verrouillage du fichier et respecte le verrouillage qui est indiqué.

Dans certaines circonstances, un système d'exploitation peut mettre en œuvre un mécanisme de verrouillage implicite.

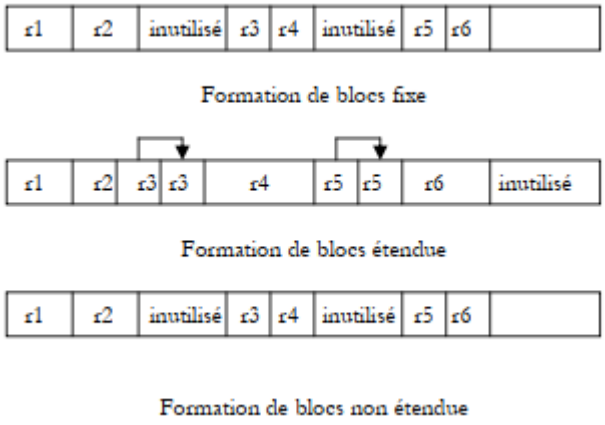
6.6.5 Attribution des blocs

La méthode d'attribution de blocs détermine comment les enregistrements d'un fichier sont attribués dans des blocs.

✈ **Attribution fixe des blocs :** pour les fichiers avec des enregistrements de taille fixe, un nombre intégral d'enregistrements est stocké dans chaque bloc. Aucun enregistrement ne peut être plus grand qu'un bloc. Si la taille du bloc n'est pas un multiple de la taille de l'enregistrement, il y aura de l'espace inoccupé à la fin du bloc. Le système d'exploitation peut calculer le bloc et l'offset à l'intérieur du bloc de tout enregistrement, en fonction de la taille de l'enregistrement et du bloc.

✈ **Attribution non étendue de blocs :** pour les systèmes avec des enregistrements de taille variable, plusieurs enregistrements peuvent être stockés dans chaque bloc, mais aucun ne peut s'étendre sur plusieurs blocs. Les enregistrements ne peuvent pas non plus être plus grands que la taille du bloc. L'espace à la fin d'un bloc est gaspillé si le prochain enregistrement est plus important que cet espace. Le système d'exploitation ne peut calculer l'emplacement d'un enregistrement, à mains qu'il ne connaisse la taille de tous les enregistrements qui le précède.

✈ **Attribution étendue de blocs :** les enregistrements peuvent être stockés dans plusieurs blocs. Il n'existe aucune limite quant à la taille d'un enregistrement et il n'y a pas d'espace inutilisé à l'intérieur d'un bloc. La seule manière de calculer l'emplacement d'un enregistrement consiste à additionner la taille de tous les enregistrements qui le précède.



Formation de blocs non étendue

6.7 Allocation

Il existe 3 modèles de base pour allouer l'espace de l mémoire auxiliaire à des fichiers. Le schéma d'allocation est chargé d'associer des blocs logiques d'un fichier à des blocs physiques de la mémoire auxiliaire. Dans la plupart des systèmes d'exploitation, la taille d'un bloc physique est une puissance de 2 comprise entre 512 et 4096.

6.7.1 Allocation contiguë

Le modèle le plus simple est l'allocation contiguë. Les blocs logiques d'un fichier sont stockés dans une partition de blocs physiques contigus. L'entrée du répertoire a uniquement besoin de stocker l'adresse de la mémoire auxiliaire de départ du fichier ainsi que la taille de ce dernier. L'emplacement physique de tout octet du fichier peut être calculé en ajoutant l'offset approprié à l'adresse de la mémoire auxiliaire de départ de fichier.

Lorsqu'un fichier est créé, l'allocation contiguë requiert une pré-allocation d'espace pour le fichier. Le système d'exploitation peut être conçu pour étendre l'allocation du fichier, si nécessaire. Si l'expansion est autorisée et que l'espace de stockage au-dehors de la fin du fichier est utilisé, un ou plusieurs fichiers doivent être déplacés pour loger le fichier le plus important. Les fichiers à déplacer doivent recevoir de nouvelles partitions sur le disque, puis être copiés sur ces nouvelles partitions.

6.7.2 Allocation chainée

Dans l'allocation chainée, les blocs physiques dans lesquels est stocké un fichier peuvent être dispersés dans l'ensemble de la mémoire auxiliaire. Les blocs physiques sont plus

importants que les blocs logiques et stockent à la fois le bloc logique et un pointeur vers le bloc physique dans lequel est stocké le prochain bloc logique du fichier. L'entrée du répertoire stocke l'emplacement du premier bloc physique. Le bloc physique associé au $N^{ème}$ bloc logique peut être déterminé uniquement en lisant les précédents blocs N-1 et en suivant les liens qu'ils contiennent. Les performances des opérations d'adjonctions (écriture à la fin d'un fichier) peuvent être améliorées de façon significative en incluant également dans l'entrée du répertoire un pointeur vers le dernier bloc de la file.

6.7.3 Allocation indexée

L'allocation indexée est une variante de l'allocation chainée. Le bloc physique stocke seulement le bloc logique, qui est par conséquent de la même taille qu'un bloc logique. Les liens vers les blocs physiques d'un fichier sont stockés de manière contiguë dans une table d'index. L'entrée du répertoire contient soit la table d'index soit un pointeur vers celle-ci.

Avec les allocations contiguës et chainées, il suffit au système de fichiers de stoker l'emplacement physique de départ du fichier. Toutes les autres adresses peuvent être déterminées à partir de l'emplacement de départ du fichier. Avec l'allocation indexée, le système de fichiers doit avoir une entrée d'index pour chaque bloc du fichier. Pour minimiser le volume d'espace requis dans les structures de répertoire, l'indexation à plusieurs niveaux peut être utilisée.

6.7.4 Espace libre

Outre le fait de garder la trace de l'emplacement de chaque fichier dans la mémoire auxiliaire, un système d'exploitation doit aussi être capable d'identifier les blocs physiques qui ne sont alloués à aucun fichier. Un certain nombre de mécanismes ont été utilisés pour maintenir la liste d'espace libre.

Un bloc est soit utilisé soit non utilisé ; son statut peut donc être stocké dans un seul bit. L'utilisation d'une série de bits, un bit par bloc physique, crée un champ de bits d'espace libre. Les bits sont regroupés en aussi peu de mots que possible. Bien que ce mécanisme fasse un usage efficace de l'espace, l'accès aux informations sur un bloc particulier est ralenti par le traitement supplémentaire nécessaire à l'extraction des informations en bits individuels depuis un mot.

Un tableau d'espace libre stocke les adresses de tous les blocs libres. Un bloc est utilisé si son adresse n'est pas dans le tableau. Un tableau permet une implémentation efficace d'opérations d'ajout ou de suppression des blocs de la liste d'espace libre.

Cette liste peut également être maintenue en créant une liste chaînée de tous les blocs libres. L'avantage de ce schéma est l'efficacité de l'utilisation de l'espace ; les blocs libres stockent eux-mêmes des informations relatives à la liste d'espace libre.