الجمهـــوريـــة الجـزائـريـــة الديمقـــراطيـــة الشعبيـــة
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التكـويـن و التعليـم المهنييـن
Ministère de la Formation et de l'Enseignement Professionnels

Centre National de la Formation et de l'Enseignement Professionnels à Distance (CNFEPD)

المـركـــز الـوطنـــي للتكويـن و التعلـيم المهـنيين عـــن بعـــد (م.و.ت.ت.م.ب)

SYSTÈME D'EXPLOITATION

LEÇON N°01 : LES CONCEPTS DE BASE DES SYSTÈMES D'EXPLOITATION

LEÇON N°02 : MÉCANISMES DE BASE D'EXÉCUTION DES PROGRAMMES

LEÇON N°03 : NOTION D'ORDONNANCEMENT DES PROCESSUS ET CES POLITIQUES

LEÇON N°04 : LES MÉCANISMES DE GESTION DE LA MÉMOIRE ET STRATÉGIE D'ALLOCATIONS DE LA MÉMOIRE

LEÇON N°05 : LES PÉRIPHERIQUES D'ENTRÉE/SORTI

LEÇON N°06 : LE SYSTÈME DE GESTION DE FICHIERS (SGF)

LEÇON N°01 : LES CONCEPTS DE BASE DES SYSTÈMES D'EXPLOITATION

OBJECTIF PÉDAGOGIQUE

À la fin de cette série, le stagiaire doit être capable de distinguer les concepts de base des systèmes d'exploitation.

PLAN DE LA LECON :

I- CONCEPTS DE BASE DES SYSTÈMES D'EXPLOITATION

INTRODUCTION AUX SYSTÈMES D'EXPLOITATION

1- Qu'est-ce qu'un système d'exploitation ?
2- Historique
3- Différents types de systèmes d'exploitation
4- Les modes d'exploitation

II- CONCEPTS ET FONCTIONS D'UN SYSTÈME D'E XPLOITATION

1- Propriétés communes des systèmes d'exploitation ;
2- Fonctions d'un système d'exploitation

RÉSUMÉ

EXERCICES D'APPLICATION

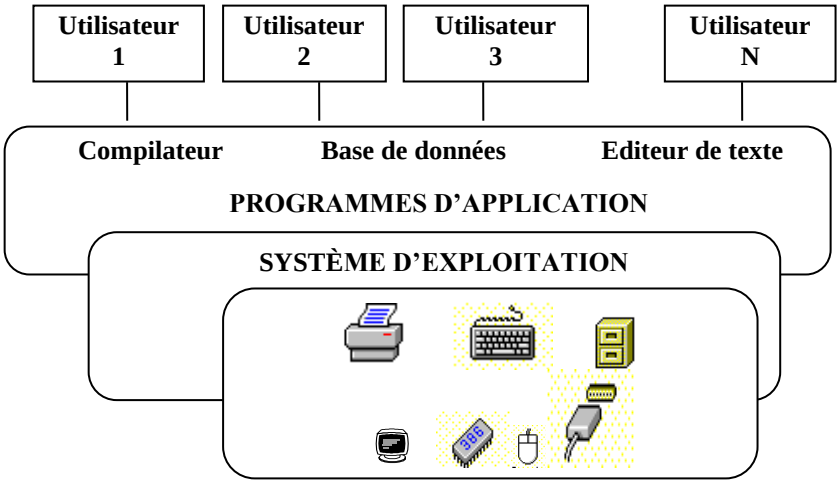
CORRIGÉ DES EXERCICES

I- CONCEPTS DE BASE DES SYSTÈMES D'EXPLOITATION :

INTRODUCTION AUX SYSTÈMES D'EXPLOITATION :

1- Qu'est-ce qu'un système d'exploitation ?

Un système informatique (ordinateur) est un ensemble de matériels et de logiciels (programmes) destinés à réaliser des tâches qui mettent en jeu le traitement automatique de l'information. Un tel système est relié au monde extérieur par des organes d'accès, qui lui permettent de communiquer avec des usagers humains ou d'interagir avec des dispositifs qu'il est chargé de commander ou de surveiller.



Le système d'exploitation est donc la partie logicielle de base (donc un ensemble de programmes élémentaires) qui coopèrent pour gérer, partager et rentabiliser les ressources du système informatique (ordinateur). Notons que ces ressources peuvent être :

1- Des ressources physiques ou matérielles tel que la mémoire, l'imprimante, etc. ...

2- Des ressources logiques tel que l'exécution des programmes des utilisateurs, la protection des données, la gestion des comptes utilisateurs, les mots de passe, etc. ...

Ce qui revient à dire que l'utilisateur n'est pas obligé de connaître comment fonctionne le matériel de son ordinateur (machine physique: circuits et cartes électroniques) pour l'utiliser, mais seulement savoir demander ce qu'il veut au système d'exploitation qui lui offre une machine virtuelle (boite noire) fournissant un très grand nombre de services.

2- Historique :

Dans les premiers ordinateurs, il n'existait pas de système d'exploitation au sens où nous l'entendons aujourd'hui. L'utilisateur devait connaître toutes les spécificités hardware de sa machine et réquisitionnait toutes les ressources de la machine à lui seul. Ce mode d'exploitation « dit porte ouverte » était non seulement très complexe mais aussi peu économique. C'est pourquoi apparurent les premiers systèmes d'exploitation « dits moniteurs d'enchaînement ». Ils permettaient l'exécution, l'un après l'autre, de plusieurs travaux (programmes) préparés d'avance sous forme de paquets de cartes perforées.

Cependant ces systèmes avaient le grand inconvénient de laisser l'unité centrale (Processeur central ou CPU) inactif pendant les opérations d'entrée/ sortie tel que impression, lecture et écriture sur support magnétique. Aussi un JOB (programme) très long laisse attendre pendant des heures d'autres jobs plus petits et qui ne consomment que quelques minutes d'exécution.

Le premier problème a été réglé par la venue d'une nouvelle génération d'ordinateurs (évolution technologique) qui permettait l'exécution en parallèle des Entées sorties (E/S) et du CPU «multiprogrammation» (équipés de plusieurs unités d'E/S autonomes ou contrôleurs d'E/S).

Les systèmes d'exploitation tels qu'UNIX ont introduit dès les années 70 la notion de temps partagé où le CPU exécutent presque en parallèle plusieurs programmes en même temps. Ainsi un programme ne doit pas attendre son exécution jusqu'à la terminaison du précédent. La notion d'interactivité (travaux non préparés d'avance mais l'utilisateur travaille directement à partir de son terminal) et le multi utilisateur (plusieurs utilisateurs utilisent en même temps la machine) s'est vu ensuite développée et chaque utilisateur avait l'équivalent d'une machine individuelle.

Les années 80, et grâce à l'évolution très rapide de l'intégration des circuits électroniques, a vu l'apparition des micros ordinateurs personnels. Ces petites machines ont révolutionné l'informatique dans le monde et l'ont rendue accessible à bon prix.

Des systèmes d'exploitation peu compliqués et très efficaces sont apparus tel que CP/M , MS-DOS, OS2, etc. ... . L'interactivité entre l'homme et la machine est alors devenue un objectif puis une exigence avec l'apparition de systèmes d'exploitation à interfaces graphiques tel que WINDOWS, Macintosh OS, XWINDOW, OPENWINDOW, ...

3- Les différents types de systèmes d'exploitation :

a- Systèmes de contrôle de processus :

Cette famille de systèmes comprend en général les systèmes de conduite de procédés industriels et les systèmes transactionnels tels que les systèmes de gestion de base de données.

La caractéristique principale de cette famille de systèmes est la prise en compte des contraintes temporelles de l'environnement.

b- Systèmes à vocation développement :

Elle est essentiellement composée des systèmes de temps partagé interactifs. Cette famille de systèmes connaît ces dernières années un essor important avec la diffusion de systèmes comme VMS de chez Digital Corporation et UNIX d'AT&T. Deux types de services sont supportés par cette famille de systèmes :

- Gérer des volumes et fichiers ;
- Développer et exécuter des programmes.

c- Systèmes à vocation exploitation :

Ces systèmes, héritiers des systèmes de traitement par lots des années 60, sont utilisés sur les grandes configurations coûteuses.

Les « mains frame » IBM, Bull, Control data sont opérationnels sous de tels systèmes. Le but recherché par ce type de systèmes est l'optimisation de l'utilisation des ressources coûteuses du matériel.

Exemple :

Il existe dans le monde de super ordinateur (dont la possession est réservée aux puissants) tel que la famille des CRAY caractérisés par une grande mémoire et des milliers d'unités de traitement. Ces ordinateurs sont utilisés dans les applications qui demandent beaucoup de calculs tels que la simulation des phénomènes naturels ou des programmes de recherches très poussés. Des systèmes d'exploitation de cette famille permettent aux utilisateurs de bénéficier de la puissance d'une telle machine.

4- Les modes d'exploitation :

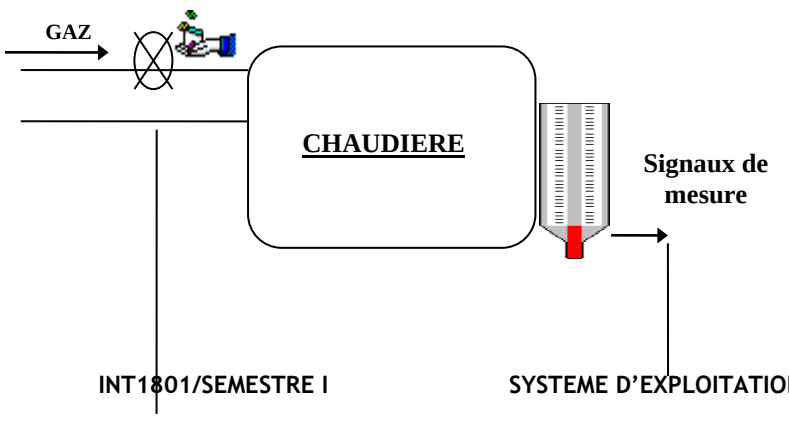
a- Exploitation en mode réel :

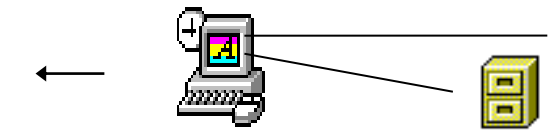
Ce mode d'exploitation a vu son apparition avec l'utilisation des calculateurs et appareils de mesure en industrie. Il s'applique lorsqu'une machine industrielle est commandée par des signaux venant d'autres appareils. Ces signaux sont traités par le calculateur (ordinateur) et suivant le résultat une commande est envoyée vers la machine industrielle. Tout ceci doit se faire très rapidement avant l'arrivée d'autres signaux. On établit ainsi une étroite relation entre le temps et la valeur de l'information (commande). Les fonctions principales d'un système supportant ce type d'exploitation peuvent se résumer comme suit :

- **A**ction sur des organes externes ;
- **R**éaction aux événements externes (signaux et valeurs d'entrée) ;
- **G**estion d'information (historique des événements et des actions) ;
- **P**rise en compte du temps.

Exemple :

Dans une usine nous avons une chaudière qui fonctionne avec du gaz naturel. Le système de régulation de sa température est contrôlé par un ordinateur qui doit stabiliser la température à 250 °C. Il doit ainsi connaître la température de la chaudière à tout moment à l'aide de capteurs de température et suivant la température lue actionner les vannes du gaz pour les ouvrir d'avantage et lever ainsi la température ou les fermer et diminuer ainsi la température. Le schéma de ce dispositif simple est le suivant :





Signaux de commande

Enregistrement

b- Exploitation en mono programme :

Dans sa forme d'expression la plus simple, cette configuration permet à un seul utilisateur de travailler en même temps sur l'ordinateur et d'exploiter ainsi les ressources de ce dernier. Généralement on la retrouve dans la micro-informatique (ex MS-DOS). La configuration minimale suffisante est:

- Une unité centrale ;
- Une mémoire centrale ;
- Un terminal de dialogue (clavier + écran).

Comme principales qualités de ce mode nous citons:

- La simplicité d'utilisation (système réduit à un seul utilisateur moins de commandes) ;
- Les performances (système de moyenne dimension) ;
- Le faible coût ;
- La fiabilité (pas de problèmes de partage, de synchronisation, de protection, ...)

c- Exploitation en multiprogramme :

- Multiprogrammation classique :

Comme nous l'avons dit dans l'historique, la multiprogrammation est caractérisée par la simultanéité entre exécution du CPU et le déroulement des Entrées/Sorties. De nos jours les micros ordinateurs (PC/AT par exemple) ne sont pas dotés de ce mode car ils sont très simples et peu coûteux. D'ailleurs on peut le constater lors de l'impression d'un document sur PC, le micro devient très lent (ou parfois bloqué jusqu'à la fin de l'impression).

Les gros systèmes par contre comme HP 3000 ou VAX par exemple ont des unités de traitement des E/S ou processeur d'E/S.

Ce mode s'appuie essentiellement sur des utilitaires comme le spooler d'impression par exemple qui gère les files d'attente derrière une imprimante et qui synchronise le travail du processeur d'E/S.

- Temps partagé (time sharing):

Ce mode est également connu sous le nom de TIME SHARING. Les principaux objectifs de ce mode sont:

- Satisfaire plusieurs utilisateurs
- Obtenir un temps de réponse court à l'échelle humaine
- Pouvoir accéder librement à la machine, c'est à dire programmer directement et mettre au point ces programmes.

Les programmes utilisateurs sont exécutés à tour de rôle par le CPU pendant un petit intervalle de temps et d'une façon circulaire.

d- Exploitation à travers des environnements graphiques (Orienté Objet) :

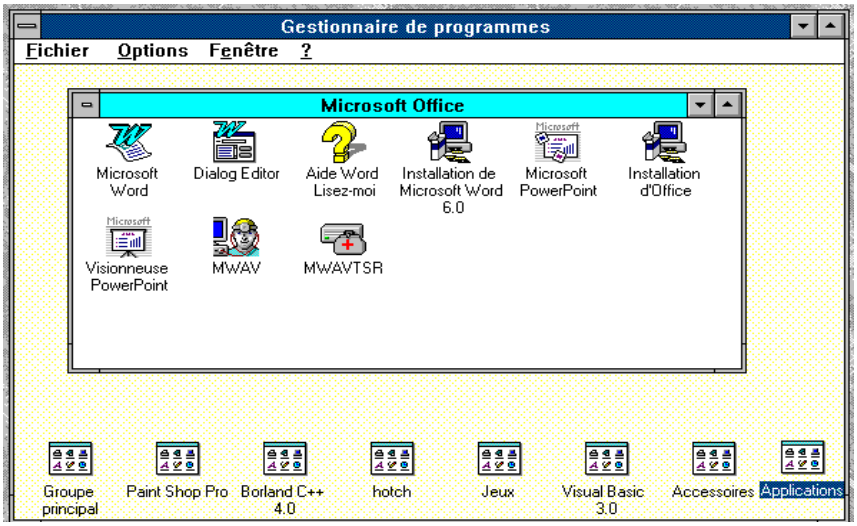
Dans tous les modes qu'on a vus précédemment l'utilisateur communique (donne des ordres) avec l'ordinateur à travers des commandes tapées au clavier. Ce mode de communication n'est ni convivial ni souple car il présente des:

- **D**ifficultés à un non informaticien de connaître toutes les commandes et ce qu'elles font pour exploiter au maximum la machine;
- Difficultés de retenir la syntaxe de chaque commande ;
- Difficultés de combiner les commandes entre elles et d'utiliser le langage de commande offert par le système d'exploitation;
- Le clavier bien qu'il soit l'interface N°1 entre l'utilisateur et l'ordinateur reste difficile à maîtriser, lent et peu convivial;

Pour toutes ces raisons, les concepteurs de systèmes d'exploitation se sont penchés sur d'autres outils de communication et ont fini par développer des environnements graphiques tels que WINDOWS qui facilitent énormément la tâche aux utilisateurs.

Cette nouvelle génération de systèmes d'exploitation est caractérisée par:

- La souris a pris le dessus sur le clavier pour demander des services à la machine;
- Les services de la machine sont présentés sous forme de menus et d'objets graphiques très faciles à repérer et à utiliser. L'aspect visuel est devenu d'un grand intérêt;
- De nouvelles techniques de présentation ont été développées tel que le multi-fenêtrage, les boites de dialogues, OLE, ...
- La programmation orientée objet a vu le jour et facilite ainsi au programmeur de mettre en place ses applications et les interfaces à ses programmes.



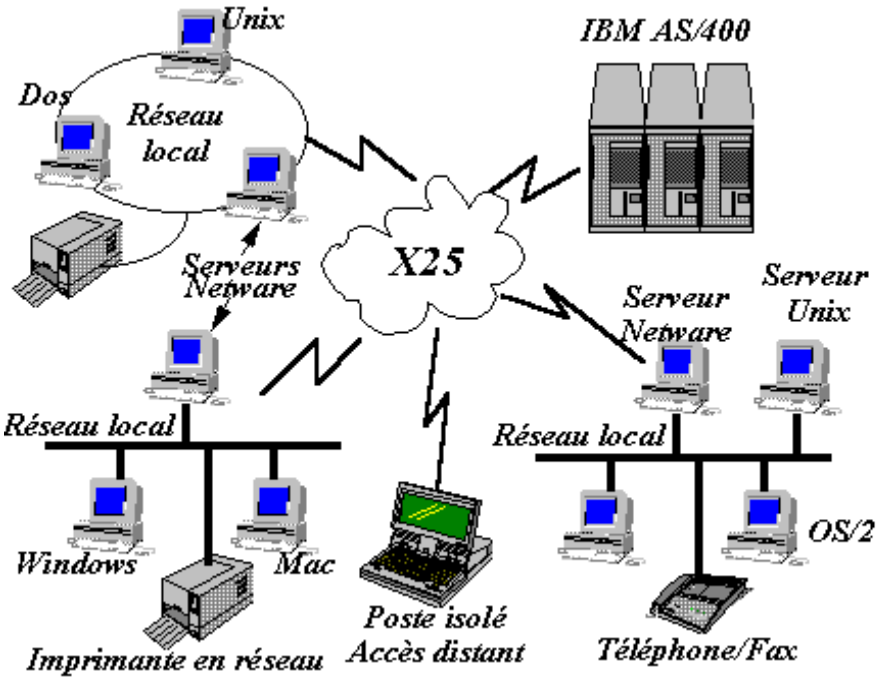
L'environnement de programmation WINDOWS de Microsoft

e- Exploitation en télétraitement (serveur/client) :

Dans ce mode d'exploitation les utilisateurs travaillent chacun sur son ordinateur (SITE) mais peuvent à travers des lignes de communication (lignes téléphoniques, câbles pour réseaux, ...) communiquer entre eux et utiliser les ressources (fichiers, données, imprimantes, ...) à distance.

Généralement on parle de site Principal ou serveur (l'ordinateur principal) et de sites secondaires (clients) qui sont les utilisateurs. Ces utilisateurs peuvent utiliser deux types d'équipements :

1- Des terminaux légers (clavier/Ecran).
2- Des ordinateurs possédant leur propre unité de calcul (CPU)



Le rôle du système d'exploitation dans ce cas est:

- Gérer les ressources sur l'ensemble des sites ;
- Gérer la communication et le transfert des informations à travers les câbles de communication ;
- Gérer les pannes et les erreurs très fréquentes dans ce mode.

Exemple :

Le système de réservation de la Compagnie Aérienne Air Algérie se fait à travers des terminaux (Clavier + Ecran) installés au niveau des agences et raccordés par un réseau national de transmission de données au Centre de calcul d'Air Algérie doté d'un ordinateur central de type IBM.

II- CONCEPTS ET FONCTIONS D'UN SYSTÈME D'EXPLOITATION :

1- Propriétés communes des systèmes d'exploitation :

Malgré la diversité des systèmes d'exploitation et des fins auxquels ils sont utilisés, on distingue certaines propriétés communes :

- Un environnement efficace de développement, de mise au point et d'exécution des programmes.
- Une disponibilité d'un large choix des moyens et d'outils de résolution des problèmes liés à toutes les activités entreprises par l'homme.
- Une efficacité caractérisée par une simultanéité des traitements et un partage transparent à l'utilisateur des ressources de la machine.
- Une interface Homme/machine souple et conviviale.
- Une documentation détaillée et facile à lire.

2- Fonctions d'un système d'exploitation :

Le système d'exploitation offre un nombre important de services à travers des fonctions qu'il offre aux utilisateurs. Pour accéder à ces fonctions deux moyens sont possibles:

a- L'interpréteur de commandes : qui est le programme qui lit les commandes tapées au clavier et qui les transforme en requêtes et appels aux fonctions du système d'exploitation. Ce moyen est utilisé par un simple utilisateur pour demander des services Assez simples

b- Le System Call : C'est une bibliothèque de fonctions que le programmeur peut appeler à partir de son programme pour demander des services plus complexes.

Nous pouvons diviser ces fonctions en trois grandes classes:

- **Gestion du matériel** : ces fonctions s'occupent de la gestion des Entrées/Sorties (Impression, écriture sur Ecran ,lecture du disque,....), la gestion de l'unité centrale (Partage du CPU, gestion des Interruptions, ...) et enfin la gestion de la mémoire physique (RAM, ...) ;

- **Gestion de l'information**: ces fonctions s'occupent de la gestion du système de fichiers, des tampons (Buffers), et de la protection dans le système ;

- **Gestion des utilisateurs***:* ces fonctions assurent la gestion de la connexion et la déconnexion des utilisateurs (de l'ordinateur) et d'autres services de l'administration (ajout d'utilisateurs, les mots de passe, etc.).

RÉSUMÉ :

Ce que nous avons appris dans cette leçon est qu'un ordinateur n'est pas seulement un ensemble de cartes et de composants électroniques mais que « l'âme » qui le fait fonctionner est un ensemble de programmes de base appelé Système d'exploitation.

Plusieurs types de systèmes existent, mais la majorité appartient à l'une des trois classes suivantes:

1 - Systèmes de contrôle de processus ;

2 - Systèmes à vocation développement ;

3 - Systèmes à vocation exploitation.

Les systèmes que nous rencontrons le plus souvent dans le domaine du travail et qui seront étudiés dans la suite de ce cours sont les systèmes à vocation Développement. Enfin les

fonctions les plus importantes d'un ordinateur sont: la gestion du matériel, de l'information et des utilisateurs.

Dans la prochaine leçon, nous verrons plus en détail les fonctions qu'offre le système d'exploitation.

EXERCICES D'APPLICATION :

Pour mieux assimiler cette leçon, essayons ensemble de répondre aux questions suivantes, pour cela nous vous conseillons de vous faire aider par des livres traitants ces sujets et aussi par des informaticiens qui ont déjà travaillé sur des systèmes réels. Comparer vos réponses avec celles que nous vous proposons.

EXERCICE N° 01:

Le mode batch (moniteur d'enchaînement) n'a pas été détaillé dans le cours. À partir de votre bibliographie, décrivez-le en donnant un exemple.

EXERCICE N° 02:

Quelle est la différence entre la multi programmation et le mode en temps partage ?

EXERCICE N° 03:

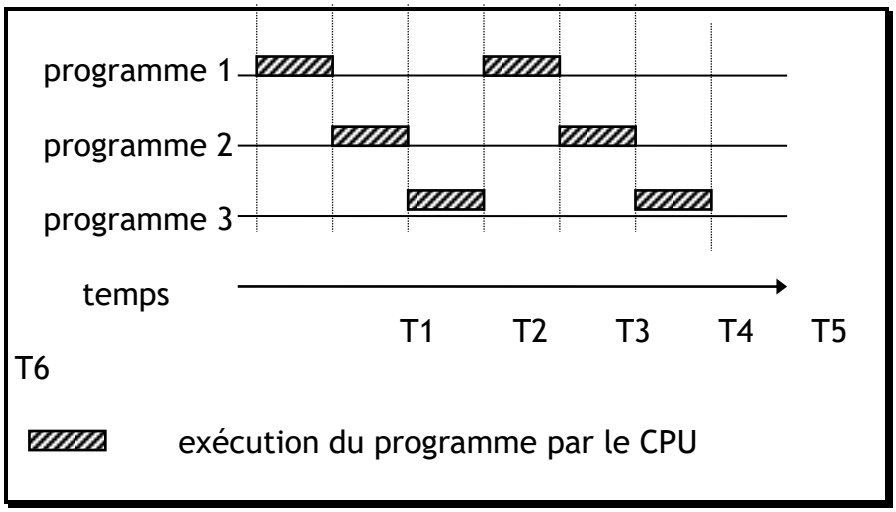
Définir une ressource et donnez la différence entre une ressource physique et une ressource logique. Citez quelques ressources physiques d'un ordinateur.

EXERCICE N° 04 :

Dans un système en temps partagé plusieurs utilisateurs travaillent en même temps sur l'ordinateur. Quels sont les types de problèmes qui peuvent surgir et par qui ils doivent être réglés ?

EXERCICE N° 05

En se basant sur votre bibliographie, citez tous les services de gestion des utilisateurs (comptes) que doit entreprendre le système d'exploitation. Déduire le rôle de l'opérateur ou système manager.



CORRIGÉ DES EXERCICES :

RÉPONSE N° 01:

Les premiers moniteurs d'enchaînement ou systèmes Batch sont les premiers systèmes d'exploitation venus améliorer les machines dites « portes ouvertes ». Et permettent ainsi d'automatiser l'exécution des programmes ou jobs (sans intervention de l'homme). Les opérations de lecture, compilation, chargement et exécution des jobs se fait sans l'intervention de l'opérateur et les programmes ou jobs passent l'un à la suite de l'autre dans l'ordinateur.

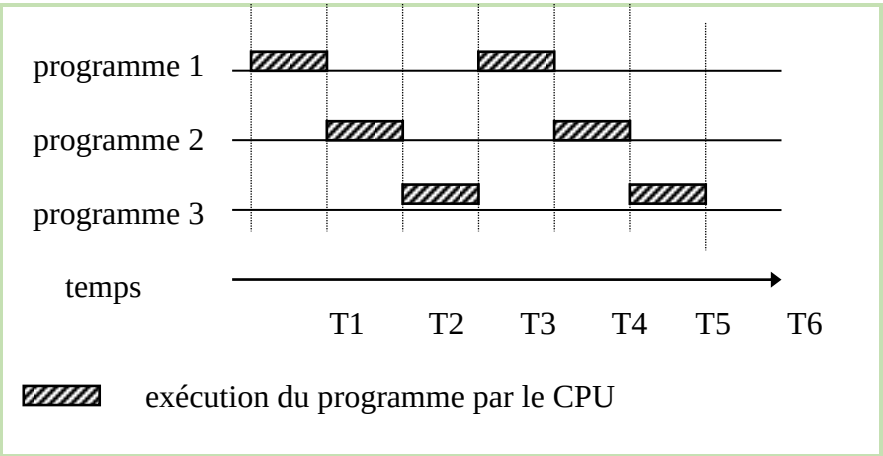
Un exemple de ces systèmes est les ordinateurs avec cartes perforées. Dans ces ordinateurs les programmes étaient codés sur des cartes (chaque instruction sur une carte). Chaque Job avait des cartes de contrôle en plus des cartes contenant le programme et les données.

RÉPONSE N° 02:

La multi programmation vise à mieux rentabiliser le CPU (unité centrale) et l'occuper le plus longtemps possible. Ainsi dès qu'un job est mis en attente d'une opération d'E/S ou pour une autre raison, le CPU est automatiquement alloué à un autre job. Le CPU est rendu au premier job dès que celui-ci sort de son état d'attente.

Le temps partagé par contre consiste à partager le CPU d'une manière équitable entre les jobs. Les programmes sont organisés en file d'attente circulaire et à passent à tour de rôle exécuter sur le CPU pour une période de temps T définit par le système d'exploitation. Soit trois programmes qui s'exécutent en temps partagé. Voici le schéma temporel du déroulement de leur exécution:

RÉPONSE N°03:



Une ressource est tout moyen mis à la disposition des utilisateurs pour leurs travaux. En général nous avons deux types de ressources:

***** Ressources matérielles: ce sont les constituants physiques de la machine comme le CPU, la mémoire, les périphériques d'E/S tel que l'imprimante, le disque, l'écran, le clavier, etc ...

* Ressources logiques: ce sont les moyens logiciels et les données mis à notre disposition par le système d'exploitation. nous pouvons citer deux classes:

- Données: tel que les fichiers, les bases de données, les banques de données, etc ...

- Programmes: Logiciels, compilateurs, Bibliothèques fonctions, fonctions du système d'exploitation ...

etc ...

RÉPONSE N° 04 :

Les types de problèmes liés au travail à temps partagé sont nombreux, nous pouvons citer:

- Accès simultané aux ressources physiques et logiques. Exemple deux programmes qui mettent à jour en même temps un même fichier.
- Gestion de plusieurs programmes résidents en même temps en mémoire centrale.
- Protection des données: un programme ne peut accéder qu'à ces données et le système d'exploitation doit lui interdire d'accéder à d'autres zones de données. En un mot il doit lui délimiter son espace de travail.

Notons que c'est le système d'exploitation qui doit régler ces problèmes.

RÉPONSE N° 05:

Les services de gestion des utilisateurs peuvent se résumer en quatre classes:

1- Gestion des comptes utilisateurs :

- Création et mise à jour des comptes utilisateurs
- Gestion des mots de passes

- Gestion des privilèges et droits d'accès des groupes d'utilisateurs
- Initialisation des fichiers scripts de login de chaque utilisateur.
etc ...

1- Gestion des ressources physiques :

- Gestion des volumes: monter et démonter les volumes
- Gestion de l'impression différée
- définition des priorités des files d'attentes du mode partagé
- Ajout et connexion de nouveaux éléments à l'ordinateur
  etc ...

2- Gestion des pannes et sauvegardes :

- Effectuer des sauvegardes journalières et des backups
- Récupération de données en cas de crash du système
- Redémarrer le système et l'arrêter
- Sauvegarde en cas de panne de courant
  etc ...

3- Gestion de la comptabilité de l'utilisation de l'ordinateur :

- Comptabiliser le temps de connexion des utilisateurs
- Comptabiliser le temps machine (exécution CPU) des programmes
- Comptabiliser les impressions et le listing
- Optimisation de la charge du système

Toutes ces tâches sont prévues dans un système d'exploitation. L'opérateur système doit donc exécuter les programmes système qui s'occupent de ces tâches et superviser le fonctionnement de ces programmes.

LEÇON N°02 : MÉCANISMES DE BASE D'EXÉCUTION DES PROGRAMMES

OBJECTIF PÉDAGOGIQUE

À la fin de cette série, le stagiaire doit être capable d'appliquer la gestion du matériel dans un système d'exploitation.

PLAN DE LA LEÇON :

INTRODUCTION

- **I- GESTION DU PROCESSEUR (CPU)**
- **II- GESTION DES INTERRUPTIONS**
- **III- GESTION DES ENTRÉES/SORTIES**
- **IV- GESTION DE LA MÉMOIRE**

Résumé

EXERCICES D'APPLICATION

CORRECTION DES EXERCICES

LEÇON N°02 : MÉCANISMES DE BASE D'EXÉCUTION DES PROGRAMMES

INTRODUCTION :

Les ordinateurs modernes se composent de processeurs, de mémoires, d'horloges, de disques, de terminaux, et d'une multitude d'autres périphériques. Dans cette optique, le travail du système d'exploitation consiste à ordonner et à contrôler l'allocation des processeurs, des mémoires et des différents périphériques d'Entrée/Sortie entre les programmes utilisateurs.

Dans cette leçon, nous allons aborder le rôle du système d'exploitation comme gestionnaire du matériel, tout en évoquant les types de difficultés qu'il est amené à résoudre.

I- LA GESTION DU PROCESSEUR (CPU) :

L'un des rôles les plus importants du système d'exploitation, est la gestion de l'unité centrale (CPU). Comme vous le savez, le CPU est le cerveau de l'ordinateur, et donc il est très important pour nous qu'il soit bien entretenu par le système.

Les programmes du système d'exploitation doivent manipuler deux éléments essentiels pour assurer la gestion du CPU, qui sont :

- **Le registre d'état :**

  Le système d'exploitation doit scruter ce registre pour tirer le maximum d'informations sur l'exécution d'une instruction (erreur, arrivée d'interruption, etc.)

- **Le contexte du processeur PSW :**

  Ce sont tous les registres utilisés par le CPU pendant l'exécution d'un programme et permettant de l'identifier.

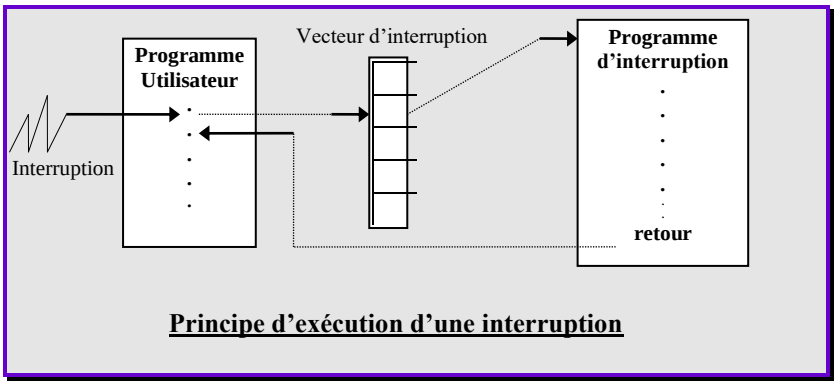
  Dans le cas des systèmes multitâches, les programmes s'exécutent à tour de rôle sur le CPU. Le contexte du CPU de chaque programme (tous les registres identifiant ce programme ou **PSW**) est sauvegardé en mémoire et suit le programme pendant sa « vie » (exécution).

II- GESTION DES INTERRUPTIONS :

1- Notion d'interruption :

Une interruption est un signal (électronique) qui permet à l'unité centrale de changer d'état. Ce signal apparaît quand certains événements extérieurs au programme se produisent et nécessitent une réaction plus ou moins immédiate du processeur. On réalise ainsi une commutation du contexte du processeur (modification de contexte).

Du point de vue hardware, une interruption est un arrêt technologique qui va provoquer un changement d'état de l'unité centrale et par suite, la suspension du programme en cours d'exécution. Une interruption permet donc de forcer un processeur à suspendre l'exécution d'un programme en cours, et à exécuter un programme prédéfini.



Principe d'exécution d'une interruption

Le programme prédéfini est appelé « programme d'interruption », son adresse de début se trouve en mémoire dans une des entrées du vecteur d'interruptions.

Le programme d'interruption s'exécute dans un contexte distinct de celui du programme interrompu, notamment en ce qui concerne le mode d'exécution, la protection, les informations accessibles, ... etc.

La gestion des différentes interruptions se fait à l'aide d'une logique câblée : « le système d'interruption ». Sur les PC, le contrôleur d'interruptions PIC représente le système d'interruptions. **(Voir cours d'architecture des ordinateurs, structure machine « Série 01 »)**.

Remarque :

A chaque interruption correspond un programme d'interruption bien précis qui s'exécute jusqu'à sa fin puis rend le contrôle du CPU au programme interrompu.

2- Les classes d'interruptions :

Les interruptions sont classées en deux catégories :

- Les interruptions logicielles
- Les interruptions matérielles

- **Les interruptions logicielles :**

On parle d'interruption logicielle lorsqu'une interruption est déclenchée par une instruction du microprocesseur et à partir d'un programme.

Exemple :

Dans le cas des microprocesseurs d'INTEL 8086,...,80486 l'instruction assembleur qui effectue une interruption logicielle est « **INT**».

L'instruction **INT 21** : Veut dire au processeur d'aller exécuter le programme d'interruption dont l'adresse de début est stockée au 21eme élément du vecteur d'interruption.

- **Les interruptions matérielles :**

Ces interruptions ne sont pas déclenchées par un programme mais plutôt par des composants électroniques (clavier, imprimante, ...), ce type d'interruptions constitue un mécanisme simple et très efficace pour faire réagir le processeur à des événements déterminés.

Les interruptions matérielles se divisent en deux groupes :

1- Les exceptions ou trappes (internes au CPU) ;
2- Les interruptions matérielles provoquées par des événements extérieurs au CPU.

a- Les exceptions :

Quand un indicateur de changement d'état (bit dans le registre d'état) est modifié par une cause liée à l'exécution d'une instruction en cours, on dit qu'il y'a déroutement ou exception.

Le mécanisme de déroutement a essentiellement pour rôle de traiter une anomalie qui survient dans le déroulement d'une instruction :

Données incorrectes (débordement arithmétique, division par zéro, ...).

- Tentative d'exécution d'une opération interdite par un dispositif de protection (violation de la mémoire, l'exécution d'une instruction privilégiée par un programme non autorisé à le faire).

Chaque numéro de l'exception correspond à une entrée dans le vecteur d'interruptions. Dans cette entrée nous trouvons l'adresse en mémoire du programme d'interruption qui est activé chaque fois que l'exception est déclenchée.

Voici sous forme de tableau quelques numéros d'exceptions du 80486.

N° du vecteur	Fonction	Type
0	erreur de division (division par zéro)	exception
1	exécution pas à pas	interruption
2	interruption non masquable NMI	interruption
3	point d'arrêt	interruption
4	dépassement	exception
5	index invalide	exception
6	code opération invalide	exception
7	coprocesseur non disponible	exception
8	double exception	exception
9		
10	segment d'état de tâche non valide	exception
11	segment non présent	exception
12	débordement de segment de pile	exception

Exceptions et interruptions réservées du 80486

b- Les interruptions matérielles provoquées par des événements extérieurs :

Ces interruptions sont provoquées par des événements extérieurs aléatoires qui requièrent l'attention immédiate du microprocesseur. Celui-ci dispose de deux entrées de demandes d'interruptions qui sont :

- **Interruptions non masquables (NMI) :**

Ce sont les interruptions dont le déclenchement implique l'exécution directe du programme d'interruption correspondant. Donc elles ne peuvent pas être retardées ou masquées par le système d'exploitation.

- **Les interruptions masquables:**

On parle ici, des interruptions qu'on peut masquer, c'est à dire retarder. Ces interruptions peuvent être par la suite démasquées et ainsi le mécanisme d'interruption est réactivé. Sachant qu'on dispose d'une seule entrée pour les interruptions masquables, et afin de multiplier les possibilités du circuit, on fait intervenir un contrôleur d'interruptions. **(Voir Leçon 01 « Structure machine » architecture des ordinateurs)**.

Remarque :

Etant donné le caractère synchrone du déroutement (associé à des instructions en cours) la notion de masquage n'a pas de sens dans ce type d'interruptions. Car comme on peut le deviner, dès que l'anomalie se produit il faut faire rapidement un traitement adéquat. Et donc les déroutements ne peuvent pas être retardés.

Exemple :

Sous MSDOS, Pendant l'impression d'un document, les caractères tapés au clavier ne sont pas pris en compte avant la fin de l'impression. Que s'est-il passé au juste ? En effet, le système **MS-DOS** masque les interruptions qui proviennent du clavier **(interruption numéro 9)** et donc le microprocesseur ignore que des caractères sont tapés au clavier. Dès que l'impression est achevée, MS-DOS démasque les interruptions provenant du clavier, et l'utilisateur voit sur écran les caractères qu'il tape sur le clavier.

3- : Le mécanisme d'interruption

▪ Mécanisme de changement d'état :

Le mécanisme de changement d'état permet en une seule opération indivisible de :

- Ranger dans un emplacement spécifié de la mémoire le contenu courant du mot d'état du processeur (PSW). C'est à dire sauvegarder le contexte du programme interrompu ;

- Charger dans le mot d'état du CPU un nouveau contenu (contexte du programme d'interruption) préparé à l'avance dans un emplacement spécifié de la mémoire.

Ce changement d'état n'est possible que si le processus en cours (ou le programme) se trouve dans un état bien défini, c'est à dire un point observable ou point interruptible.

Le changement d'état est déclenché suivant l'état d'un indicateur (bit) consulté par le processeur lors de l'exécution de chaque instruction. Donc, on ne peut pas interrompre un processus dans des points non observables ou non interruptibles.

L'opération de changement d'état d'un processeur est schématisée par la figure suivante :

4- Niveau de priorité :

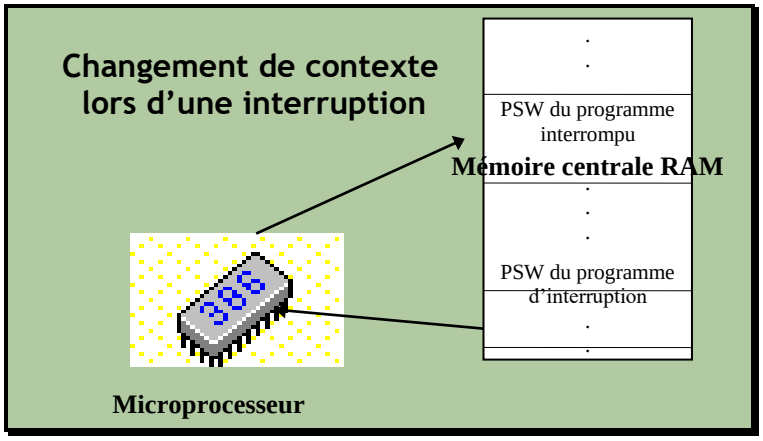
Un processeur peut être interrompu par diverses causes. Le mécanisme d'interruptions doit pouvoir distinguer ces causes afin de les traiter.

Comme nous l'avons vu, il existe des interruptions masquables qui peuvent être retardées et d'autres qui ne le peuvent pas. Cependant pour assurer leur gestion, on a associé à chaque interruption un degré d'importance, ce qu'on appelle :

Niveau de priorité. Ainsi, si deux interruptions arrivent en même temps, la plus prioritaire passe la première. Pour réaliser ce principe de priorité, deux schémas sont employés :

- Un indicateur distinct est associé à chaque cause d'interruption (on parle souvent de niveau d'interruptions), à chaque niveau est associé un couple distinct d'emplacements pour la sauvegarde et le chargement du mot d'état, en d'autres termes à chaque niveau correspond un programme de traitement distinct, automatiquement activé par le mécanisme d'interruption.

  - Un indicateur unique est utilisé pour toutes les interruptions avec un couple unique



d'emplacements pour la sauvegarde et le changement du mot d'état. Une information supplémentaire (code d'interruption) contenu dans le mot d'état ou dans un emplacement conventionnel en mémoire permet de distinguer entre les causes possibles des interruptions. La première tâche du programme est de consulter ce code pour déterminer l'origine de l'interruption et appeler la routine appropriée.

Remarque :

En pratique, on trouvera souvent une combinaison de ces deux mécanismes

III- LA GESTION DES ENTRÉES /SORTIES :

Le contrôle des périphériques d'Entrées/Sorties de l'ordinateur est une des fonctions primordiales d'un système d'exploitation. Le système d'exploitation doit envoyer les commandes aux périphériques, intercepter les interruptions provenant des périphériques d'E/S, récupérer les données lues et traiter les erreurs.

Les principes du point de vue matériel d'une E/S sont présentés dans les modules structure des ordinateurs et réseaux. Dans ce paragraphe, nous verrons l'Entrée/Sortie du point de vue software c'est à dire comme partie intégrante du système d'exploitation.

Avant tout, donnons la définition d'une opération d'Entrée Sortie :

On désigne sous le terme général d'entrée/Sortie (**E/S** ou **I/O**) tout transfert d'informations depuis ou vers l'ensemble mémoire adressable. Les E/S se divisent en deux types :

- Les E/S de type caractère ;
- Les E/S de type bloc

Le rôle primordial des programmes du système d'exploitation s'occupant de la gestion des E/S est d'acheminer les données entre les programmes des utilisateurs et les périphériques.

En partant du principe que seul le système d'exploitation connaît les spécificités de la machine et que lui seul peut savoir où se trouvent les programmes des utilisateurs en mémoire, nous pouvons déduire que le système d'exploitation est le plus habilité à faire ce transfert entre programmes utilisateurs et périphériques d'Entrée/Sortie.

Ainsi, pour l'utilisateur chaque port d'E/S ou périphérique a un nom logique tel que LPT1 : pour l'imprimante par exemple. Le système doit faire le lien entre ce nom logique et l'adresse physique (espace d'adressage I/O) d'une façon transparente à l'utilisateur.

De la même façon, les zones de données de l'utilisateur sont identifiées par des noms de variables. Le système doit faire le lien entre ces noms de variables et l'adresse physique en mémoire et pouvoir ainsi effectuer le transfert.

1- Les E/S de type caractère :

Les différentes fonctions du système d'exploitation pour l'entrée et la sortie de caractères permettent de commander des périphériques de type caractère tels que le clavier, l'écran, l'imprimante et l'interface série.

Deux modes de fonctionnement existent pour ce type :

➢ Lecture indirecte avec contrôle (COOKED) :

Elle consiste à contrôler les caractères transférés entre le programme utilisateur et les périphériques. Sur le clavier, les touches de contrôle tel que **CTRL-Z**, **CTRL-C** sont alors reconnues et une routine correspondante est déclenchée.

Exemple 1 :

Quand ce mode est actif et qu'on tape **CTRL-C** sur le clavier, le programme en cours d'exécution est arrêté et l'exécution est retournée au **DOS** (système d'exploitation).

Exemple 2 :

L'instruction READ en pascal permet de lire des données. Ceux-ci ne sont transférés au programme utilisateur (pascal) que si on tape la touche ''entrée''. Dès que le système d'exploitation reconnaît la touche ''entrée'' il prend le contenu du buffer (tampon du clavier) et le transfert vers la zone de données du programme utilisateur.

▪ Mode direct (RAW) :

Dans ce mode dès qu'une touche est tapée elle est immédiatement transférée vers le programme qui en a besoin sans vérifier sa valeur, ni attendre une validation par la touche ''Entrée''.

2- Les E/S de type bloc :

Les périphériques de type bloc mémorisent les données dans des blocs de taille fixe, chaque bloc ayant une adresse propre. Les tailles courantes des blocs vont de 128 octets à 1024 octets. La propriété fondamentale de ces périphériques est qu'ils permettent de lire ou d'écrire un bloc indépendamment de tous les autres. En d'autres termes, un programme peut écrire ou lire n'importe quel bloc stocké dans ces périphériques. Les disques, CD-ROM, Bandes magnétiques,… etc. font partie de ce type de périphériques.

3- Les étapes de déroulement d'une E/S :

L'E/S est une opération très courante dans la vie des logiciels, elle met en jeu plusieurs acteurs en plus du programme utilisateur voulant lire ou écrire sur un périphérique d'E/S. Une opération d'E/S passe par plusieurs phases ou couches avant de s'achever. Les quatre principales couches sont :

- La couche de bas niveau ;
- La couche des pilotes de périphériques ;
- La couche des routines de gestion d'E/S ;
- La couche d'interface avec le programme utilisateur.

Cette Présentation en forme de couches a plusieurs avantages dont le plus important est l'indépendance (dissociation) entre les procédures qui touchent au matériel (qui sont très spécifiques à la marque de la machine) et les procédures de gestion logique des E/S (ne touchent pas au matériel). Ceci est schématisé par la figure suivante :

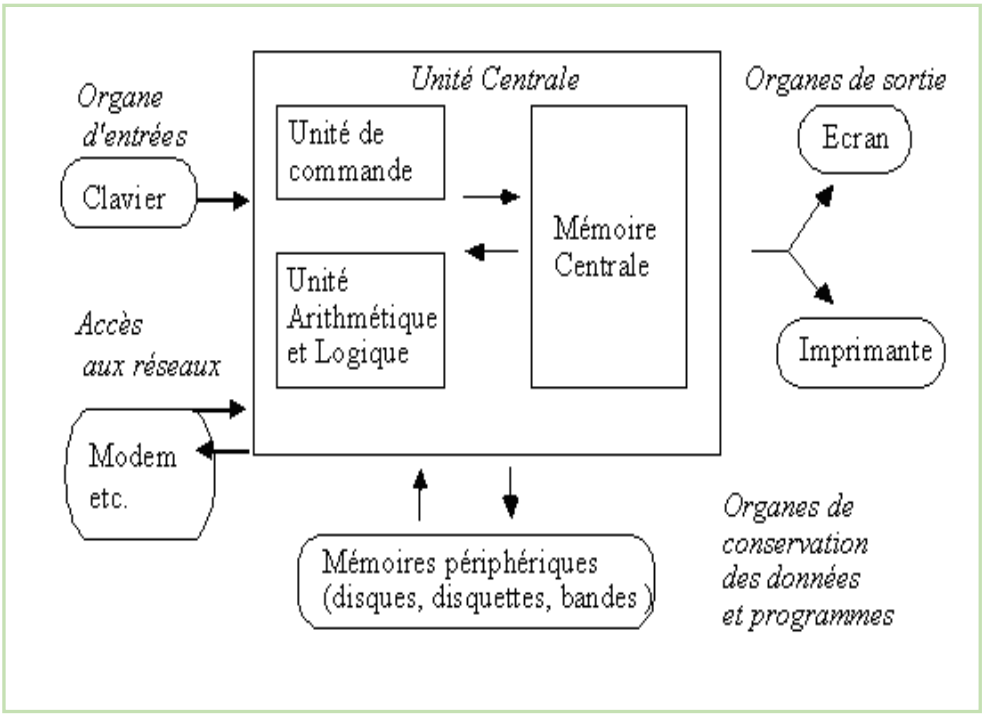


Schéma synoptique d'une entrée/sortie

- **Les routines de bas niveau :**

La couche de bas niveau est très proche du matériel et s'occupe d'effectuer physiquement le transfert et d'activer les circuits associés au périphérique (DMA et contrôleurs). Elle utilise le mécanisme d'interruptions pour communiquer avec les périphériques, et stocke les données devant être acheminées à partir ou vers le périphérique dans des tampons intermédiaires (Buffers d'Entrées sorties).

Exemple1 :

Sur les machines PC ces routines sont appelées BIOS (Basic **I**nput **O**utput System), elles sont écrites par le constructeur du micro-ordinateur, et sont appelées par MS-DOS et les autres logiciels.

Exemple 2:

Sur micro-ordinateur, la lecture à partir du clavier se fait d'une manière très simple :

Les étapes de déroulement de l'entrée de caractères se fait comme suit :

- Chaque fois qu'un caractère est tapé sur le clavier, une interruption matérielle N° 9 (INT 9) est déclenchée ;

- Elle réveille la routine de bas niveau (BIOS) qui se chargera de transférer le caractère tapé du tampon du clavier vers une zone en mémoire centrale ;

- Quand un programme veut lire à partir du clavier, il appelle le système MS-DOS à l'aide de l'interruption logicielle N°21h (l'instruction INT 21h), et exécute ainsi le programme de gestion de l'E/S ;

- MS-DOS de son côté appelle le pilote qui gère le clavier et qui s'appelle CON : et lui demande d'aller chercher un caractère ;

- Le pilote va dans la zone mémoire et cherche le caractère qui a été tapé à l'aide de la routine d'interruption logicielle N° 16h.

- **Les pilotes de périphériques :**

Les pilotes de périphériques sont des programmes du système d'exploitation qui doivent formuler la demande d'E/S sous forme de commandes suivant la nature de l'opération et le type du périphérique. Ils communiquent avec le matériel d'une façon directe ou à travers les routines de bas niveau.

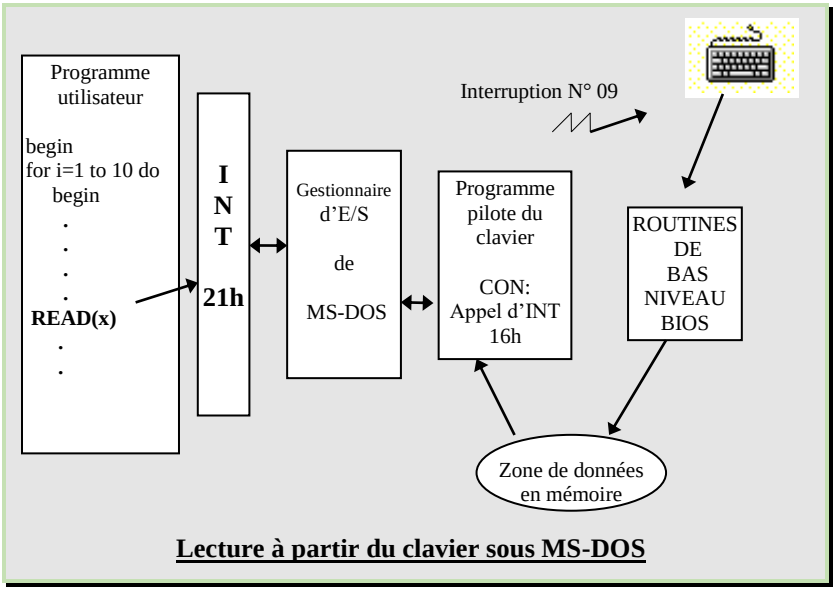
Dans le cas des systèmes multitâches, les pilotes ont pour rôle de gérer les files d'attentes des demandes d'E/S sur un périphérique donné. Ils gèrent également les buffers et tampons (mémoires de stockage intermédiaire entre le périphérique et le programme utilisateur).

Notons que plusieurs logiciels s'en passent du système d'exploitation pour effectuer leurs opérations d'E/S et donc accèdent directement aux composants et aux ports en utilisant leurs propres pilotes (DRIVERS).

- **Les routines de système de gestion d'E/S :**

C'est un ensemble de routines de haut niveau : Elles s'occupent de la gestion logique de l'opération d'E/S et de la prise en compte de la requête (demande) d'E/S du programme utilisateur. Parmi ses tâches élémentaires nous citons :

- Correspondance entre le nom logique du périphérique et de son adresse physique (port d'E/S) ;
- Vérification des droits d'accès et établissement des modes d'accès ;



Lecture à partir du clavier sous MS-DOS

- Allocation de mémoire intermédiaire (tampons) nécessaire à cet effet ;
- Acheminer les données lues à partir du périphérique dans la zone de données du programme utilisateur appelant ;
- En cas d'erreurs d'E/S prévenir le programme utilisateur et afficher un message.

- **L'interface de la mémoire :**

L'interface d'appels au système (System call) : Elle comprend les fonctions et programmes qui jouent le rôle d'interface entre le programme utilisateur (client) et le système d'exploitation (serveur) devant effectuer l'opération. Elles se présentent sous plusieurs formes :

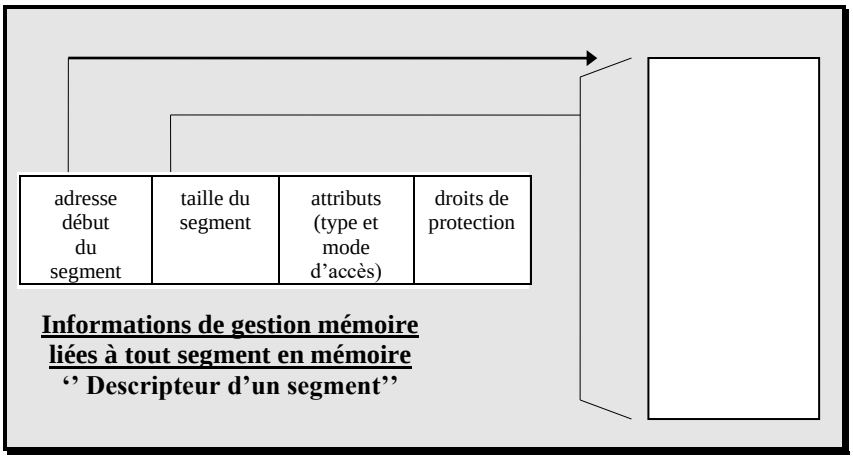
- Fonctions d'appel aux bibliothèques système, accessibles à travers les langages de programmation tel que PASCAL, FORTRAN, etc... .
- Interruptions logicielles réservées.

V- GESTION DE LA MÉMOIRE :

La mémoire est une ressource de stockage de données et de programmes. Elle est subdivisée en une suite finie de composants appelés emplacements. La mémoire est une ressource nécessaire à l'exécution d'un processus ou programme. Ce dernier ne peut s'exécuter sans être chargé (au moins partiellement) en mémoire centrale (RAM).

Objectifs :

Le gestionnaire de la mémoire est un module du système d'exploitation dont le rôle est la gestion de la mémoire principale. Il doit viser les objectifs suivants :



- **L'allocation :** Tout programme qui s'exécute en mémoire a besoin d'espace mémoire (RAM) pour stocker ses données et variables. Le gestionnaire de mémoire, est un ensemble de fonctions et procédures du système d'exploitation qui s'occupe de cette allocation.
- **La réallocation :** Un programme ne connaît pas à priori l'environnement de son exécution. Lorsqu'un programme terminé libère de l'espace mémoire, il est nécessaire de réorganiser les programmes en mémoire afin d'optimiser l'utilisation de cette ressource.
- **La protection :** La coexistence de plusieurs processus en mémoire centrale nécessite la protection de chaque espace mémoire vis à vis des autres. Ainsi, un processus P_1 ne peut accéder à l'espace d'un processus P_2 que si ce dernier l'autorise.
- **Le partage :** Parfois, il est utile de partager un espace mémoire entre plusieurs processus. Ainsi, le sous-système de gestion de la mémoire doit autoriser des accès contrôlés sans compromettre la protection.

Exemple : Un éditeur de texte peut être partagé entre plusieurs utilisateurs d'un système à temps partagé.

Il existe plusieurs stratégies d'allocation de la mémoire. Nous citons parmi elles la **PAGINATION** et la **SEGMENTATION**.

1- Segmentation de la mémoire :

Comme nous l'avons vu pour la gestion de la mémoire virtuelle, les programmes et les données doivent être divisés en parties pour faciliter leur manipulation.

La segmentation est une façon de diviser le programme et les données, suivant leur signification logique par exemple un programme peut être divisé de la façon suivante :

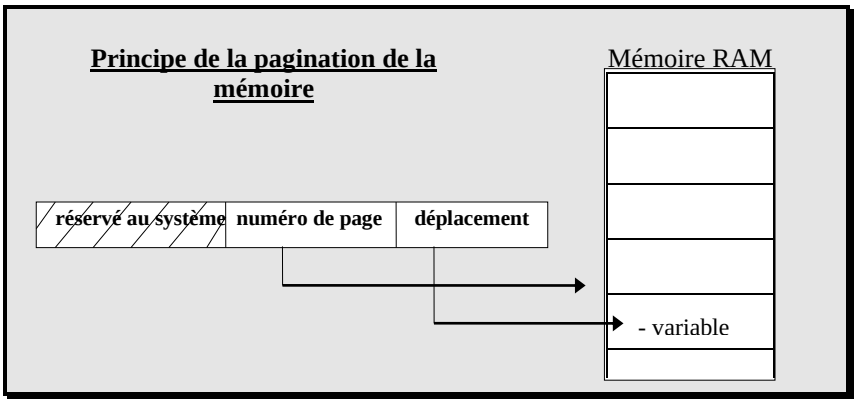
- Le premier segment comporte les données et les variables ;
- Le deuxième segment contient le programme principal ;
- Le troisième segment contient les sous programmes ;
- Le quatrième segment contient la pile.

Le système d'exploitation ne gère plus des cases mémoires mais des segments, il les mets en attente sur le disque quand on n'a pas besoin d'eux et les charge en mémoire dès qu'ils deviennent utiles pour l'exécution du programme.

L'autre avantage de l'adressage segmenté est que les adresses générées par les programmes ne voient pas la mémoire physique mais plutôt un ensemble de segments avec des déplacements à l'intérieur de chaque segment ce qui permet :

- Un espace d'adressage plus grand ;

- Une indépendance de l'emplacement des programmes en mémoire (adresses virtuelles).



2- Technique d'OVERLAY (recouvrement) :

Il y a de nombreuses années, les programmes étaient trop volumineux pour entrer dans la mémoire disponible (la taille de la RAM était beaucoup plus petite que la taille du programme). La solution générale adoptée fut de diviser le programme en plusieurs « morceaux » appelés OVERLAYS. Ainsi, pour exécuter un programme volumineux, il suffisait exécuter d'abord son premier overlay. Quand celui-ci est terminé, il est effacé de la mémoire et l'overlay suivant est chargé en mémoire (à partir du disque), à la même place qu'occupait le précédant overlay. Ce mode est effectué autant de fois qu'il y a d'overlays dans un programme.

L'inconvénient majeur de cette méthode de gestion de la mémoire se résume en deux points :

- La division d'un programme en overlays doit être faite par le programmeur lui-même.

- Ce qui représente un long et fastidieux travail, car le programmeur doit gérer lui-même son espace de travail en mémoire ;

- Si le programmeur n'est pas expérimenté, ses programmes risquent de devenir très lents, dans le cas où le programme effectue plusieurs va et vient entre le disque et la mémoire (chargement et déchargements d'overlays).

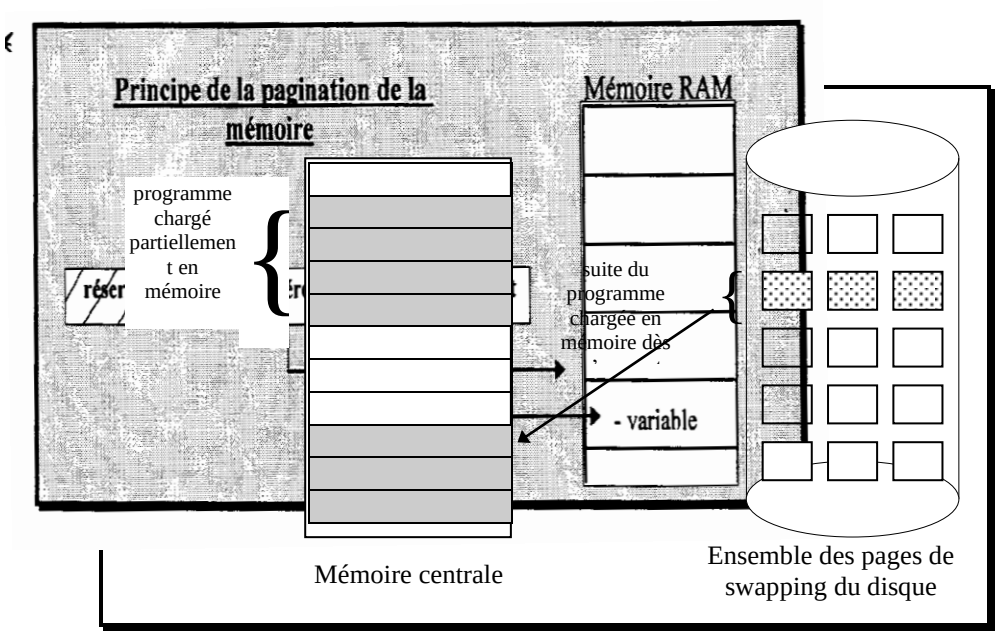
La finalité de ce type de fonctionnement est que l'utilisateur doit gérer lui-même sa mémoire centrale (RAM).

3- Mémoire paginée :

La notion de pagination de la mémoire est une technique qui est venue combler les insuffisances de la segmentation de la mémoire et la complémenter.

Le principe est très simple, il consiste à diviser la mémoire RAM en plusieurs blocs de même taille (par exemple de 4 K octets). Ainsi, un programme voulant accéder à une donnée en mémoire doit lui accéder à travers la page qui la contient et non pas directement. Pour cela, l'adresse est composée de deux informations essentielles qui sont :

- Le numéro de la page ;
- La position à l'intérieur de la page.

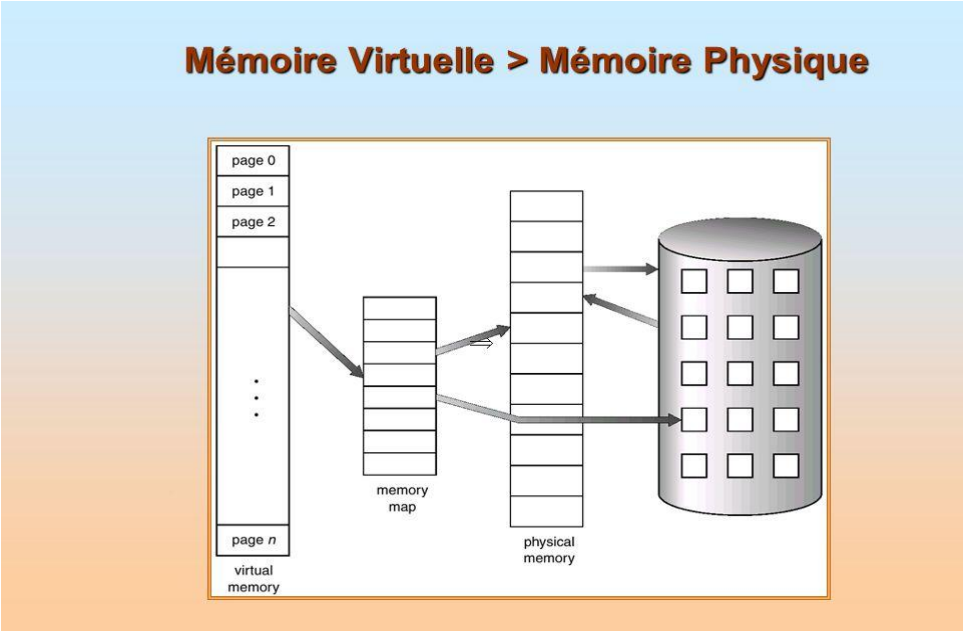


4- Mémoire virtuelle et SWAPPING :

Pour pallier aux inconvénients du système d'overlays, la notion de mémoire virtuelle a vu le jour. Le but est de décharger l'utilisateur de gérer la mémoire, et de réserver cette tâche de gestion des longs programmes au système d'exploitation.

L'idée de base de la mémoire virtuelle est que la taille du programme, des données et de la pile peut dépasser la mémoire disponible. Le système d'exploitation garde en mémoire les parties de programme qui sont utilisées et stocke le reste sur le disque dur.

Dès que le programme tente d'exécuter une instruction (ou accéder à une donnée) qui n'est pas chargée en mémoire, mais qui se trouve sur le disque dur, une interruption de type exception se produit et le système d'exploitation est réveillé. Il charge alors la partie qui contient cette instruction (ou cette donnée) et rend le contrôle au programme interrompu de l'utilisateur.



RÉSUMÉ :

Dans cette leçon nous avons exposé, du mieux qu'on a pu, les plus importantes fonctionnalités de gestion du matériel par le système d'exploitation. Les plus importantes sont :

- **Gestion du CPU**: Le système d'exploitation, doit à tout moment connaître l'état du processeur et pouvoir le partager à travers plusieurs programmes utilisateurs en sauvegardant le contexte du CPU.
- **Les interruptions** Etant très importantes, le système d'exploitation doit pouvoir les gérer en préparant un ensemble de programmes d'interruption, et en sauvegardant les contextes des programmes interrompus. Il peut gérer deux types d'interruptions :
  - Les interruptions matérielles ;
  - Les interruptions logicielles.
- **Les opérations d'E/S** Sont également prises en compte par le système d'exploitation, plusieurs étapes sont définies à partir du moment où la requête est présentée jusqu'à la réalisation matérielle de l'E/S à l'aides des routines matérielles en passant par les pilotes de périphériques et du gestionnaire d'E/S.
- Enfin, **la gestion de la mémoire** reste une tâche importante et ardue du système d'exploitation. Tous les types de gestion ont été présentés : la segmentation, le recouvrement, la pagination, la mémoire virtuelle.

La gestion du matériel par le système d'exploitation étant présentée, nous pouvons désormais voir la gestion logique du logiciel, des données et des utilisateurs. Ceci fera l'objet de la prochaine leçon.

EXERCICES D'APPLICATION :

QUESTION 1:

Quel est le rôle d'une exception ? Dans le 80486 quel est le numéro de l'exception qui est déclenchée chaque fois qu'un programme essaie d'accéder à un segment non présent en mémoire ? Que se passe-t-il ensuite ?

QUESTION 2 :

Classer par ordre croissant de priorité les interruptions suivantes :

- Exceptions ;
- Interruptions logicielles ;
- Interruptions matérielle non masquables ;
- Interruptions matérielles masquables.

QUESTION 3 :

Quel est le rôle du pilote de périphériques dans un système d'exploitation ?

QUESTION 4 :

Pour accéder à un disque dur, le système d'exploitation utilise des buffers ou tampons. Quel est le rôle de ces buffers ?

QUESTION 5 :

Citer les objectifs d'un gestionnaire de mémoire ?

QUESTION 6 :

Répondez par vrai ou faux ?

- Les segments de mémoire sont de taille fixe ;
- Une page ne peut contenir en mêmes temps des données et des instructions ;
- Un segment peut contenir en mêmes temps des données et des instructions ;
- La mémoire virtuelle ne s'applique pas aux données mais seulement aux programmes ;
- Le programme d'interruption s'exécute dans un contexte distinct de celui du programme interrompu.

CORRECTION DES EXERCICES :

QUESTION 1 :

1- Une exception a essentiellement pour rôle de traiter une anomalie dans le déroulement d'une instruction machine :

- Données incorrectes (débordement arithmétique, division par zéro, ...) ;

- Tentative d'exécution d'une opération interdite par un dispositif de protection (violation de la mémoire, l'exécution d'une instruction privilégiée en mode esclave)

2- Le numéro de l'exception est le 11.
Un programme d'interruption est alors exécuter, généralement c'est le gestionnaire de la MEMOIRE VIRTUELLE, son rôle est de continuer l'exécution du programme interrompu, il doit:

- Chercher une place libre en mémoire ;
- Réserver cette place mémoire ;
- Localiser sur le disque dur le segment absent de la mémoire ;
- Charger ce segment en mémoire à la place réservée ;
- Exécuter l'instruction qui a provoqué l'exception.

QUESTION 2 :

L'ordre est le suivant (du moins prioritaire au plus prioritaire) :

1- Interruptions logicielles ;
2- Interruptions matérielles masquables ;
3- Interruptions matérielles non masquables ;
4- Exceptions.

QUESTION 3 :

Les pilotes de périphériques ont pour rôle:

- Formuler la demande d'E/S sous forme de commandes suivant la nature de l'opération et le type du périphérique ;

- Communiquer avec le matériel d'une façon directe ou à travers les routines de bas niveau ;

- Dans le cas des systèmes multitâches, gérer les files d'attentes des demandes d'E/S derrière un périphérique donné ;

- Gérer également les buffers et tampons (mémoires de stockage intermédiaire entre le périphérique et le programme utilisateur).

QUESTION 4 :

Les buffers ou tampons d'E/S ont deux rôles fondamentaux :

- Ils servent de zones de stockage intermédiaires entre les programmes utilisateur et les périphériques. Car, comme vous l'avez vu, ces zones se trouvent entre les pilotes de périphériques et les routines de bas niveau. Un mode de fonctionnement de serveur/client ou de boite aux lettres est constaté ;

- Ils permettent d'accélérer la vitesse de travail. En effet quand un programme écrit dans le disque, le système d'exploitation n'effectue pas l'opération directement dans le disque (qui sont très lents) mais écrit dans ces buffers en mémoire (opération très rapide). De temps en temps seulement un rafraîchissement est établi (transfert des contenus des tampons vers leurs emplacements dans le disque dur). L'avantage est donc d'éviter d'accéder souvent au disque dur, et donc d'améliorer la vitesse des systèmes.

QUESTION 5 :

Le programme de gestion de la mémoire qui est une partie du système d'exploitation doit viser les objectifs suivants :

1- L'allocation de mémoire libre pour les programmes et leurs données ;

2- La réallocation : un programme ne connaît pas à priori l'environnement d'exécution de son programme ;

3- La protection : la coexistence de plusieurs processus en mémoire centrale nécessite la protection de chaque espace mémoire vis à vis des autres ;

4- Le partage : Parfois, il est utile de partager un espace mémoire entre plusieurs programmes comme par exemple les buffers d'E/S qui doivent être utilisés en même temps par les pilotes de périphériques et les routines de bas niveau.

QUESTION 6 :

- Les segments de mémoire sont de taille fixe**. FAUX**

- Une page ne peut contenir en mêmes temps des données et des instructions. **FAUX**

- Un segment ne peut contenir en mêmes temps des données et des instructions. **FAUX**

- La mémoire virtuelle ne s'applique pas aux données mais seulement aux programmes. **FAUX**

- Le programme d'interruption s'exécute dans un contexte distinct de celui du programme interrompu. **VRAI**

LEÇON N°03 : NOTION D'ORDONNANCEMENT DES PROCESSUS ET CES POLITIQUES

OBJECTIF PÉDAGOGIQUE

À l'issue de cette série, le stagiaire doit être capable de connaître :

1- Notion d'ordonnancement des processus ;
2- Les différentes politiques d'un ordonnanceur.

PLAN DE LA LEÇON :

I- NOTION DU PROCESSEUR

II- L'ORDONNANCEMENT (LE SCHEDULING)

1- Objectifs
2- Critères
3- Priorités
4- Niveaux

III- LES POLITIQUES DE L'ORDONNANCEMENT

1- La politique FIFO (first in first out)
2- La politique SJF (shortest job first)
3- La politique d'ordonnancement par tourniquet (round robin)
4- La politique a plusieurs niveaux

IV- PERFORMANCE DES POLITIQUES D'ORDONNANCEMENT

EXERCICE D'APPLICATION

LEÇON N°03 : NOTION D'ORDONNANCEMENT DES PROCESSUS ET CES POLITIQUES

I- NOTION DU PROCESSEUR :

Un processeur est un circuit intégré complexe caractérisé par une très grande intégration et doté des facultés d'interprétation et d'exécution des instructions d'un programme. Il est chargé d'organiser les tâches précisées par le programme et d'assurer leur exécution. Il doit aussi prendre en compte les informations extérieures au système et assurer leur traitement. C'est le cerveau du système.

II- L'ORDONNANCEMENT (LE SCHEDULING) :

Pour la très grande majorité des ordinateurs, avoir un seul processeur implique de ne pouvoir effectuer qu'un traitement à la fois. Or, à un instant donné, il est possible qu'il y ait plus de processus à exécuter qu'il n'y a de processeurs.

- On appelle **ordonnancement** ou **Scheduling**, l'organisation qui prend en charge l'allocation du processeur central aux programmes.
- On appelle **ordonnanceur** ou **Scheduler** la partie du système d'exploitation qui s'occupe de cette organisation et repartit le temps processeur entre ces programmes.
- Un ordonnanceur fait face à deux problèmes principaux :

Le choix du processus à exécuter, et Le temps d'allocation du processeur au processus choisi.

1- Objectifs :

La technique d'un ordonnanceur doit viser les objectifs suivant :

L'équité : L'ordonnanceur doit servir les programmes d'une manière juste et équitable.

Rendement : L'ordonnanceur doit permettre et assurer l'exécution du maximum de programme (augmenter le rendement).

Utilisation des ressources : L'ordonnanceur doit assurer une occupation maximale des ressources.

2- Critères :

L'objectif d'une politique 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.

Équité : 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- Priorités :

La priorité d'un processus est une information qui permet de classer un processus parmi d'autre lors d'un choix, il existe deux types de priorités :

Priorité fixe : défini a priori une fois pour toute. Elle peut être en fonction du temps estimé du processus.

Priorité variable au cours du temps : c'est une priorité qui peut changer dans le temps en fonction du temps d'attente écoulé, ou du temps déjà eu, etc.

Exemple : un processus qui fait beaucoup d'E/S passe beaucoup de temps à attendre et donc il faut lui allouer le PC dès qu'il le demande.

4- Niveaux :

Il est possible de distinguer trois niveau d'ordonnanceurs : à long terme, à moyen terme et à court terme. Leurs principales fonctions sont les suivantes :

À long terme : L'ordonnanceur fait la sélection de programmes à admettre dans le système pour leur exécution. Les programmes admis deviennent des processus à l'état **prêt**. L'admission dépend de la capacité du système (degré de multiprogrammation) et du niveau de performance requis.

À moyen terme : Il fait la sélection de processus déjà admis à débarquer ou rembarquer sur la mémoire. Il effectue ses tâches de gestion en fonction du degré de multiprogrammation du système, et aussi des requêtes d'E/S des périphériques.

À court terme : L'ordonnanceur à court terme a comme tâche la gestion des processus prêts. Il sélectionne en fonction d'une certaine politique le prochain processus à exécuter. Il effectue aussi le changement de contexte des processus. Il peut implanter un :

- **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.

- **Ordonnancement coopératif (non préemptif) :**

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.

L'ordonnanceur est activé par un événement : interruption du temporisateur, interruption d'un périphérique, appel système ou signal.

III- POLITIQUES DE L'ORDONNANCEMENT :

1- La politique 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

Exemple :

Exemple : **N° JOB**	**Temps d'exécution estime**	**Temps d'arrivé**	**Temps début d'exécution**	**Temps fin d'exécution**	**Temps réponse**
1	2 h	10 h 00	10 h 00	12 h 00	2 h
2	1h	10 h 10	12 h 00	13 h 00	2 h 50
3	25mn	10 h 25	13 h 00	13 h 25	3 h

Remarque :

- Le 1er job arrivé dans le système est servi.
- C'est une technique (politique) de scheduling non préemptive (coopérative) qui désavantage les jobs courts.
- Non préemptive== non arrêté algorithme avec réquisition AAR

2- La politique 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). À 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.

Exemple :

N° Job	Temps d'Exécution Estime	Temps Arrives	Temps Début Exécution	Temps Fin Exécution	Temps Réponse
1	2 h	10 h 00	10 h 00	12 h 00	2 h
2	1h	10 h 10	12 h 25	13 h 25	03 h 15
3	25mn	10 h 25	12 h 00	12 h 25	2 h

3- La politique d'ordonnancement par tourniquet (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.

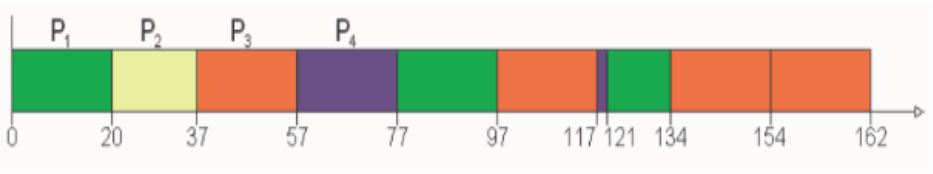
La commutation de processus dure un temps non nul pour la mise à jour des tables, la sauvegarde des registres. Un quantum trop petit provoque trop de commutations de processus et abaisse l'efficacité du processeur. Un quantum trop grand augmente le temps de réponse en mode interactif. On utilise souvent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

Exemple :

Etant donnes les processus suivants :

P1 d'une durée de 53 UT ; P2 d'une durée de 17 UT ;
P3 d'une durée de 68 UT ; P4 d'une durée de 24 UT

Nous paramétrons Q (quota) à 20 UT, nous obtenons le diagramme suivant :



Remarque :

- Temps de rotation et temps d'attente moyens
- Aucun processus n'est favorisé

4- La politique a plusieurs niveaux :

Les politiques présentées jusqu'à présent utilisent une seule file d'attente des processus prêts.

On choisit ici de définir plusieurs files de processus prêts, chaque file correspondant à un niveau de priorité ; on peut alors avoir n files de priorités différentes variant de 0 à n-1. Dans

une file donnée, tous les processus ont la même priorité et sont servis soit selon une politique à l'ancienneté sans préemption, soit selon une politique de tourniquet.
Le quantum peut être différent selon le niveau de priorité de la file. L'ordonnanceur sert d'abord tous les processus de la file de priorité n, puis ceux de priorité n-1 dès que la file de niveau n est vide et ainsi de suite...

On peut définir deux variantes de l'algorithme, fonction de l'évolution de la priorité des processus :

• Les priorités des processus sont constantes tout au long de leur exécution. À ce moment-là, un processus en fin de quantum est toujours réinséré dans la même file d'attente, celle correspondant à son niveau de priorité.

• Les priorités des processus évoluent dynamiquement en fonction du temps de service dont a bénéficié le processus. Ainsi un processus de priorité n, à la fin du quantum de la file n, s'il n'a pas terminé son exécution, n'est pas réinséré dans la file de priorité n, mais dans la file de priorité n-1. Et ainsi de suite... On cherche ici à minimiser les risques de famine pour les processus de faible priorité en faisant petit à petit baisser la priorité des processus de plus forte priorité.

IV- PERFORMANCE DES POLITIQUES D'ORDONNANCEMENT :

Les performances d'une politique pour un ensemble de processus donné peuvent être analysées si les informations appropriées relatives aux processus sont fournies.

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

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

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

$$\text{Rendement} = \frac{\sum Temps\ d'exécution}{nbre\ de\ processus}$$

EXERCICE D'APPLICATION :

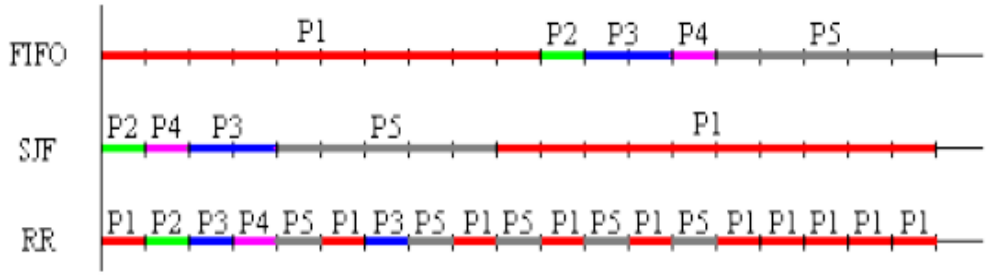
1- Citez les différentes politiques d'un ordonnanceur ?
2- Quelle est la différence entre un ordonnanceur préemptif et non préemptif ?
3- Cinq processus, P1, P2, P3, P4, P5 sont dans une file d'attente dans cet ordre (P1 est le premier, P5 est le dernier). Leur exécution demande un temps total de service exprimé en unités arbitraires :

Processus	P1	P2	P3	P4	P5
Temps de service estimé	10	1	2	1	5

1- Décrire l'exécution des processus dans le cadre des politiques d'ordonnancement FIFO, SJF, RR (avec un quantum de 1).

SOLUTION :

- Question 1 et 2 se référer au cours.
- Le schéma ci-dessous décrit l'enchaînement des processus :



	P1	**P2**	**P3**	**P4**	**P5**	**TOTAL**	**TEMPS MOYEN**
FIFO	10	11	13	14	19	57	13.4
SJF	19	1	4	2	9	35	7
RR	19	2	7	4	14	46	9.2

Le tableau, ci-dessus, indique les temps de présence, dans le système, des processus. La dernière colonne décrit le temps moyen passé par chaque processus. Il est clair que, sur cet exemple, **la stratégie SJF est la meilleure.**

RÉFÉRENCES BIBLIOGRAPHIQUES :

1. CT BULABULA DEMA Faustin, Cours de Systèmes d'exploitation et le Bureautique I, ISP/Bukavu en G1 IG, Inédit, 2004-2005
2. Ass TASHO KASONGO, Cours de l'Introduction à l'Informatique, ISP/Bukavu en G1 IG, Inédit, 2006-2007
3. Ass KYENDA SULIKA Dieu-donné, Cours de Systèmes d'exploitation, ISP/Bukavu en G1 IG, Inédit, 2000-2001.
4. AUMIAUX, M. Initiation à l'informatique de gestion, 2ème éd. Masson, Paris, 1983.
5. CAPRON, H.L. Computer, a tool for information age, 3ème ed. Upper saddle river, New Jersey, 2000.
6. CHAUMEL J.L.: L'implantation d'une technologie, nouvelle leçon corrigée, ed. Hériat, Montréal, 1989.
7. DANIEL, C. : Organiser le développement de la micro-informatique, Ed. d'organisations, Paris, 1987.
8. DELEPONE, J.F., Introduction théorique à l'informatique, Africa Computing, Abidjan, 2004.
9. DONALD H. : L'informatique : Un instrument de la gestion, Mc Graw-Hill, 1980.
10. DULONG, A. et SUTTER, E. : Les technologies de l'information, Paris, 1990,
11. JAVEAU, C. : Enquête par questionnaire, éd. ULB, 1995.
12. LOOIJEN M. : Information Systems : Management, Control and Maintenance, Kluwer Bedrijfsinformatique, 1998.

WEBO GRAPHIE

http://www.bestcours.com/systeme-exploitation
http://depinfo.u-cergy.fr/~pl/docs/slidesOS.pdf
http://deptinfo.cnam.fr/Enseignement/CycleA/AMSI/transparents_systemes/14_gestion_memoire.pdf

LEÇON N°04 : LES MÉCANISMES DE GESTION DE LA MÉMOIRE ET STRATÉGIE D'ALLOCATIONS DE LA MÉMOIRE

OBJECTIF PÉDAGOGIQUE

À l'issue de cette série le stagiaire doit être capable de **c**onnaître :

1- Les mécanismes de gestion de la mémoire.

2- Les différentes stratégies d'allocation de la mémoire.

PLAN DE LA LEÇON :

INTRODUCTION

I- OBJECTIFS

II- FONCTIONS D'UN GESTIONNAIRE DE MÉMOIRE

III- STRATÉGIE D'ALLOCATIONS DE LA MÉMOIRE

VI- NOTION DE MÉMOIRE VIRTUELLE

EXERCICES

LEÇON N°04 : LES MÉCANISMES DE GESTION DE LA MÉMOIRE ET STRATÉGIE D'ALLOCATIONS DE LA MÉMOIRE

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.

I- OBJECTIFS :

La gestion de la mémoire est un difficile compromis entre les performances (temps d'accès) et la quantité (espace disponible). On désire en effet tout le temps avoir le maximum de mémoire disponible, mais l'on souhaite rarement que cela se fasse au détriment des performances.

La gestion de la mémoire doit de plus remplir les objectifs suivants :

- Permettre le partage de la mémoire (pour un système multitâches) ;
- Permettre d'allouer des blocs de mémoire aux différentes tâches ;
- Protéger les espaces mémoire utilisés (empêcher par exemple à un utilisateur de modifier une tâche exécutée par un autre utilisateur) ;
- Optimiser la quantité de mémoire disponible, notamment par des mécanismes d'extension de la mémoire.

II- FONCTIONS D'UN GESTIONNAIRE DE MÉMOIRE :

Le gestionnaire de mémoire est un sous-ensemble du système d'exploitation. Son rôle est de partager la mémoire entre l'OS et les diverses applications. Le terme "mémoire" fait surtout référence à la mémoire principale, c'est à dire à la RAM, mais la gestion de celle-ci demande la contribution de la mémoire auxiliaire (mémoire de masse, spacieuse mais lente) et à la mémoire cache (rapide mais de taille restreinte).

Voici quatre fonctions qu'on attend du gestionnaire de mémoire :

- **L'allocation de la mémoire aux processus :**
  - Répertorier les emplacements libres de la mémoire
  - Allouer la mémoire nécessaire aux nouveaux processus
  - Récupérer la mémoire des processus qui s'achèvent.

Cette récupération peut nécessiter une réallocation des processus en cours pour optimiser l'emploi de la mémoire. La zone mémoire attribuée à un processus peut donc changer au cours de son exécution.

- **La protection**

Il faut s'assurer que les adresses générées par chaque processus ne concernent que la zone mémoire qui lui est impartie, sans quoi, l'intégrité du système d'exploitation et des autres processus n'est pas garantie.

Certaines zones mémoire doivent pourtant servir simultanément à plusieurs processus : le code de fonctions servant à plusieurs applications qui tournent en parallèle ou les données utilisées simultanément par divers processus

- **La segmentation de l'espace d'adressage :**

Les programmes sont subdivisés en segments : le code, les données modifiables, celles qui ne le sont pas, la pile. On attend donc du gestionnaire de mémoire qu'il permette la segmentation de l'espace d'adressage des programmes pour les raisons suivantes :

- Pouvoir coder les segments séparément et les paramétrer en fonction de l'application.
- Permettre des degrés de protection différents selon les segments (lecture seule, exécution...)
- Accepter le partage de certains segments.

- **La mémoire virtuelle**

Elle offre aux applications une mémoire de taille supérieure à celle de la mémoire principale.

L'espace d'adressage qu'offre la mémoire centrale est parfois insuffisant. Les disques suppléent à cette insuffisance en fournissant une mémoire auxiliaire plus vaste mais plus lente et qui n'est pas directement accessible au processeur.

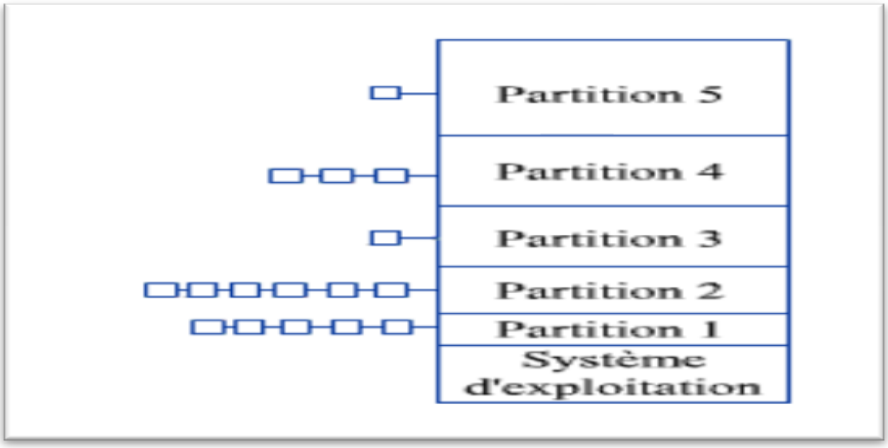
II- STRATÉGIE D'ALLOCATIONS DE LA MÉMOIRE :

Plusieurs processus doivent se partager la mémoire sans empiéter sur l'espace réservé au système d'exploitation ni aux autres processus. Quand un processus se termine, le S.E. doit libérer l'espace mémoire qui lui était alloué pour pouvoir y placer de nouveaux processus.

1-Allocation en zones contiguës :

1.1- Partitions fixes :

Le plus simple est de diviser la mémoire en partitions fixes dès le démarrage du système. Les partitions sont de différentes tailles pour éviter que de grandes partitions ne soient occupées que par de petits processus. Le gestionnaire de mémoire, en fonction de la taille des processus, décide quelle partition lui allouer pour ne pas gaspiller trop de mémoire.



Une file d'attente est associée à chaque partition. Quand vient une nouvelle tâche, le gestionnaire détermine quelle est la plus petite partition qui peut la contenir puis place cette tâche dans la file correspondante.
Le fait d'éviter d'allouer une partition trop grande à un petit processus conduit parfois à des aberrations. Il arrive que des partitions plus grandes restent inutilisées alors que se forment ailleurs des files interminables de petits processus. La mémoire est donc mal utilisée.

Une autre solution est de créer une file unique. Lorsqu'une partition se libère, on consulte la file pour trouver la tâche qui l'occuperait de manière optimale.

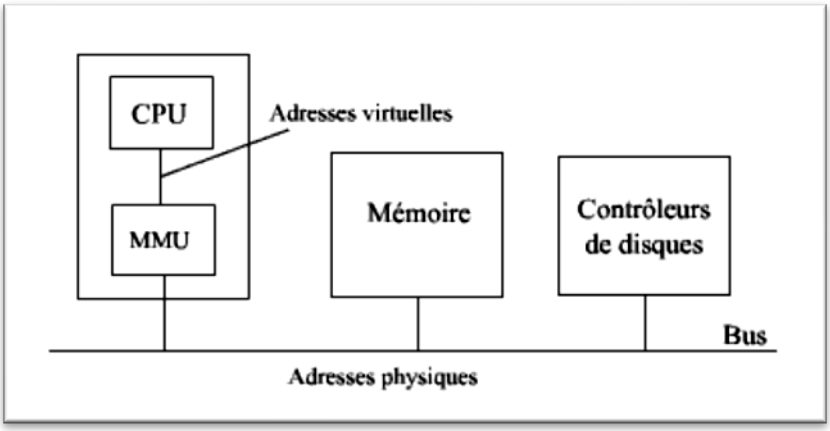
Le risque est que les petites tâches soient pénalisées. Une parade est de conserver une petite partition au moins qui ne sera accessible qu'aux petites tâches. Une autre solution, serait de dire qu'un processus ne peut être ignoré qu'au maximum un certain nombre de fois. Après **n** refus, il prendra place dans une partition même si la partition est bien plus grande qu'il ne faut.

1.2- Partitions variables *:*

Une autre manière d'éviter les emplacements mémoires inoccupés en fin de partitions est d'allouer aux processus des espaces qui correspondent exactement à l'espace qui leur est utile.

Au fur et à mesure que les processus se créent et se terminent, des partitions s'allouent et se libèrent laissant des zones mémoires morcelées et inutilisables.

La mémoire se fragmente et est de plus en plus mal employée. Il faudrait la compacter en déplaçant régulièrement les processus mais cette tâche supplémentaire ralentit le système.



1.3- Conclusion :

Le partitionnement de la mémoire que ce soit avec des partitions de tailles fixes ou de tailles variables, ne permet pas d'utiliser la mémoire au mieux.

2- La pagination :

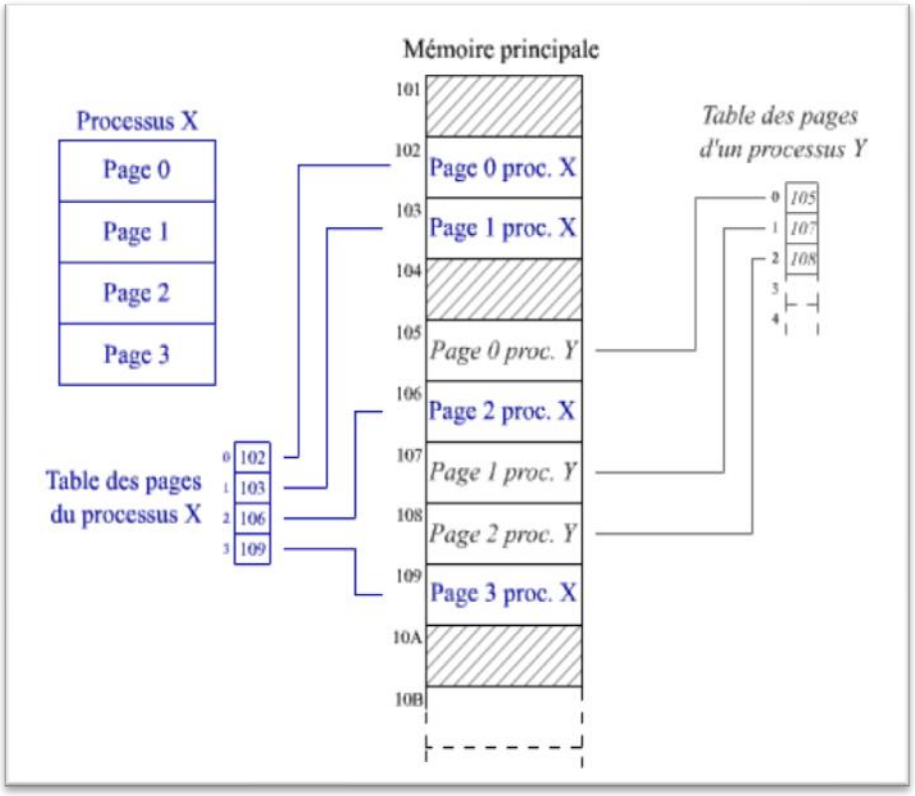
Les processus requièrent des espaces d'adresses continus. On a vu que cela est difficilement réalisable en découpant la mémoire en partions dont les tailles correspondent à celles des processus. La pagination est une technique d'allocation de la mémoire bien plus efficace. Elle fournit aux processus des *espaces d'adresses* séquentiels à partir d'*espaces mémoire* discontinus.

La pagination consiste à diviser la mémoire et les processus en blocs de mêmes tailles appelés **pages**. Les *pages mémoire* sont souvent appelées *"frames"* ou *"cadres"* tandis que les *pages de processus* sont simplement appelées *"pages"*.

Les pages (de processus) ne sont pas toutes simultanément actives ; elles ne sont donc pas nécessairement toutes présentes simultanément dans la mémoire principale. Les pages inactives attendent sur le disque. L'*espace d'adressage* est donc *virtuel* sa taille peut être supérieure à celle de la mémoire réelle.

Les processeurs disposent actuellement d'un dispositif, le MMU *"Memory Manager Unit"* qui permet de placer des processus en mémoire sans nécessairement placer les pages de processus dans des cadres de pages contigus. On distingue les adresses logiques qui se réfèrent aux pages de processus des adresses physiques qui se réfèrent aux cadres de pages.

Voyons à présent comment l'unité de gestion mémoire (MMU) met en correspondance les adresses physiques et logiques. Elle contient pour ce faire une table de pages où sont inscrits les numéros des cadres de pages.



- Fonctionnement des tables de pages :

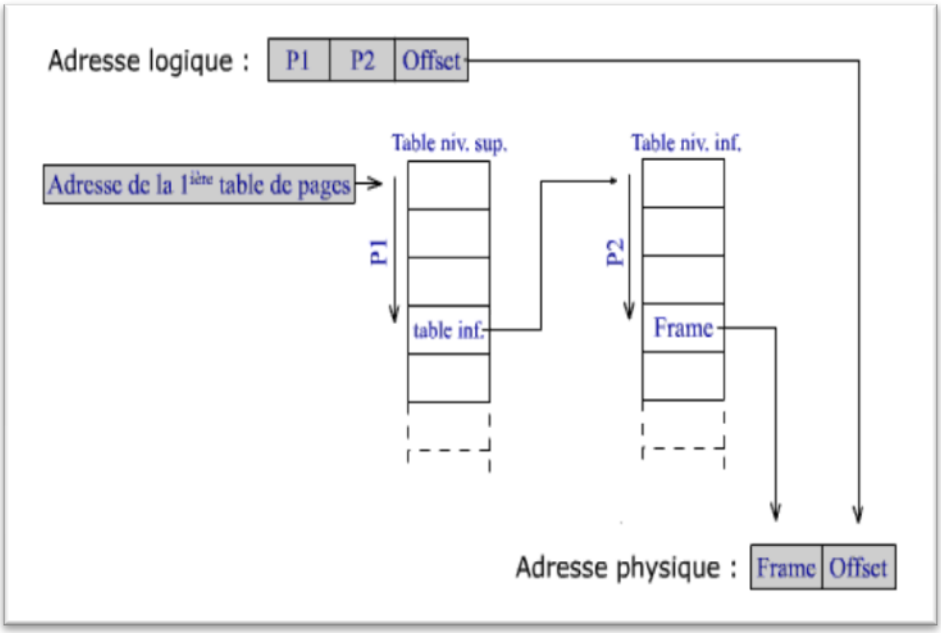
L'adressage se fait au moyen de *numéros de pages* et d'*offsets*. L'offset (= déplacement ou décalage) est la position relative au début de la page.

L'adresse logique est composée du numéro de *page de processus* et d'un offset.

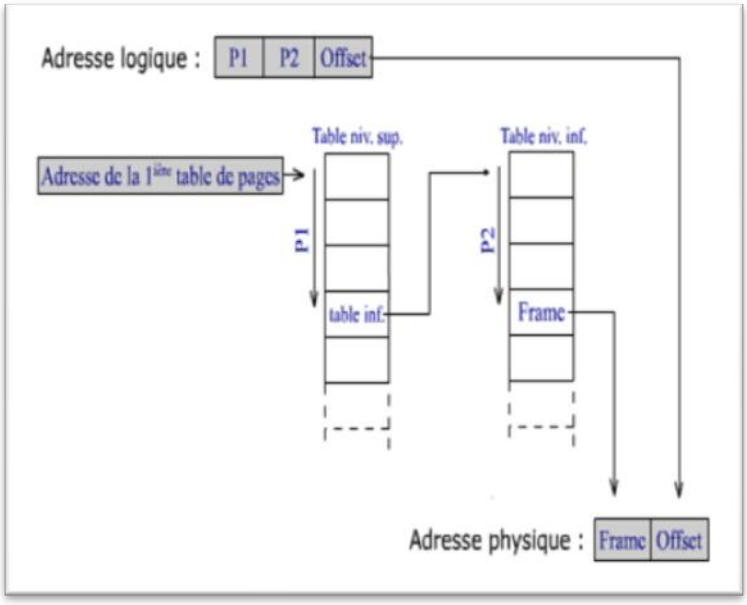
L'adresse physique correspondante est formée à partir du numéro du *cadre de page* où est chargé la page de processus et du même offset que celui de l'adresse logique.

Le numéro du cadre de page est consigné dans une table des pages associée au processus. On y retrouve le numéro du cadre de page en se servant du numéro de page de processus comme d'un index.

Pagination simple

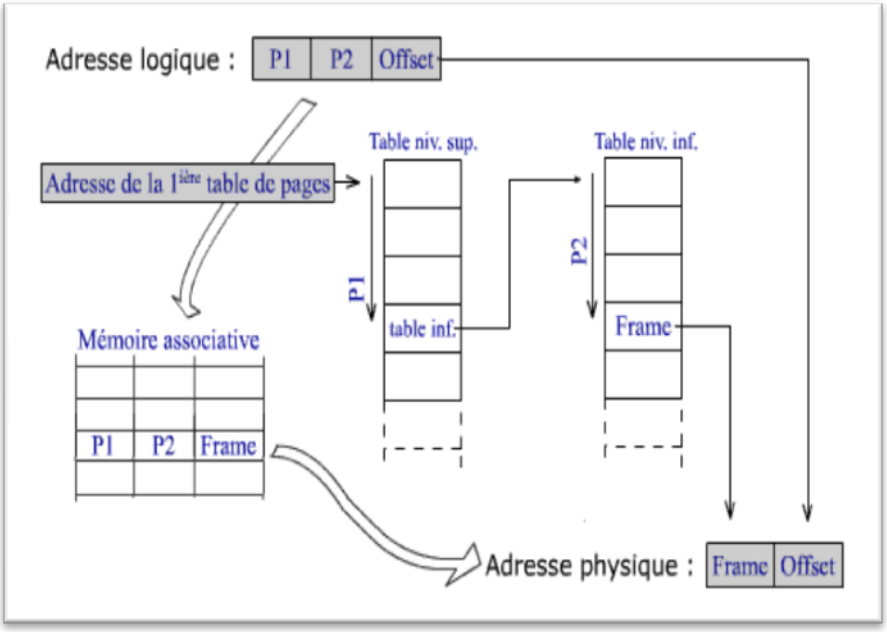


Le nombre de pages étant souvent très grand les tables des pages deviennent volumineuses et peuvent même occuper ... plusieurs pages. On les fractionne donc en plusieurs niveaux : une table de page de niveau supérieur dont chaque élément pointe vers une table de niveau inférieur. L'adresse logique contient dès lors deux nombres pour aboutir au numéro de page. Le premier sert d'index dans la table de niveau supérieur, le second sert d'index dans la table du niveau suivant.



Tables de pages multi niveaux

Ces accès multiples à différentes pages pour aboutir à l'adresse finale ralentissent fortement l'adressage. On évite de répéter ces recherches en notant les correspondances trouvées entre les adresses logiques et les adresses physiques dans une mémoire associative. Ce qui permet ensuite de retrouver presque immédiatement les correspondances les plus récentes.



NB. L'espace d'adressage est perçu par le programmeur comme une suite continue d'octets. La subdivision de l'adresse en numéros de page et d'offset est transparente. Elle est prise en charge par le matériel.

3- La segmentation :

Chaque processus est constitué d'un ensemble de segments. Chaque segment est un espace linéaire.

Les segments sont des espaces d'adressages indépendants de différentes longueurs et qui peuvent même varier en cours d'utilisation. Ils correspondent à des subdivisions logiques déterminées par le programmeur ou par le compilateur.

Les segments contiennent des informations de même nature : le code, les données, la pile, des tables, etc. Il est dès lors possible d'attribuer des protections adaptées à chaque type de segment : un segment de code peut être déclaré en exécution seule, une table de constantes en lecture seule mais pas en écriture ni en exécution. Certaines zones de code en exécution seule peuvent être partagées par plusieurs processus ; cela se fait par exemple pour des bibliothèques de sous-programmes.

L'accès aux segments se fait via une table de segments.

Chaque entrée de la table comporte l'adresse de départ du segment et sa taille.

L'adresse logique est constituée du numéro de segment et d'un offset. Contrairement aux pages dont le fonctionnement est transparent pour le programmeur, les segments sont des entités logiques qu'il connaît et manipule. Il distingue les deux informations contenues dans l'adresse : le numéro du segment et l'offset.

Le numéro de segment sert d'index pour retrouver l'adresse du début du segment dans la table de segment. Cet offset doit être inférieur à la taille du segment consignée elle aussi dans la table de segment. Si ce n'est pas le cas, une erreur est générée qui provoque l'abandon du programme. L'offset est ensuite ajouté à l'adresse de début de segment pour former l'adresse physique.

4- Segmentation avec pagination :

La segmentation et la pagination concernent des problèmes différents. Ce sont deux techniques qui peuvent se combiner :

- La segmentation découpe les processus en zones linaires pouvant être gérées différemment selon que ces segments sont propres au processus, qu'ils sont partagées, lus, écrits ou exécutées et de manière à protéger les processus entre eux.
- La pagination découpe la mémoire en pages non contiguës mais de même taille. Elle procure aux processus des espaces d'adresse continus (nécessaires aux segments). Les pages mémoires peuvent n'être allouées que lorsqu'un processus en a besoin. On obtient de la sorte une mémoire virtuelle de taille supérieure à la mémoire réelle.

Les systèmes d'exploitation qui gèrent ces deux techniques simultanément administrent **une table de segments et plusieurs tables de pages**.
Un segment peut contenir plusieurs pages mais toutes ne doivent pas nécessairement être présentes en mémoire à tout moment. On ne garde en mémoire que celles qui sont réellement utilisées. Chaque segment a sa propre table de pages.

I.V- 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.

EXERCICES D'APPLICATIONS :

EXERCICE N°01 : Segmentation

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

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

1- Calculez les adresses réelles correspondant aux adresses virtuelles suivantes (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 ?

Rappel : Les adresses sont données sous la forme (n°segment: déplacement)

EXERCICE N°02 : Pagination

Dans un système paginé, les pages font 256 mots mémoire et on autorise chaque processus à utiliser au plus 4 cadres de la mémoire centrale. On considère la table des pages suivante du processus P1 :

Page	0	1	2	3	4	5	6	7
Cadre	011	001	000	010	100	111	101	110
Présence	oui	non	oui	non	non	non	oui	non

1- Quelle est la taille de l'espace d'adressage du processus P1 ?
2- De combien de mémoire vive dispose ce système ?
3- Calculez les adresses réelles correspondant aux adresses virtuelles suivantes (vous signalerez éventuellement les erreurs d'adressage) :240, 546, 1578, 2072
4- Que se passe-t-il si P1 génère l'adresse virtuelle 770 ?
5- On considère l'adresse virtuelle suivante: 0000 0000 0000 0111. Sachant que les 4 bits de poids fort désigne le numéro de page et que 12 bits suivants représentent le déplacement dans la page, donnez l'adresse physique (exprimée en binaire) correspondant à cette adresse.

EXERCICE N°03 : Citez les objectifs de la gestion de la mémoire

CORRECTION DES EXERCICES :

EXERCICE N°01 :

1- L'adresse physique s'obtient en ajoutant l'adresse de base du segment au déplacement dans le segment, mais à condition que le déplacement ne soit pas supérieur à la taille du segment moins 1 (on compte le déplacement en partant de 0) :

- (0:128) : déplacement valide (128<234). Adr_physique = base + limite =540+128 = 668.
- (1:100) : déplacement valide (100<128). Adr_physique = bas +limite= 1254+100 = 1354.
- (2:465) : déplacement invalide (465>328).
- (3:888) : déplacement valide (888<1024). Adr_physique= base+limite= 2048+888= 2936.
- (4:100) : déplacement valide (100<200). Adr_physique = base+ limite= 976+100 = 1076.
- (4:344) : déplacement invalide car (344>200)

2- Non. Dans un segment de longueur 200, les déplacements valides sont dans l'intervalle [0-199]

EXERCICE N°02 :

1- L'espace d'adressage du processus est l'espace d'adressage virtuel formé par les pages. Comme il y a 8 pages, la taille de l'espace virtuel est de 8*256 = 2048 mots.
2- Comme les cadres sont numérotés sur 3 bits, il y a 23= 8 cadres. Taille d'un cadre = taille d'une page donc la mémoire physique comporte 8*256 =2048 mots (= 2Ko).
3- La conversion d'une adresse virtuelle en adresse réelle est réalisée de la façon suivante**:**

a. Calcul du numéro de la page et du déplacement dans la page.
b. Recherche dans la table de pages de l'entrée qui correspond à la page de façon à en déduire le numéro du cadre.
c. L'adresse physique (réelle) est obtenue en ajoutant le déplacement à l'adresse physique de début du cadre.

Voici le détail des calculs pour les adresses demandées :

- 240 = 0*256 + 240 → page = 0 et déplacement = 240

D'après la table des pages, cadre = 3. D'où Adr_phys =3*256 + 240 = 1008
- 546 = 2 *256 + 34 → page = 0 et déplacement = 34.

D'après la table des pages, cadre = 0. D'où Adr_phys =0*256+ 34 = 34.
- 1578 = 6*256 + 42 → page = 6 et déplacement = 42.

D'après la table des pages, cadre = 5. D'où Adr_phys =5*256 + 42 = 1322.
- 2072 est en dehors de l'espace d'adressage virtuel du processus (2048 mots).

4- 770 = 3*256 + 2. Il s'agit d'une adresse située dans la page 3. Or d'après la table des pages, cette page n'est pas présente en mémoire. Une référence à cette adresse provoquera donc un défaut de page.

5- D'après la table de pages, cette page se trouve dans le cadre 010. L'adresse physique s'obtient donc simplement en substituant aux 4 bits de poids fort de l'adresse virtuelle les 3 bits du numéro de cadre : 010 0000 0000 0111.

EXERCICE N°03 :

- Protection : par défaut, un processus ne doit pas pouvoir accéder à l'espace d'adressage d'un autre
- Partage : s'ils le demandent, plusieurs processus peuvent partager une zone mémoire commune,
- Réallocation : les adresses vues par le processus (adresses relatives ou virtuelles) sont différentes des adresses d'implantation (adresses absolues).

RÉFÉRENCES BIBLIOGRAPHIQUES :

13. CT BULABULA DEMA Faustin, Cours de Systèmes d'exploitation et le Bureautique I, ISP/Bukavu en G1 IG, Inédit, 2004-2005
14. Ass TASHO KASONGO, Cours de l'Introduction à l'Informatique, ISP/Bukavu en G1 IG, Inédit, 2006-2007
15. Ass KYENDA SULIKA Dieu-donné, Cours de Systèmes d'exploitation, ISP/Bukavu en G1 IG, Inédit, 2000-2001.
16. AUMIAUX, M. Initiation à l'informatique de gestion, 2ème éd. Masson, Paris, 1983.
17. CAPRON, H.L. Computer, a tool for information age, 3ème ed. Upper saddle river, New Jersey, 2000.
18. CHAUMEL J.L.: L'implantation d'une technologie, nouvelle leçon corrigée, ed. Hériat, Montréal, 1989.
19. DANIEL, C. : Organiser le développement de la micro-informatique, Ed. d'organisations, Paris, 1987.
20. DELEPONE, J.F., Introduction théorique à l'informatique, Africa Computing, Abidjan, 2004.
21. DONALD H. : L'informatique : Un instrument de la gestion, Mc Graw-Hill, 1980.
22. DULONG, A. et SUTTER, E. : Les technologies de l'information, Paris, 1990,
23. JAVEAU, C. : Enquête par questionnaire, éd. ULB, 1995.
24. LOOIJEN M. : Information Systems : Management, Control and Maintenance, Kluwer Bedrijfsinformatique, 1998. .

WEBOGRAPHIE :

http://www.bestcours.com/systeme-exploitation
http://depinfo.u-cergy.fr/~pl/docs/slidesOS.pdf
http://deptinfo.cnam.fr/Enseignement/CycleA/AMSI/transparents_systemes/14_gestion_memoire.pdf

LEÇON N°05 : LES PÉRIPHERIQUES D'ENTRÉE/SORTIE

OBJECTIF PÉDAGOGIQUE

À la fin de la série le stagiaire doit être capable de :

* Connaître l'utilité et l'organisation des périphériques d'entrée/sortie ;
* Connaître le principe de fonctionnement des périphériques d'entrée/sortie ;
* Connaitre les différents modes d'entrée/sortie.

PLAN DE LA LEÇON:

INTRODUCTION

I- ORGANISATION DES DISPOSITIFS D'ENTRÉES ET SORTIES
II- CONTROLE DES ENTRÉES ET SORTIES
III- PORTS DES ENTRÉES ET SORTIES
IV- LES DIFFERENTS MODES D'ENTRÉES ET SORTIES PHYSIQUES

1- Les entrées-sorties synchrone.
2- Les entrées-sorties asynchrone.
3- Les entrées-sorties par accès direct à la mémoire (DMA).
4- Les entrées-sorties tamponnées.

QUESTIONS DE COURS

LEÇON N°05 : LES PÉRIPHERIQUES D'ENTRÉE/SORTIE

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.

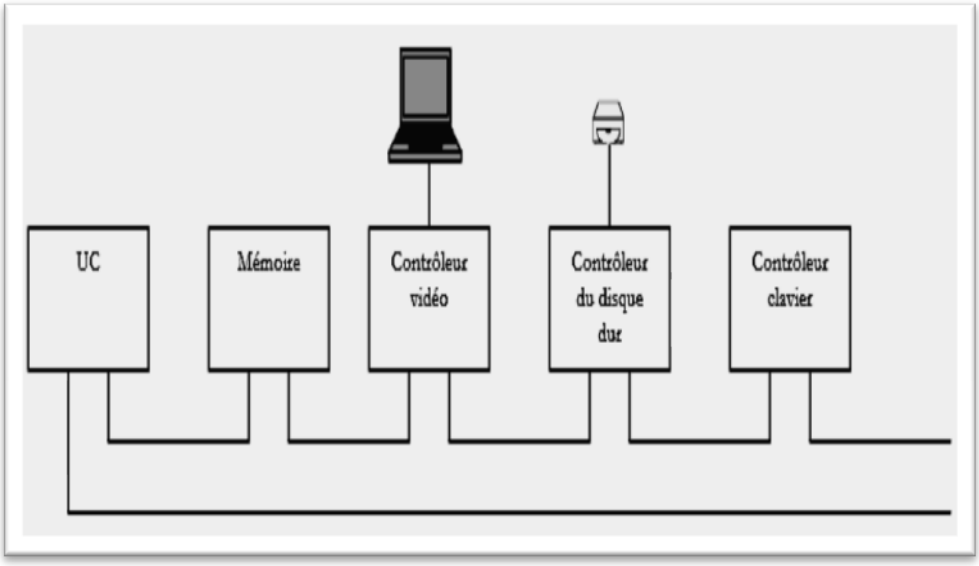
Le système d'exploitation, qui la plupart du temps s'exécute sur une seule UC, doit donc gérer des requêtes simultanées en provenance d'un grand nombre de périphérique.

I- ORGANISATION DES DISPOSITIFS D'ENTRÉES ET SORTIES :

Même si certains ordinateurs sont différents dans les détails, ils sont conçus autour de la même philosophie. Les dispositifs d'E/S, la 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.

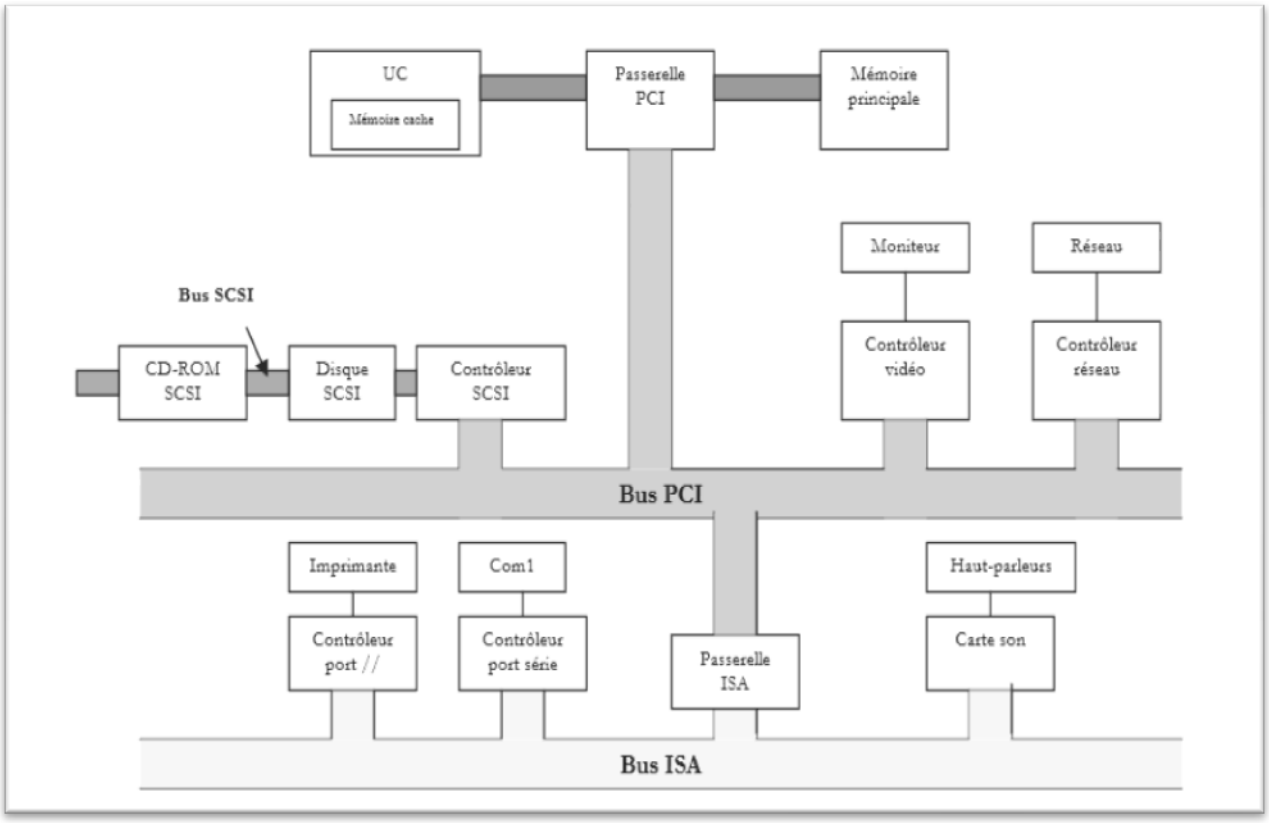
Des accès fréquents à la mémoire et une vitesse relativement rapide de l'UC conduisent à une utilisation élevée du bus par cette dernière. Bien que le bus leur soit fréquemment nécessaire, les E/S ont des besoins en communication généralement plus urgents que les requêtes de l'UC. C'est pourquoi les 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



II- CONTROLE DES ENTRÉES ET SORTIES :

Dans le modèle le plus simple l'UC communique directement avec les périphériques d'entrées et sorties 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 d'E/S possède un contrôleur spécifique.

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

III- PORTS DES ENTRÉES ET SORTIES :

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).

IV- LES DIFFERENTS MODES D'ENTREES/SORTIES PHYSIQUES:

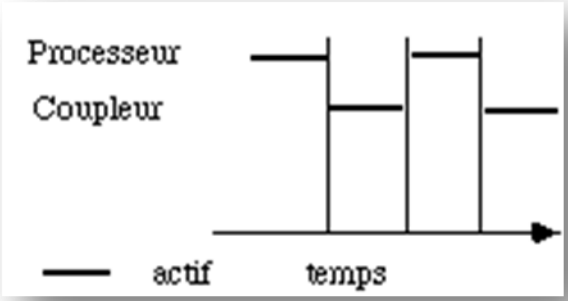
Plusieurs modes d'entrées-sorties ont été proposés dans les systèmes informatiques : les E/S programmées, les E/S direct synchrone et asynchrone ; les E/S tamponnées, les E/S avec accès DMA et les E/S avec processeur spécialisé.

1- Les entrées-sorties synchrone :

Lors d'entrées-sorties synchrone le processeur est immobilisé pendant toute la durée du transfert (fig. 3). Le coupleur contient un mot d'état qui indique entre autre :

- S'il est prêt lorsque le périphérique est apte à fonctionner
- Fini lorsque le transfert est terminé et qu'il est prêt pour un nouvel échange. Le processeur peut alors reprendre son activité
- Erreur lorsqu'une erreur est détectée au cours du transfert. La nature de celle-ci est indiquée par un code qui fait partie du mot d'état. La qualité du diagnostic dépend de l'électronique (nombre de bits de contrôle par octet) et des possibilités de traitement du système d'exploitation.

Figure 3 : Couplage synchrone

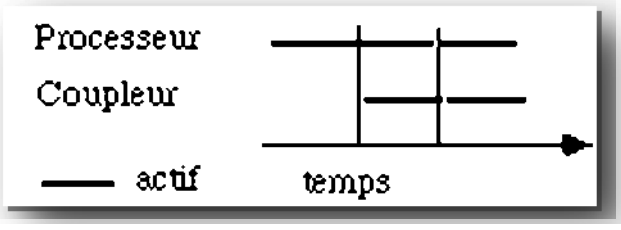


Les entrées-sorties synchrones ne présentent d'intérêt que pour des processeurs rudimentaires lorsqu'il n'y a pas de raison de vouloir mieux employer les temps d'attente. C'est le cas, par exemple, du microprocesseur affecté au clavier d'un ordinateur. Leur programmation est simple puisque l'état de l'activité, en chaque point interruptible, est toujours parfaitement déterminé. De fait les coupleurs synchrones ne se rencontrent que dans des dispositifs spécialisés dont l'état est parfaitement prévisible à tout instant. Leur programmation est facile donc efficace et ne demande que peu de mémoire pour stocker le programme.

2- Les entrées-sorties asynchrone :

Le pilotage des entrées-sorties asynchrones est plus complexe. On ne peut pas prévoir à l'avance l'état des différents processus qui devront communiquer entre eux... Leur programmation nécessite le recours à des interruptions car le processeur et le coupleur travaillent simultanément (fig. 4) à la différence du pilotage synchrone où ils travaillent en alternance (fig. 3). Chacun doit pouvoir être interrompu à des moments imprévisibles entre deux instructions. Ce mode de fonctionnement est évidemment plus performant puisque le processeur n'est pas immobilisé inutilement : lorsqu'il attend que le coupleur ait effectué une opération d'écriture pour un processus il peut retourner à d'autres activités. Cependant ceci est la source de nombreuses possibilités d'erreurs. Il faut veiller à synchroniser correctement le périphérique et le processeur. Les données doivent être prêtes au moment voulu. Une mauvaise synchronisation est la source de nombreuses erreurs souvent difficiles à détecter et à corriger.

Figure 4 : Couplage asynchrone



Les canaux et dispositifs DMA fonctionnent généralement dans ce mode. Le processeur peut consulter leur mot d'état pour connaître leur activité. Dans le dispositif DMA le plus simple il se réduit à un registre qui contient l'adresse en mémoire des données à transférer, le nombre d'octets et le sens du transfert. Le processeur connaît l'état d'avancement du travail grâce à cette information.

3- Les entrées-sorties par accès direct à la mémoire (DMA) :

Le dispositif DMA est un composant matériel permettant d'effectuer des échanges entre la mémoire centrale et unité d'échange sans utilisation du processeur ; Avec cette technique, les données ne doivent pas transiter par l'UC. Cet accès est utile pour des dispositifs tels que les disques

4- Les entrées-sorties tamponnées :

Il est apparu très vite que la réalisation matérielle d'une opération d'entrées-sorties par le processeur de calcul conduisait à une mauvaise rentabilité de la machine. Les concepteurs du matériel ont donc introduit des processeurs spécialisés qui prenaient en charge ces opérations de façon autonome. Il était ainsi possible de poursuivre les traitements pendant l'exécution de l'opération. Ceci a permis de connecter de nouveau les périphériques de type lecteur de cartes ou imprimante sur l'ordinateur principal, et de supprimer l'ordinateur secondaire. Le superviseur d'entrées-sorties assure la lecture des cartes dans une zone dédiée de mémoire centrale, avant que le programme n'en ait effectivement besoin, permettant ainsi de satisfaire immédiatement sa demande ultérieure. De même, lorsque le programme demande une impression, celle-ci est remplacée par une recopie dans une zone dédiée de mémoire centrale, le superviseur assurant l'impression du contenu de cette zone ultérieurement.

Ce mode de fonctionnement n'a été rendu possible que par l'introduction du mécanisme d'*interruption*, permettant à un dispositif extérieur d'arrêter momentanément le déroulement normal d'un programme pour exécuter un traitement spécifique. Par exemple, lorsque le lecteur de cartes a fini le transfert du contenu de la carte dans la mémoire centrale, il le signale au superviseur par le biais d'une interruption. Celui-ci peut alors commander la lecture de la carte suivante dans un autre emplacement mémoire. De même, l'imprimante signale au superviseur, par une interruption, la fin de l'impression d'une ligne. Celui-ci peut alors commander l'impression de la ligne suivante si elle est disponible.

Ce mode de fonctionnement attire les remarques suivantes:

- Le temps de réponse est amélioré, puisqu'il n'est plus nécessaire de remplir une bande pour pouvoir la transférer depuis (ou vers) l'ordinateur secondaire d'entrées-sorties.
- La rentabilité du système est améliorée par la récupération au niveau processeur de traitement des temps des opérations d'entrées-sorties devenues autonomes.

EXERCICES D'APPLICATIONS :

1. Décrivez brièvement ce qui se passe, du côté du système 'exploitation, lorsqu'une touche de clavier est pressée
2. Décrivez brièvement comment se fait le transfert d'un bloc de disque vers la mémoire, si le système dispose d'un DMA.
3. Qu'est-ce qu'un bus ?
   a- un programme informatique.
   b- une mémoire spéciale.
   c- un système de communication entre les éléments d'un ordinateur.
4. Citez les différents types de bus.
5. Un lot est composé de 50 travaux, que pour simplifier, on suppose tous constitués de 3 phases :
   Lecture des cartes (20 secondes) ·calcul (15 secondes) ·Impression des résultats (5 secondes).

Le temps mis pour passer d'un travail à un autre est négligeable.

Calculer le temps de traitement total du lot et le taux d'utilisation de l'unité centrale pour le calcul dans les deux cas suivants :

1- L'unité centrale gère les périphériques d'entrée-sortie.
2- Les périphériques sont autonomes et disposent d'un accès direct à la mémoire.

CORRIGÉ DES EXERCICES :

1. Après chaque touche pressée, une interruption (de type matérielle associée au clavier) est générée. Le processeur interrompt son traitement pour lancer la routine d'interruption associée.

2. Le processeur envoie la commande d'E/S au driver du disque. Le driver détaille la commande et la traduit au contrôleur. Le contrôleur prépare les données en copiant les données du disque vers le buffer du disque. Le dispositif DMA envoie les données prépare directement vers la mémoire (sans passer par le processeur). A la fin du transfert, une interruption est générée pour informer le processeur que le transfert est terminé.
3. C
4. les bus de donnée.
   Les bus d'adresses.
   Les bus de contrôle.

5. Durée du traitement = 20+15+5=40s. r=15/40=0,375
   Durée du traitement = 20 (le temps le plus long puisque temps transfert en mémoire =0), r=15/20=0,75

LEÇON N°06 : LE SYSTÈME DE GESTION DE FICHIERS (SGF)

OBJECTIF PÉDAGOGIQUE

À la fin de cette série, le stagiaire doit être capable de **c**onnaître :

- L'organisation en mémoire des fichiers
- La structure des fichiers
- Les différents SGF
- Les différents mécanismes d'allocation et de synchronisations des processus.

PLAN DE LA LEÇON :

INTRODUCTION

I- CATALOGUE EN MÉMOIRE

II- ARCHITECTURE DU SYSTÈME DE FICHIERS

1- Structure de fichiers
2- Méthodes d'accès
3- Contrôle des droits d'accès
4- Verrouillage des fichiers
5- Attribution des blocs

III- LES DIFFÉRENTS TYPES DE SGF

1- Le système de gestion de fichiers FAT
2- Le système de gestion de fichiers NTFS

IV- OUVERTURE ET FERMETURE DES FICHIERS

V- LES MÉCANISMES D'ALLOCATION

VI- LES MÉCANISMES DE SYNCHRONISATION

LEÇON N°06 : LE SYSTÈME DE GESTION DE FICHIERS (SGF)

INTRODUCTION :

Un **système de gestion de fichiers** (SGF) est une structure de données permettant de stocker les informations et de les organiser dans des fichiers sur ce que l'on appelle des mémoires secondaires (disque dur, disquette, CD-ROM, clé USB, disques SSD , etc.). :

Une telle gestion des fichiers permet de traiter et de conserver des quantités importantes de données ainsi que de les partager entre plusieurs programmes informatiques. Il offre à l'utilisateur une vue abstraite sur ses données et permet de les localiser à partir d'un chemin d'accès.

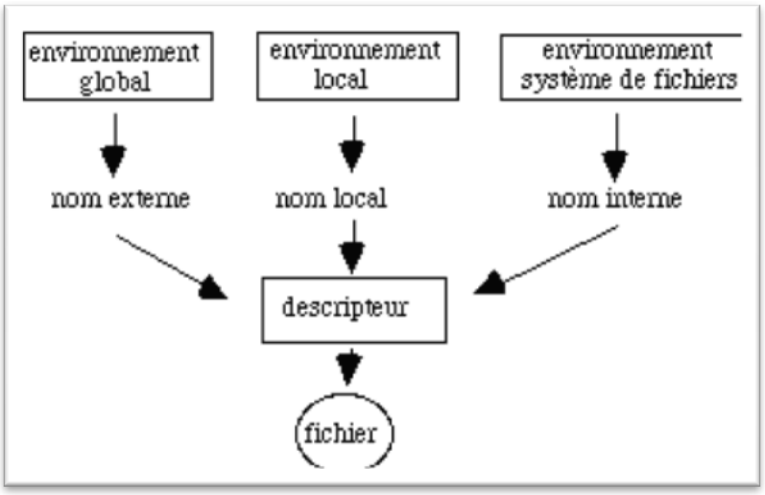
I- CATALOGUE EN MÉMOIRE :

Les fichiers sont regroupés dans des unités logiques appelées système de fichiers. Il correspond à un espace physique qui s'étend sur une partie d'un disque ou plusieurs disques. Pour l'utilisateur il apparaît comme un disque C:, D: ... sous Windows, avec un nom choisi à sa création pour Unix.

Le fichier est l'unité de conservation de l'information. C'est un objet complexe qui n'est pas constitué uniquement des informations visibles par l'utilisateur. Son descripteur contient les informations nécessaires à sa localisation physique sur le disque et à son usage. Le nom de ce descripteur est interne, c'est à dire que l'utilisateur ne le connaît pas. Il n'accède au fichier que par son nom externe. La liaison entre ces deux noms est établie au moyen du catalogue.

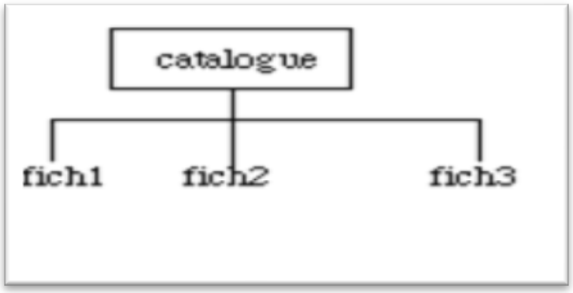
Un fichier peut éventuellement être défini par un nom local à la procédure en cours d'utilisation. C'est ce qui est fait lors de l'appel à la primitive d'ouverture dans un langage de programmation. Cette méthode est plus efficace que l'usage du nom externe car son interprétation est plus rapide et peut apporter une plus grande souplesse dans la gestion du fichier.

Ces différentes possibilités sont résumées dans la figure suivante :

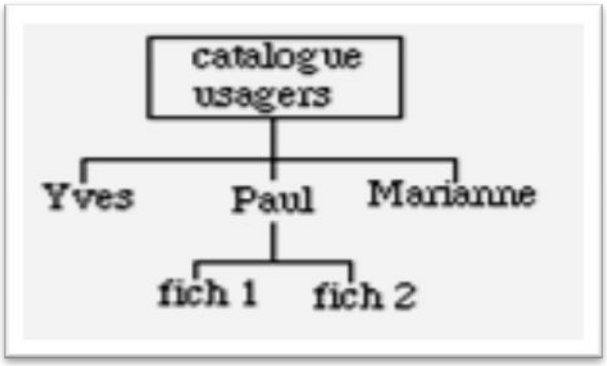


On rencontre le plus souvent trois modèles de catalogue :

- **Organisation en un seul niveau** : on associe directement le descripteur aux noms externes. Il n'y a aucun classement possible aussi ce système n'existe pratiquement plus.

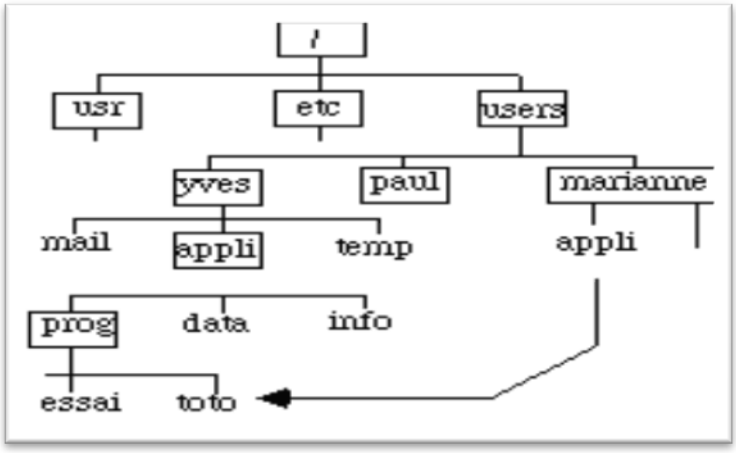


- **Organisation à deux niveaux**: il existe des catalogues secondaires par l'utilisateur. Certains systèmes comme ceux propres à IBM emploient des structures à un niveau qui ressemblent aux structures à deux niveaux. Les règles d'écriture dans le catalogue imposent un nom à plusieurs champs séparés par des points, le premier étant l'identificatif de l'utilisateur. Les mécanismes de sécurité interdisent à une personne de retrouver les éléments du catalogue qui ne correspondent pas à ses propres fichiers, sauf autorisation explicite.



- **Organisation arborescente** :

C'est la plus connue de nos jours puisqu'elle est utilisée par Unix et les systèmes qui s'en sont inspirés (DOS puis Windows...).

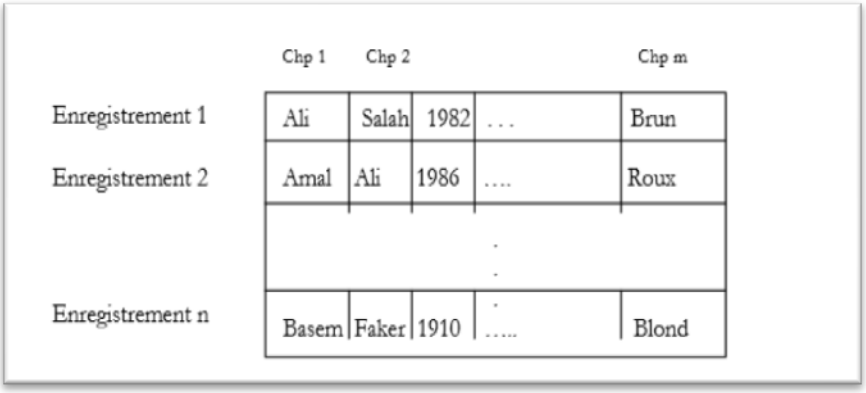


II- ARCHITECTURE DU SYSTÈME DE FICHIERS :

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 dE/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 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.

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-mêmes. 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 l'E/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 d'E/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.

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.

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.

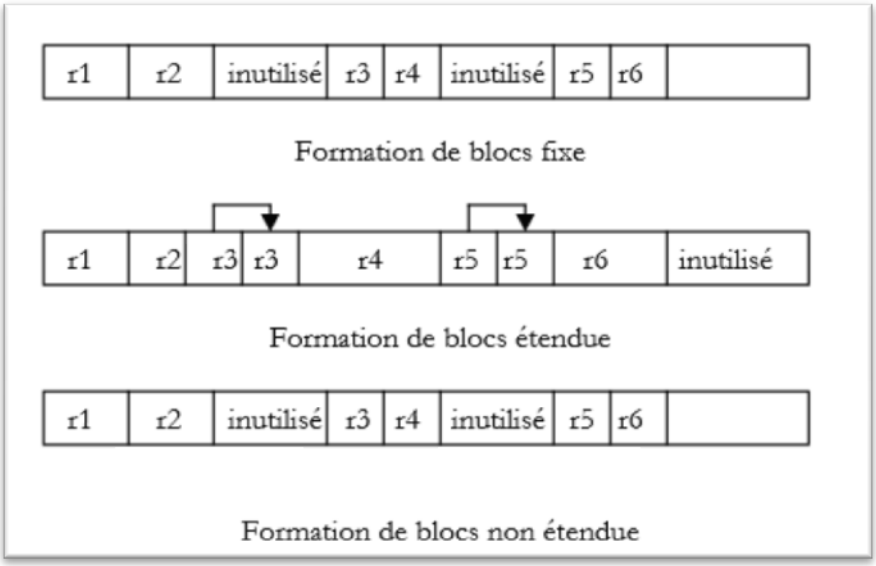
5- Attribution des blocs :

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 d'un enregistrement consiste à additionner la taille de tous les enregistrements qui le précède



III- LES DIFFÉRENTS TYPES DE SGF :

Le SGF est dépendant du système d'exploitation. D'une manière générale, plus le système d'exploitation est récent plus le nombre de systèmes de fichiers supportés sera important.

Système d'exploitation	Type de système de fichiers supportés
DOS	FAT 16
Windows 95	FAT 16, FAT 32 (selon la version de Windows 95)
Windows 98	FAT 16, FAT 32 (selon la version de Windows 98)
Windows NT4	NTFS
Windows 2000 et XP	FAT 32, NTFS
MacOS	HFS
Linux / Unix	EXT2, ReiserFS

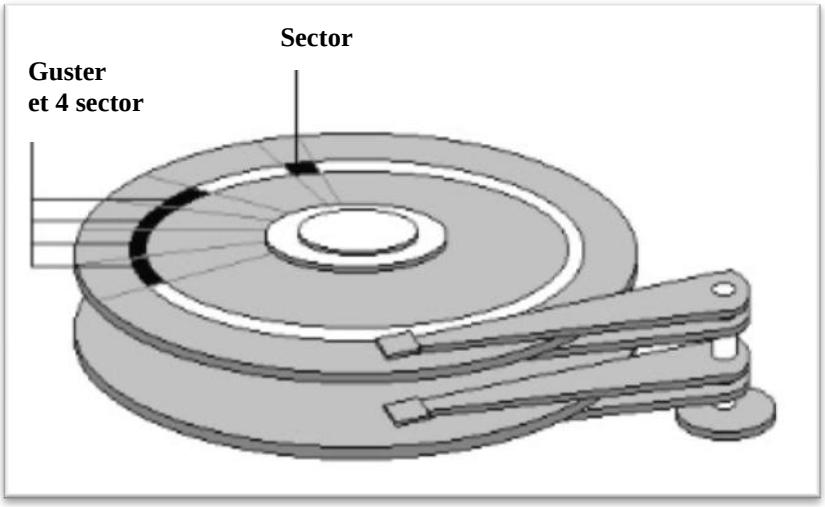
1- Le système de gestion de fichiers FATl :

Il comprend les éléments suivants :

- Un enregistrement d'amorçage principal avec la "Table des partitions" (MBR),
- Une zone réservée au secteur de chargement (Boot Loader),
- Un exemplaire de la FAT,
- Une copie optionnelle de la FAT,
- Le répertoire principal ou répertoire racine,
- La zone des données et sous-répertoires.

Le secteur de démarrage (MBR) (cylindre 0, tête 0 et secteur 1) contient la table de partitions et le code qui, une fois chargé en mémoire, va permettre d'amorcer le système (booter).

Ce programme, une fois en mémoire, va déterminer sur quelle partition le système va s'amorcer, et il va démarrer le programme (appelé boot strap) qui va amorcer le système d'exploitation présent sur cette partition.



D'autre part, c'est ce secteur du disque qui contient toutes les informations relatives au disque dur (fabricant, numéro de série, nombre d'octets par secteur, nombre de secteurs par cluster,…) Ce secteur est le secteur le plus important du disque dur, il sert au set up du BIOS à reconnaître le disque dur. Ainsi, sans celui-ci le disque dur est inutilisable, c'est donc une des cibles préférées des virus.
Ce secteur de démarrage est appelé le boot manager sous Windows NT.

Les répertoires stockent toutes les informations sur chaque fichier qu'ils contiennent :

- Le nom de fichier ;
- La taille du fichier ;
- La date et l'heure de la dernière modification du fichier;
- Les attributs du fichier (lecture seule, archive, …);
- Le numéro du cluster auquel le fichier commence(les autres clusters constitutifs étant retrouvés par la FAT;
- Le répertoire parent (pour les répertoires autre que racine).

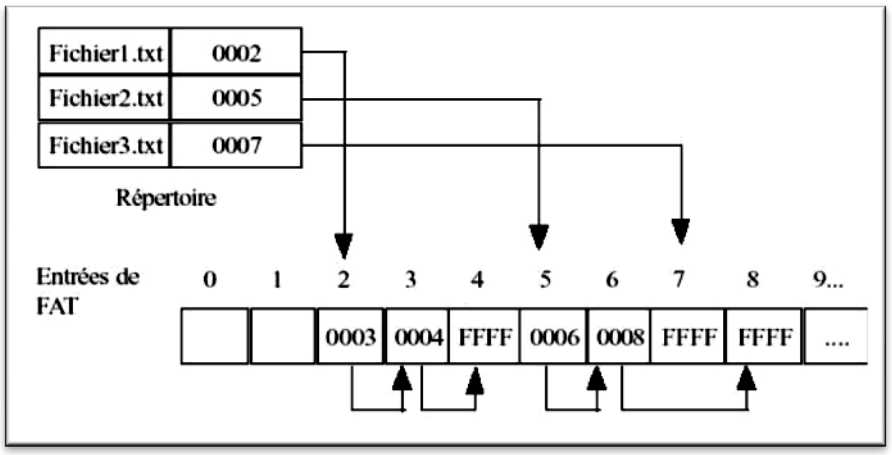
Le système de gestion de fichier FAT utilise un répertoire racine (représenté sur les systèmes d'exploitation qui utilisent ce type de systèmes de fichiers par le signe C:\), qui doit être situé à un endroit spécifique du disque dur.

Ce répertoire racine stocke les informations sur les sous-répertoires et fichiers qu'il contient.

Le système de gestion de fichier FAT est aussi caractérisé par l'utilisation d'une table d'allocation de fichiers et de clusters (ou blocs).

La FAT (File Allocation Table: table d'allocation des fichiers) est le cœur du système de gestion de fichiers. Elle est localisée dans le secteur 1 du cylindre 0 à la tête 1

La Table d'Allocation de Fichiers est une liste de valeurs numériques permettant de décrire l'allocation des clusters d'une partition, c'est-à-dire l'état de chaque cluster de la partition dont elle fait partie.



La table d'allocation est en fait un tableau dont chaque case (ou entrée) correspond à un cluster. Chaque case contient un chiffre qui permet de savoir si le cluster qu'elle représente est utilisé par un fichier, et, le cas échéant, indique l'emplacement du prochain cluster que le fichier occupe.

On obtient donc une chaîne FAT, c'est-à-dire une liste chaînée de références pointant vers les différents clusters successifs, jusqu'au cluster de fin de fichier. Chaque entrée de la FAT a une longueur de 16 ou 32 bits (selon qu'il s'agit d'une FAT16 ou d'une FAT32).

Les deux premières entrées permettent de stocker des informations sur la table elle-même, tandis que les entrées suivantes permettent de référencer les clusters.

Certaines entrées peuvent contenir des valeurs indiquant un état du cluster spécifique. Ainsi la valeur 0000 indique que le cluster n'est pas utilisé, FFF7 permet de marquer le cluster comme défectueux pour éviter de l'utiliser, et les valeurs comprises entre FFF8 et FFFF spécifient que le cluster contient la fin d'un fichier. Chaque partition contient en réalité deux copies de la table, stockées de manière contiguë sur le disque, afin de pouvoir la récupérer si jamais la première copie est corrompue. Derrière la copie de la fat commence le répertoire principal.

2- Le système de gestion de fichiers NTFS :

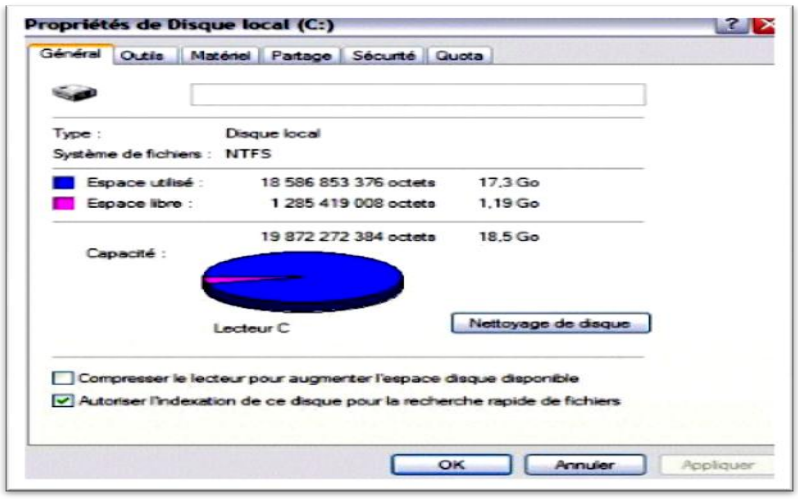
NTFS (New Technology File System) utilise un système basé sur une structure appelée « table de fichiers maître », ou MFT (Master File Table), permettant de contenir des informations détaillées sur les fichiers. Ce système permet ainsi l'utilisation de noms longs, mais, contrairement au système FAT32, il est sensible à la casse, c'est-à-dire qu'il est capable de différencier des noms en majuscules de noms en minuscules.

Pour ce qui est des performances, l'accès aux fichiers sur une partition NTFS est plus rapide que sur une partition de type FAT.

La limite physique d'un disque est de 2To. Les noms des fichiers peuvent comporter jusqu'à 255 caractères.

C'est au niveau de la sécurité que NTFS prend toute son importance, car il permet de définir des attributs de sécurité pour chaque fichier (droits associés aux utilisateurs …).

La version 5 de ce système de fichiers (en standard sous Windows 2000) amène encore de nouvelles fonctionnalités parmi lesquelles des performances accrues, des quotas de disque par volume définis pour chaque utilisateur.



La table des fichiers maîtres : MFT

La FAT est un tableau de valeurs numériques dont chaque case permet de décrire l'allocation des clusters d'une partition, c'est-à-dire l'état (l'occupation ou non par un fichier) de chaque cluster de la partition dont elle fait partie.

Le système de fichiers NTFS est basé sur une structure différente, appelée table de fichiers maître, contenant des enregistrements sur les fichiers et les répertoires de la partition.

Le premier enregistrement, appelé descripteur, contient des informations sur la MFT (une copie de celui-ci est stockée dans le second enregistrement).

Le troisième enregistrement contient le fichier journal, un fichier qui contient es actions effectuées sur la partition.

Les enregistrements suivants, constituant ce que l'on nomme le noyau, référencent chaque fichier et répertoire de la partition sous forme d'objets affectés d'attributs.

Cela signifie que les informations relatives à chaque fichier sont stockées dans le fichier, qui est lui même enregistré au sein de la MFT. La MFT représente donc une structure de stockage des données de la partition, et non une liste de clusters.

IV- OUVERTURE ET FERMETURE DES FICHIERS :

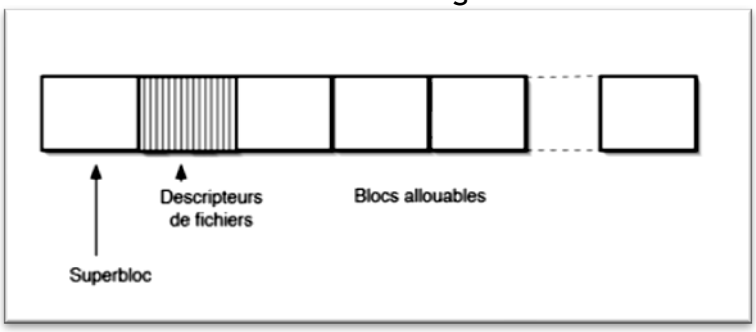
À l'ouverture d'un fichier par un processus, l'appelant fournit un nom symbolique absolu ou relatif et l'appel système lui retourne un numéro de descripteur de fichier ouvert ou un code d'erreur. Le numéro de descripteur sera utilisé comme désignation locale au processus pour toute la phase pendant laquelle le fichier reste ouvert.

C'est à l'ouverture que sont effectuées toutes les vérifications nécessaires (nom correspondant à un fichier inexistant par exemple) et que sont construites les différentes tables permettant de réaliser les accès. L'appel système correspondant est

ouvrir (nom_fich,mode) ->i_dfo

Lorsque l'ouverture se passe bien, ouvrir retourne un indice de descripteur de fichier ouvert.

En cas d'erreur, ouvrir retourne un code d'erreur négatif avec le sens suivant **:**



1- On devrait avoir un catalogue et on traverse un fichier,
2- Nom inexistant dans un catalogue,
3- Trop de fichiers ouverts dans le système,
4- Trop de fichiers ouverts par le processus.

L'appel système de fermeture reçoit en paramètre un numéro de descripteur de fichier ouvert.

Les tables construites à l'ouverture sont détruites. L'appel système est :

Fermer(i_dfo) -> code de retour

La procédure retourne :

- 0 si l'opération s'est bien exécuté,
- 1 si le descripteur est invalide,
- 2 s'il ne correspond pas à un fichier ouvert.

V- LES MÉCANISMES D'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.

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.

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 Nème 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.

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.

VI- LES MÉCANISMES DE SYNCHRONISATION :

On peut les classer en deux catégories :

1- Les mécanismes simples qui se limitent à des échanges de signaux de marche et d'arrêt, analogues à des feux rouges. Ces signaux sont divers : sémaphores, signaux Unix...

2- Des mécanismes plus complexes qui permettent également d'échanger des informations ou messages.

1- Synchronisation par moniteur et sémaphores :

a- Synchronisation par moniteur :

Le moniteur est le cœur du système d'exploitation. Il est constitué d'un ensemble de procédures et de variables d'état utilisées par ces procédures. Le moniteur représente le cœur du système d'exploitation. Il échappe aux règles habituelles des processus car il est toujours présent et son rôle est d'ordonnancer leur fonctionnement.

Certaines de ses variables sont accessibles à l'utilisateur grâce à des bibliothèques de fonctions spéciales au travers de points d'entrée du moniteur. Les processus externes au moniteur peuvent interroger et utiliser ces variables mais elles ne peuvent pas les modifier car il est fort probable que cela perturberait le fonctionnement du moniteur donc de l'ordinateur.

Parmi ces fonctions il en existe qui permettent de bloquer ou de réveiller les processus écrits par les utilisateurs. Le blocage et le réveil s'effectuent au moyen d'une condition **c** utilisable dans les trois opérations suivantes:

- **attendre(c):** Bloque le processus p qui l'utilise et le place en attente de l'événement c.
- **vide(c) :** Retourne vrai s'il n'existe pas de processus en attente de c, faux sinon.
- **Activer(c) :** Réveille le premier processus en attente de l'événement c.

On notera que la fonction attendre(c) suppose l'existence d'une file d'attente associée. Le processus réveillé reprend son activité à l'instruction qui suit le point d'arrêt.

Une caractéristique essentielle de la synchronisation par moniteur : les processus consultent l'état de variables qu'ils ne peuvent modifier en aucun cas. C'est donc un moyen assez frustre car les communications sont réduites au minimum mais simple à réaliser. Ces mécanismes existent à des degrés divers dans tous les systèmes d'exploitation ; il est possible de bloquer le père au moyen d'un appel à une fonction wait().

b- Synchronisation par sémaphore :

La synchronisation par sémaphore ou flag est un moyen simple et ancien de synchroniser des processus parallèles. A la différence de la synchronisation par le moniteur le programmeur a accès aux états de ce drapeau et peut le manipuler. Le principe est directement hérité des chemins de fer d'où son nom : lorsque le sémaphore est levé, le processus P peut continuer

son chemin; lorsqu'il est baissé il doit attendre jusqu'à ce qu'un autre processus Q le lève. P et Q sont l'équivalent de deux trains roulant sur deux voies distinctes qui doivent se synchroniser pour pouvoir franchir un croisement sans accident.

Dans le modèle le plus simple il existe trois primitives:

- **Lever (c) :** fait passer le sémaphore c de la valeur "baissé" à "levé".
- **Baisser(c) :** fait passer le sémaphore c de "levé" à "baissé".

- **Flag(c) :** retourne vrai si le sémaphore c est baissé.

Les sémaphores sont des variables communes mises à la disposition des différents processus par le système d'exploitation qui peuvent les consulter, les modifier et sur lesquelles ils peuvent se mettre en attente.

Pour l'illustrer voici un exemple du transfert de résultats au moyen d'un tampon:

Processus P	**Processus Q**

baisser(e);
while (calculs à faire){
calculs des éléments du tampon (a);
baisser(e);
if (flag(f)) {
attendre(f);
}
écrire tampon (a);
lever(e);
	lever(f);
	while (éléments à transférer) {
	if (flag(e)) {
	attendre(e)
	}
	baisser(f);
	lire tampon(a);
	lever(f);
	}

e et f sont des variables qui pouvaient être communes à deux processus distincts.

À la différence des threads les segments de données sont disjoints donc il faut imaginer un moyen de partager ces informations. Les sémaphores sont une solution pour résoudre ce problème. Ils font partie du contexte commun à l'ensemble des processus. Ces variables ne peuvent prendre que deux valeurs. Elles doivent être déclarées au moyen d'instructions spéciales qui précisent les noms des processus qui peuvent les partager. Les sémaphores sont un moyen simple de synchronisation qui, cependant, est en désuétude car il ne permet qu'un partage pauvre (binaire) d'informations.
On préfère aujourd'hui une synchronisation par messages qui, comme son nom l'indique, permet d'échanger des informations.

2- Synchronisation par messages :

❖ Principes généraux :

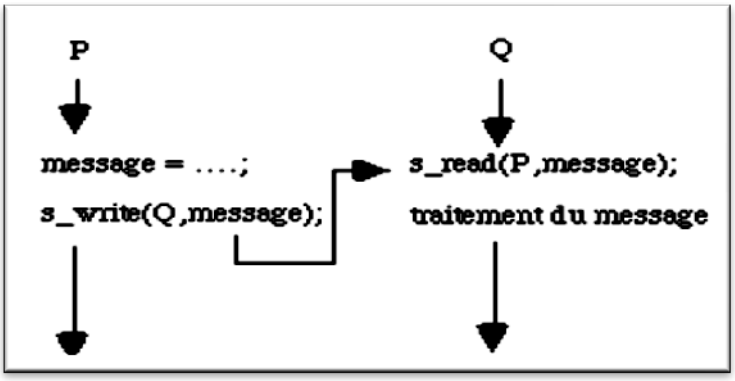
Le sémaphore est un moyen de communication frustre entre processus car limité à un message binaire : on passe ou on ne passe pas. On peut imaginer des primitives plus riches qui permettent la synchronisation en échangeant simultanément des informations appelées messages. Les primitives s'apparentent à des ordres de lecture et d'écriture :

- s_read (id,message) : met le processus en attente d'une lecture jusqu'à ce qu'un processus de nom id envoie un message. On l'appelle une lecture bloquante.

- s_write (id,message) : envoie au processus de nom id un message. Cette fonction n'est pas bloquante. Le processus qui l'utilise continue son exécution même si le processus receveur n'a pas lu le message. Celui-ci est stocké dans un tampon. Si plusieurs messages sont envoyés successivement à ce processus, leur ordre de lecture sera l'ordre d'émission des messages.

La synchronisation par messages est très utilisée dans les systèmes d'exploitation modernes car elle permet, grâce à l'information échangée, d'envisager les traitements les plus variés. Dans la réalité il existe de nombreuses autres primitives qui permettent de réaliser des fonctions complexes.

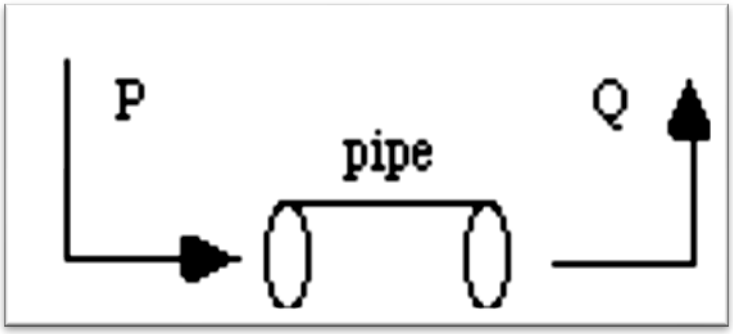
La synchronisation par message trouve aujourd'hui un usage encore plus large. Ainsi PVM et MPI sont des bibliothèques qui permettent de réaliser les fonctions d'échange, de synchronisation et de communication entre processus qui sont exécutés sur des machines en réseau. PVM et MPI sont employés sur les machines massivement parallèles.



Unix connaît un mécanisme de communication entre processus, appelé pipe, qui est un exemple de synchronisation possible par messages. Un premier processus P écrit dans le pipe, un deuxième, Q, le lit. La file d'attente correspond au modèle FIFO

Un fichier partagé est une autre méthode, très simple et efficace, pour faire communiquer et synchroniser deux processus: le premier crée et écrit dans un fichier, le deuxième en lit le contenu. On peut également utiliser l'existence ou la non-existence d'un fichier en guise de sémaphore. Ceci est souvent utilisé dans les systèmes Unix où l'on voit souvent se créer de nombreux fichiers dont l'existence est très temporaire, simplement pour permettre à un ensemble de processus de synchroniser leur activité et de communiquer l'information indispensable au bon déroulement de la fonction pour laquelle ils ont été créés.

Il se peut parfois, parce qu'un processus ne fonctionne pas correctement, que ce fichier ne soit pas détruit après usage. Il suffit alors de le détruire manuellement pour retrouver un fonctionnement correct. La synchronisation par fichiers est peu performante car les opérations de lecture et d'écriture sont lentes. Elle ne peut donc pas être employée lorsqu'on a besoin de temps de réaction courts.



EXERCICES D'APPLICATION :

1- Quelles sont les commandes qui permettent d'ouvrir et de fermer des fichiers?

2- Citez les différentes méthodes de synchronisation des processus ?

3- Qu'est ce qu'un cluster?

4- Comment se nomme l'unité minimale allouée par un disque dur lors d'une opération d'écriture ?

b- Le Secteur.
c- Le Cluster.
d- La FAT.
e- Le Block.

5- À quoi sert un système de fichier ?

a- Il permet de stocker les informations et de les organiser sur la mémoire cache.

b- Il permet de stocker les informations et de les organiser sur la mémoire vive.

c- Il permet de stocker les informations et de les organiser sur les mémoires secondaires.

CORRECTIONS DES EXERCICES :

1 et 2 revoir le cours

- 3.
- Un cluster (ou unité d'allocation » ou bloc) est la plus petite unité de disque que le système d'exploitation est capable de gérer.
- Un fichier occupe un nombre entier de cluster.
- Un petit fichier, aussi petit soit il, occupe tout un cluster. Un fichier peut occuper plusieurs clusters, mais le dernier cluster ne sera pas rempli:
- Il y a un gaspillage de l'espace disque, d'autant plus grand que la taille des clusters est grande.
- Plus la taille d'un cluster est importante, moins le système d'exploitation aura d'entités à gérer et plus il sera rapide...
- En contrepartie, étant donné qu'un système d'exploitation ne sait gérer que des unités d'allocation entière, c'est-a dire qu'un fichier occupe un nombre entier de cluster, le gaspillage d'espace disque est d'autant plus grand que le secteur est grand.
- Il faut choisir une taille de cluster optimale pour ne pas trop ralentir l'OS et ne pas trop gaspiller d'espace disque.

4- b

5- c