République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique
Université de Tissemsilt
Faculté des Sciences et de la Technologie
Département des Mathématiques et Informatique

Système exploitation I

Cours et Exercices Corrigés

Destiné aux etudiants de 2emme année license Informatique

Dr HAMEURLAINE Messaoud

2023/2024

Avant-propos

L'objectif de ce polycopié intitulé "Systèmes d'Exploitation 1, Partie 1 (Cours et exercices corrigés)" est de fournir un support pédagogique destiné aux étudiants de deuxième année en licence d'informatique pour les accompagner dans l'apprentissage de la première partie du cours "Systèmes d'Exploitation". Ce document est aligné avec le programme du socle commun et présente les principes fondamentaux du fonctionnement des systèmes d'exploitation. Il vise à aider les étudiants à comprendre des concepts clés tels que la gestion des processus et de la mémoire.

Ce polycopié s'adresse aussi bien aux étudiants qu'aux enseignants chargés des cours et travaux dirigés (TD). Cette première partie est structurée en chapitres qui regroupent le contenu théorique et des exercices corrigés pour renforcer l'apprentissage. Une deuxième partie est en préparation et sera présentée sous forme de tutoriel. Elle portera principalement sur le système Unix et inclura des fiches de travaux pratiques pour approfondir la compréhension des concepts étudiés.

Ce document pédagogique est le fruit d'années d'expérience dans la formation des étudiants, ainsi que d'une synthèse des ressources pédagogiques de mes collègues et des ouvrages scientifiques cités en bibliographie. Je tiens à exprimer ma gratitude envers mes collègues et les auteurs des livres dont j'ai utilisé leurs ouvrages pour enrichir ce travail. Mon apport a consisté à sélectionner les informations les plus pertinentes, à les reformuler pour les rendre accessibles aux étudiants, et à y intégrer une dimension pédagogique personnelle. Certaines figures ont été reproduites en fonction de leur utilité pédagogique. Je souhaite également autoriser tout enseignant à réutiliser ou reproduire ce document à des fins éducatives.

Enfin, j'espère que ce polycopié constituera une ressource précieuse et un soutien pédagogique pour les enseignants et les étudiants dans leur formation en informatique.

Table des matières

CHAPITRE 2 Gestion des Processus

Introduction Générale

Les systèmes d'exploitation constituent le cœur des systèmes informatiques. Ils assurent la gestion efficace des processus, de la mémoire, du stockage, et des périphériques d'entrée/sortie, tout en garantissant la sécurité et l'interface utilisateur. Comprendre comment fonctionnent les systèmes d'exploitation est essentiel pour tout étudiant en informatique, car ces connaissances sont la base sur laquelle repose la compréhension des technologies modernes. Ce polycopié vise à fournir une compréhension approfondie des systèmes d'exploitation, un domaine fondamental en informatique qui joue un rôle clé dans la gestion des ressources matérielles et logicielles d'un ordinateur. Ce document a été conçu en accord avec le programme du socle commun, afin de répondre aux exigences académiques et d'offrir un apprentissage structuré et pratique des concepts des systèmes d'exploitation.

Ce polycopié a pour principaux objectifs de :

1. Introduire les concepts fondamentaux des systèmes d'exploitation : Permettre aux étudiants de saisir le rôle d'un système d'exploitation, ses fonctions et sa structure, ainsi que ses interactions avec les utilisateurs et le matériel. Ce point de départ est essentiel pour développer une vue globale du fonctionnement des systèmes informatiques.
2. Développer des compétences en gestion des processus et de la mémoire : Les étudiants apprendront à comprendre les mécanismes internes qui permettent aux systèmes d'exploitation de gérer efficacement les processus en cours d'exécution et d'optimiser l'utilisation de la mémoire centrale. Ces compétences sont cruciales pour appréhender les défis liés à la performance et à la stabilité des systèmes.
3. Fournir des exercices pratiques pour appliquer les concepts théoriques : Chaque chapitre est accompagné d'exercices corrigés qui offrent aux étudiants l'opportunité de tester et de consolider leurs connaissances à travers des problèmes concrets. Cette approche renforce l'apprentissage et prépare les étudiants à des situations réelles.

Ce document est organisé comme suit :

Le premier chapitre pose les bases en introduisant les concepts clés des systèmes d'exploitation. Il propose une exploration détaillée de la vue utilisateur et de la vue système, exposant les différentes perspectives d'un système d'exploitation. L'organisation du système informatique est décrite en mettant en avant les structures de stockage, les entrées/sorties

(E/S), et les mécanismes d'interruption. Ce chapitre aborde également les architectures des systèmes, distinguant les systèmes à processeur unique des systèmes multiprocesseurs. Il termine par une analyse des objectifs et fonctions des systèmes d'exploitation, offrant aux étudiants une vision globale de leur rôle central en tant que machine virtuelle, gestionnaire de ressources et mécanisme de sécurité.

Le deuxième chapitre se penche sur la gestion des processus, un aspect fondamental des systèmes d'exploitation. Il couvre en détail les concepts de processus, y compris leurs caractéristiques, le contexte d'exécution, et les états qu'un processus peut traverser. Ce chapitre présente également le mécanisme de commutation de contexte et les interruptions, qui permettent au système de gérer efficacement les tâches. Une attention particulière est accordée aux différents algorithmes d'ordonnancement (scheduling), tels que le FCFS, SJF, Round Robin, et les algorithmes de scheduling par priorité. En étudiant ces stratégies, les étudiants découvriront comment les systèmes d'exploitation organisent et contrôlent l'exécution des processus pour maximiser l'efficacité et l'équité du processeur.

Le troisième chapitre explore les techniques de gestion de la mémoire centrale, un élément crucial pour le fonctionnement efficace des systèmes d'exploitation. Il couvre les méthodes de gestion de la mémoire dans les environnements monoprogrammés et multiprogrammés, comme le swapping et les techniques de recouvrement. Le chapitre détaille également les approches d'allocation de mémoire, qu'elles soient contiguës (partitions fixes et variables) ou non contiguës (pagination, segmentation, segmentation paginée). Enfin, il présente la mémoire virtuelle, une technologie avancée permettant aux systèmes d'exploitation d'étendre l'utilisation de la mémoire physique en exploitant l'espace de stockage. Ce chapitre vise à fournir aux étudiants les outils nécessaires pour comprendre comment les systèmes gèrent la mémoire afin d'optimiser les performances et la sécurité des processus.

Ce polycopié est conçu pour être un guide complet qui accompagne les étudiants à travers les concepts théoriques et les défis pratiques des systèmes d'exploitation. En progressant à travers les chapitres, les étudiants acquerront non seulement des connaissances solides sur le fonctionnement des systèmes d'exploitation, mais aussi la capacité d'appliquer ces concepts dans des scénarios concrets. Ce document aspire à être un outil pédagogique utile pour les enseignants et un compagnon d'étude précieux pour les étudiants, renforçant ainsi leur parcours académique en informatique.

CHAPITRE 1
Introduction aux Systèmes d'Exploitation

I. Introduction

Dès le début, les pionniers de l'informatique avaient une vision claire : utiliser les machines pour simplifier et améliorer la vie humaine en automatisant une large gamme de tâches. Lorsque les premiers ordinateurs ont été développés dans les années 1940, ils étaient des machines complexes que seuls quelques individus — ingénieurs, mathématiciens et scientifiques — avaient l'expertise nécessaire pour concevoir, construire et exploiter. Cela faisait de l'informatique un domaine exclusif, confiné aux institutions de recherche, aux applications militaires et aux grandes entreprises disposant des ressources pour investir dans cette technologie de pointe.

À cette époque, les ordinateurs étaient immenses, coûteux et difficiles à programmer. Ils étaient principalement utilisés pour des tâches spécialisées, telles que le décryptage de codes pendant la Seconde Guerre mondiale ou l'exécution de calculs mathématiques complexes. Cependant, la vision de ces premiers scientifiques en informatique était toujours plus large. Ils imaginaient un futur où les ordinateurs pourraient être utilisés par des personnes ordinaires pour des tâches quotidiennes, allant de la gestion des finances personnelles au jeu vidéo.

Au fil des décennies, les avancées technologiques ont commencé à réduire la taille, le coût et la complexité des ordinateurs. Dans les années 1980, cette évolution a atteint un point culminant avec l'avènement de l'ordinateur personnel (PC). Un élément clé de cette transformation a été l'introduction de logiciels plus conviviaux qui servaient d'intermédiaires entre l'utilisateur et le matériel complexe de la machine. Ces logiciels permettaient aux utilisateurs d'interagir avec l'ordinateur sans avoir besoin de comprendre les détails complexes de son fonctionnement interne.

Au cœur de cette expérience conviviale se trouve le système d'exploitation (OS). Sans OS, un ordinateur n'est guère plus qu'un ensemble de composants électroniques — processeurs, mémoire, dispositifs de stockage — qui ne peuvent accomplir aucune fonction utile par eux-mêmes. L'OS est le logiciel qui insuffle la vie à ce matériel, lui permettant de stocker, traiter et récupérer des données, ainsi que d'exécuter une multitude d'autres tâches.

Les logiciels sont généralement classés en deux catégories : les logiciels système et les logiciels applicatifs. Le logiciel système, dont l'OS est le noyau, est chargé de gérer les ressources de l'ordinateur, telles que le processeur, la mémoire et le stockage. Il fournit un environnement stable dans lequel les logiciels applicatifs peuvent fonctionner. Les logiciels applicatifs, quant à eux, sont conçus pour aider les utilisateurs à accomplir des tâches spécifiques, comme rédiger un document, naviguer sur Internet ou retoucher une photo.

Le développement des systèmes d'exploitation a été une étape cruciale pour rendre les ordinateurs plus accessibles. Avant leur avènement, les programmeurs devaient écrire du code qui communiquait directement avec le matériel, un processus non seulement chronophage mais aussi sujet aux erreurs. Les systèmes d'exploitation ont permis d'abstraire cette complexité, en présentant aux programmeurs et aux utilisateurs une interface simplifiée et standardisée. Cette "machine virtuelle" a facilité le développement et l'utilisation des logiciels, favorisant ainsi l'explosion de l'informatique personnelle à la fin du XXe siècle.

Aujourd'hui, les systèmes d'exploitation sont omniprésents, fonctionnant sur tout, des ordinateurs de bureau et des smartphones aux serveurs et aux systèmes embarqués. Ils gèrent les ressources matérielles, appliquent des mesures de sécurité et fournissent une plateforme pour les logiciels applicatifs, tout en dissimulant la complexité sous-jacente de la machine à l'utilisateur. Cette évolution reflète l'ambition initiale des pionniers de l'informatique : créer des machines non seulement puissantes, mais aussi accessibles et utiles à tous.

L'objectif du chapitre d'introduction aux systèmes d'exploitation est de fournir une compréhension fondamentale de ce qu'est un système d'exploitation, de son rôle essentiel dans la gestion des ressources matérielles et logicielles d'un ordinateur, et de son importance pour faciliter l'interaction entre l'utilisateur et la machine. Ce chapitre explore les concepts clés, l'évolution des systèmes d'exploitation, ainsi que leur impact sur l'accessibilité et l'efficacité des ordinateurs, préparant ainsi le lecteur à approfondir ses connaissances sur le fonctionnement et les différentes fonctions des systèmes d'exploitation.

II. Système informatique

Un système informatique peut être globalement divisé en quatre composants principaux : le matériel, le système d'exploitation, les programmes d'application et l'utilisateur, comme illustré en Figure 1.

Le matériel, composé de l'unité centrale de traitement (CPU), de la mémoire et des périphériques d'entrée/sortie (E/S), constitue les ressources informatiques fondamentales du

système. Les programmes d'application, tels que les traitements de texte, les tableurs, les compilateurs et les navigateurs Web, déterminent comment ces ressources sont exploitées pour résoudre les problèmes informatiques des utilisateurs. Le système d'exploitation gère le matériel et coordonne son utilisation entre les différents programmes d'application pour divers utilisateurs.

Un système informatique peut également être perçu comme un ensemble de matériel, de logiciels et de données. Le système d'exploitation assure une utilisation optimale de ces ressources pour le bon fonctionnement de l'ensemble.

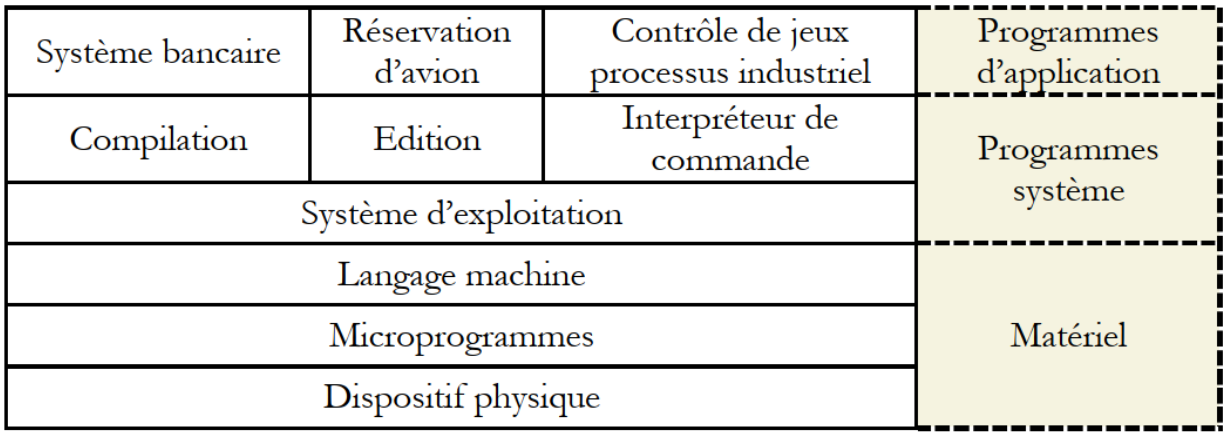


Figure 1 Présentation d'un système informatique

Pour mieux comprendre le rôle du système d'exploitation, nous allons maintenant l'examiner sous deux angles : celui de l'utilisateur et celui du système lui-même.

Le matériel (l'unité centrale de traitement (UC), la mémoire et les périphériques d'entrée/sortie (E/S)) fournit les ressources informatiques de base du système. Les programmes d'application, tels que les traitements de texte, les tableurs, les compilateurs et les navigateurs Web, définissent la manière dont ces ressources sont utilisées pour résoudre les problèmes informatiques des utilisateurs. Le système d'exploitation contrôle le matériel et coordonne son utilisation entre les différents programmes d'application pour les différents utilisateurs.

On peut également considérer qu'un système informatique est constitué de matériel, de logiciels et de données. Le système d'exploitation fournit les moyens d'utiliser correctement ces ressources dans le fonctionnement du système informatique.

Pour mieux comprendre le rôle du système d'exploitation, nous allons maintenant explorer les systèmes d'exploitation de deux points de vue : celui de l'utilisateur et celui du système.

IV.1 Vue utilisateur

La perception qu'un utilisateur a d'un ordinateur dépend de l'interface qu'il utilise. Beaucoup d'utilisateurs interagissent avec un ordinateur portable ou un PC composé d'un écran, d'un clavier et d'une souris. Ces systèmes sont conçus pour être utilisés principalement par une seule personne, qui monopolise les ressources de la machine. Dans ce contexte, le système d'exploitation est orienté vers la facilité d'utilisation, avec une certaine attention aux performances et à la sécurité, mais sans se soucier particulièrement de la gestion des ressources partagées entre plusieurs utilisateurs.

De plus en plus, les utilisateurs se tournent vers des appareils mobiles comme les smartphones et les tablettes, qui remplacent souvent les ordinateurs de bureau et les ordinateurs portables. Ces appareils sont généralement connectés à des réseaux via des technologies cellulaires ou sans fil, et l'interaction avec le système se fait principalement via un écran tactile, où l'utilisateur touche et glisse ses doigts pour naviguer, plutôt que d'utiliser un clavier physique et une souris. De plus, de nombreux appareils mobiles offrent la possibilité d'interagir par commande vocale, comme avec Siri d'Apple.

Il existe aussi des ordinateurs qui sont pratiquement invisibles pour l'utilisateur. Par exemple, les ordinateurs embarqués dans les appareils domestiques et les automobiles peuvent avoir des claviers numériques et des indicateurs lumineux pour afficher leur état, mais ils sont principalement conçus pour fonctionner automatiquement, sans intervention directe de l'utilisateur. Ces systèmes d'exploitation et applications sont donc orientés vers une utilisation autonome, sans nécessiter une interaction humaine régulière.

IV.2 Vue système

Du point de vue de l'ordinateur, le système d'exploitation est le programme le plus étroitement connecté au matériel. Dans cette optique, on peut envisager un système d'exploitation comme un gestionnaire de ressources. Un système informatique dispose de diverses ressources essentielles à la résolution de problèmes, telles que le temps processeur (CPU), l'espace mémoire, l'espace de stockage et les périphériques d'entrée/sortie (E/S). Le rôle du système d'exploitation est de gérer ces ressources. Face à des demandes multiples et parfois conflictuelles, il doit déterminer comment les allouer efficacement et équitablement aux programmes et utilisateurs pour assurer le bon fonctionnement du système.

Une autre perspective sur le système d'exploitation insiste sur son rôle de contrôle des périphériques d'E/S et des programmes utilisateur. Dans ce cadre, le système d'exploitation agit comme un programme de contrôle, supervisant l'exécution des programmes pour prévenir les erreurs et les abus dans l'utilisation de l'ordinateur. Il se concentre

particulièrement sur la gestion et le contrôle des périphériques d'E/S pour garantir une utilisation correcte et sécurisée du système.

V. Organisation du système informatique

Un système informatique moderne à usage général se compose d'un ou plusieurs processeurs et de divers contrôleurs de périphériques, tous connectés via un BUS (Voir Figure 2). Ce BUS facilite l'accès entre les périphériques et la mémoire partagée. Selon le contrôleur, plusieurs périphériques peuvent être reliés. Chaque contrôleur de périphérique gère une mémoire tampon locale ainsi qu'un ensemble de registres spécialisés. Il est responsable du transfert des données entre les périphériques qu'il contrôle et sa mémoire tampon locale.

En général, les systèmes d'exploitation incluent un pilote de périphérique pour chaque contrôleur. Ce pilote, qui intègre le contrôleur de périphérique, offre au reste du système d'exploitation une interface uniforme pour interagir avec le périphérique. La CPU et les contrôleurs de périphériques peuvent fonctionner en parallèle, partageant les cycles de mémoire. Pour assurer un accès ordonné à la mémoire partagée, un contrôleur de mémoire est chargé de synchroniser cet accès, garantissant ainsi une gestion efficace et ordonnée des ressources.

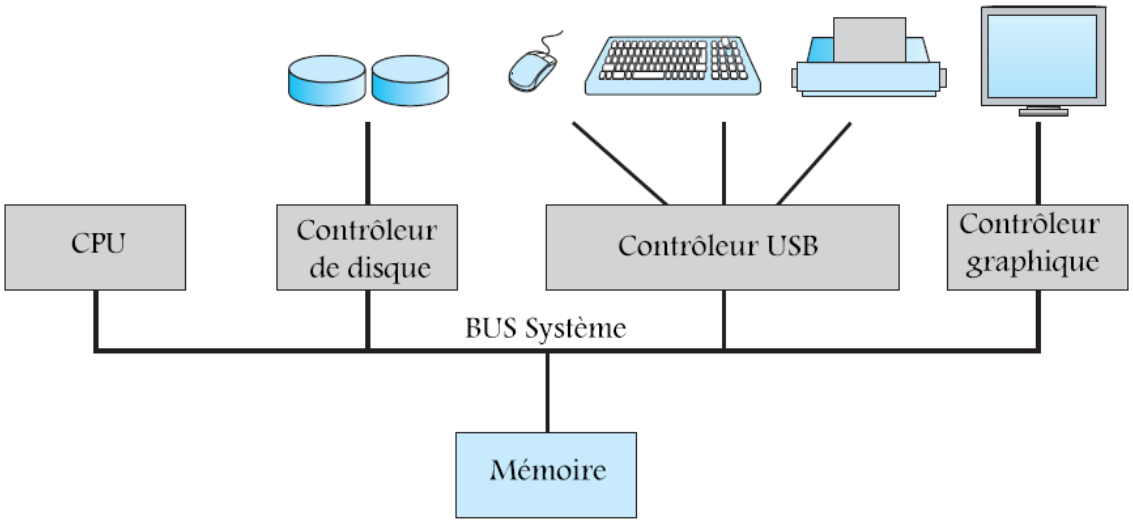


Figure 2 Un système informatique PC typique

VI.1 Structure de stockage

La CPU ne peut exécuter des instructions qu'à partir de la mémoire, ce qui signifie que tout programme doit d'abord être chargé en mémoire avant de pouvoir s'exécuter. Les ordinateurs à usage général exécutent la plupart de leurs programmes à partir de la mémoire réinscriptible, également appelée mémoire principale ou mémoire à accès aléatoire (RAM).

La mémoire est organisée en un tableau d'octets, chacun ayant une adresse unique. Les interactions avec la mémoire se font par des séquences d'instructions de chargement (load) ou de stockage (store) vers des adresses mémoire spécifiques :

- ✓ L'instruction de chargement transfère un octet ou un mot de la mémoire principale vers un registre interne de la CPU.
- ✓ L'instruction de stockage déplace le contenu d'un registre vers la mémoire principale.

Idéalement, nous souhaiterions que les programmes et les données résident en permanence dans la mémoire principale. Cependant, cette configuration est rarement possible pour deux raisons principales :

- ✓ La mémoire principale est souvent trop petite pour contenir tous les programmes et données nécessaires de manière permanente.
- ✓ La mémoire principale est volatile, ce qui signifie qu'elle perd son contenu lorsque l'alimentation est interrompue.

Par conséquent, la plupart des systèmes informatiques utilisent un stockage secondaire comme extension de la mémoire principale. Le stockage secondaire doit répondre à l'exigence principale d'offrir une capacité importante pour conserver de grandes quantités de données de manière permanente.

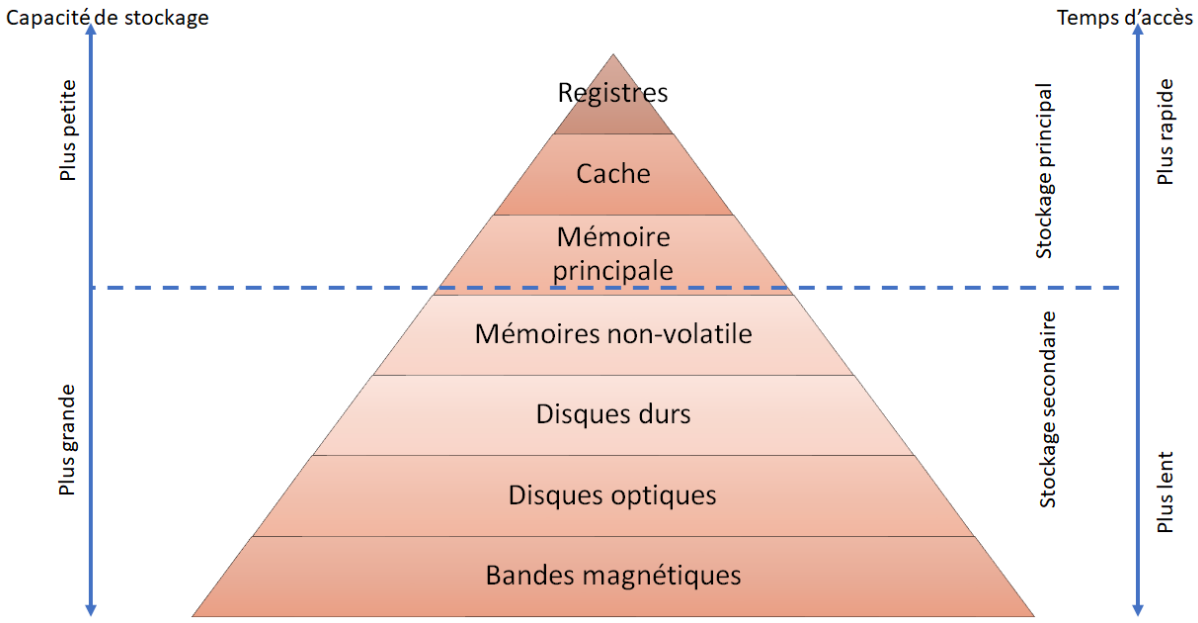


Figure 3 Hiérarchie des périphériques de stockage.

VI.2 Structure des entrées/sorties (E/S)

Une grande partie du code du système d'exploitation est dédiée à la gestion des entrées/sorties (E/S), en raison de son rôle crucial pour la fiabilité et les performances du système, ainsi que de la diversité des périphériques.

Lorsque le contrôleur de périphérique est configuré avec les tampons, pointeurs et compteurs nécessaires, il peut transférer un bloc complet de données directement entre le périphérique et la mémoire principale sans nécessiter l'intervention de la CPU. Ce processus génère une seule interruption par bloc de données pour notifier au pilote de périphérique que l'opération est terminée, contrairement aux périphériques à faible vitesse qui génèrent une interruption pour chaque octet. Pendant que le contrôleur de périphérique gère ces transferts, la CPU reste disponible pour exécuter d'autres tâches.

Certains systèmes informatiques utilisent une architecture de commutation au lieu d'un bus partagé. Dans ces systèmes, plusieurs composants peuvent communiquer simultanément entre eux, éliminant ainsi les conflits pour les cycles d'accès qui se produisent sur un bus partagé.

VI.3 Les interruptions

Les interruptions sont des signaux utilisés pour informer la CPU d'événements nécessitant une attention immédiate. Elles jouent un rôle crucial dans l'interaction entre les systèmes d'exploitation et le matériel. Lorsqu'une interruption se produit, elle doit transférer le contrôle à une routine de service d'interruption spécifique. Pour gérer ce transfert efficacement, plutôt que d'utiliser une routine générique pour traiter toutes les interruptions, un vecteur d'interruption, qui est une table de pointeurs vers les routines de traitement des interruptions, est souvent employé pour offrir la rapidité nécessaire.

Le mécanisme d'interruption de base fonctionne comme suit : le matériel de la CPU dispose d'une ligne de demande d'interruption que la CPU surveille après chaque instruction exécutée. Lorsque la CPU détecte un signal sur cette ligne de demande d'interruption émis par un contrôleur, elle lit le numéro d'interruption et accède à la routine appropriée en utilisant ce numéro comme index dans le vecteur d'interruption. Ainsi, le contrôleur de périphérique déclenche une interruption en envoyant un signal, que la CPU capte et transmet au gestionnaire d'interruption, lequel efface l'interruption en traitant le périphérique concerné.

Les CPU modernes possèdent généralement deux lignes de demande d'interruption :

- ✓ L'interruption non masquable : réservée aux événements critiques comme les erreurs de mémoire irrécupérables.
- ✓ L'interruption masquable : qui peut être désactivée par la CPU avant l'exécution de séquences d'instructions critiques pour éviter les interruptions pendant ces processus. Elle est principalement utilisée par les contrôleurs de périphériques pour signaler des besoins de service.

Pour prioriser les tâches les plus urgentes, les ordinateurs modernes utilisent un système de priorités d'interruption. Étant donné que les interruptions sont fréquemment sollicitées pour des traitements urgents, une gestion efficace de ces interruptions est essentielle pour maintenir de bonnes performances du système.

VII.Architecture du système informatique

Un système informatique peut être organisé de différentes manières, que nous pouvons les classer en fonction du nombre de processeurs à usage général utilisés.

VIII.1 Systèmes à processeur unique

Il y a plusieurs décennies, la majorité des systèmes informatiques utilisaient un seul processeur avec un seul cœur de traitement. Le noyau de ce processeur était responsable de l'exécution des instructions et du stockage local des données dans les registres. Ce processeur principal était capable d'exécuter un ensemble d'instructions général, y compris celles nécessaires aux processus. En plus de ce processeur principal, ces systèmes pouvaient comporter des processeurs spécialisés, tels que les contrôleurs de disque, de clavier et graphiques.

Ces processeurs spécialisés, qui effectuent des tâches spécifiques, possédaient un ensemble d'instructions limité et ne géraient pas de processus. Ils étaient parfois supervisés par le système d'exploitation, qui leur fournissait des instructions sur les tâches à accomplir et surveillait leur état. Par exemple, un microprocesseur de contrôleur de disque recevait des requêtes du processeur principal et gérait sa propre file d'attente de disque ainsi que son algorithme de planification, ce qui permettait de décharger le processeur principal de cette tâche. De même, un microprocesseur dans un clavier convertissait les frappes en codes envoyés au processeur principal. Dans certains systèmes, ces processeurs spécialisés sont intégrés en tant que composants de bas niveau du matériel et fonctionnent de manière autonome, sans interaction directe avec le système d'exploitation.

Aujourd'hui, l'utilisation de microprocesseurs spécialisés est courante, mais cela ne transforme pas un système à processeur unique en un système multiprocesseur. Selon cette

définition, très peu des systèmes informatiques modernes sont encore considérés comme des systèmes à processeur unique.

VIII.2 Systèmes multiprocesseurs

Aujourd'hui, les systèmes multiprocesseurs dominent les ordinateurs modernes, les appareils mobiles et les serveurs. Traditionnellement, ces systèmes sont équipés de deux processeurs (ou plus), chacun possédant un seul cœur de traitement. Les processeurs partagent le bus informatique, et parfois l'horloge, la mémoire et les périphériques. L'avantage principal des systèmes multiprocesseurs est l'augmentation du débit global : en ajoutant plus de processeurs, on s'attend à accomplir plus de tâches en moins de temps. Cependant, le gain de performance n'est pas proportionnel au nombre de processeurs ; il est généralement inférieur au nombre total de processeurs. Cette réduction est due à la surcharge liée à la coordination des tâches entre les processeurs et à la contention pour les ressources partagées.

Les systèmes multiprocesseurs les plus répandus utilisent le multitraitement symétrique (SMP), où chaque processeur peut exécuter toutes les tâches, y compris les fonctions du système d'exploitation et les processus utilisateur. Dans une architecture SMP typique, chaque processeur dispose de son propre ensemble de registres et d'un cache privé, tout en partageant la mémoire physique via le bus système. Cela permet une gestion efficace des ressources, tout en permettant à chaque processeur de travailler sur des tâches indépendantes ou collaboratives, optimisant ainsi les performances globales du système.

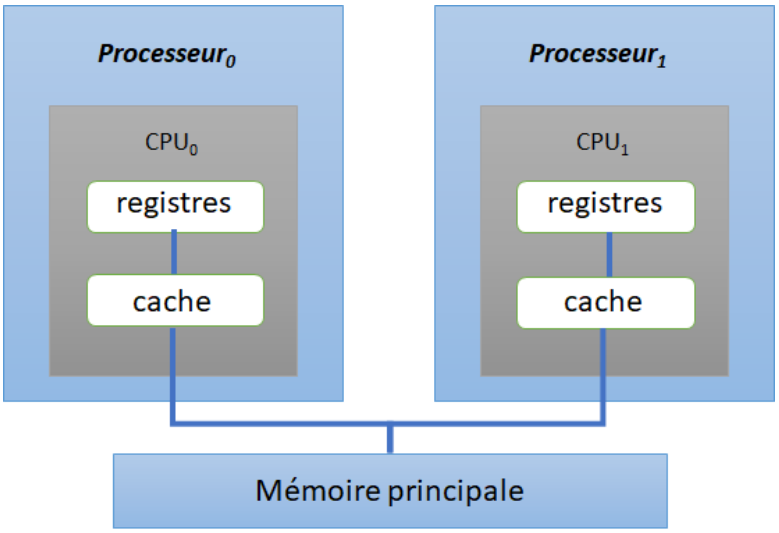


Figure 4 Systèmes multiprocesseurs

Le modèle multiprocesseur présente l'avantage de permettre l'exécution simultanée de nombreux processus, avec N processus pouvant être traités si N processeurs sont disponibles, sans que les performances ne se dégradent de manière significative. Toutefois,

comme les unités centrales sont indépendantes, il peut arriver qu'une unité soit inactive alors qu'une autre est surchargée, créant ainsi des inefficacités. Ces problèmes peuvent être atténués si les processeurs partagent certaines structures de données, ce qui permet de répartir dynamiquement les processus et les ressources (comme la mémoire) entre les différents processeurs, réduisant ainsi les déséquilibres de charge de travail. Une telle configuration nécessite une gestion soignée pour éviter les inefficacités.

La notion de multiprocesseur a évolué pour inclure les systèmes multicœurs, où plusieurs cœurs de traitement sont intégrés sur une seule puce. Les systèmes multicœurs sont souvent plus efficaces que les systèmes multi-processeurs à cœurs uniques, car la communication entre les cœurs sur une même puce est plus rapide que la communication entre plusieurs puces. De plus, les puces multicœurs consomment généralement moins d'énergie que plusieurs puces à un seul cœur, ce qui est particulièrement avantageux pour les appareils mobiles et les ordinateurs portables.

Dans une architecture multicœur typique, chaque cœur possède son propre ensemble de registres ainsi qu'un cache local, appelé cache de niveau 1 (L1). Un cache de niveau 2 (L2), qui est partagé entre les cœurs mais reste local à la puce, est également présent. Cette combinaison de caches locaux et partagés permet d'optimiser les performances en offrant des caches locaux plus rapides et plus petits, tout en ayant des caches partagés plus grands mais plus lents. Du point de vue du système d'exploitation, un processeur multicœur avec N cœurs est vu comme N processeurs distincts, ce qui représente un défi pour les concepteurs de systèmes d'exploitation et les programmeurs afin d'utiliser ces cœurs de manière optimale.

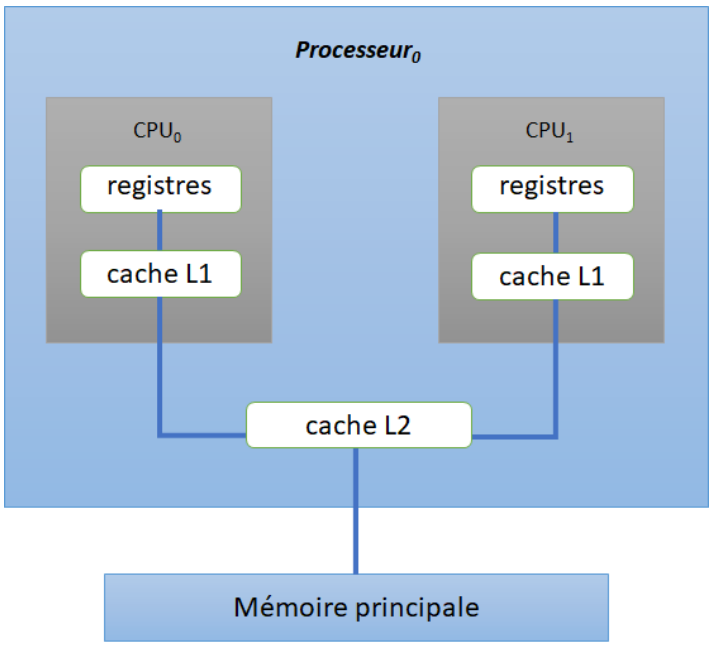


Figure 5 Architecture multicœur typique

IX. Définitions

X.1 Le Système d'Exploitation comme Machine Virtuelle

Un système d'exploitation (SE) peut être considéré comme une machine virtuelle car il fournit une couche d'abstraction entre l'utilisateur (ou le programmeur) et le matériel physique de l'ordinateur. Cette abstraction simplifie le processus de programmation en permettant aux développeurs d'interagir avec l'ordinateur à l'aide d'un ensemble standardisé de commandes et d'interfaces, sans avoir à comprendre et manipuler les détails complexes du matériel.

Concrètement, cela signifie que lorsqu'un programmeur souhaite réaliser une tâche, comme écrire des données sur un disque dur, afficher des informations à l'écran ou gérer des fichiers, il n'a pas besoin de se soucier de la manière dont ces tâches sont exécutées au niveau du matériel. Le système d'exploitation prend en charge les opérations complexes, comme la recherche de l'adresse mémoire correcte, la gestion des pilotes matériels et l'assurance que les données sont correctement stockées et récupérées. Cela rend la programmation plus accessible et réduit les risques d'erreurs liées à une interaction directe avec le matériel.

Par exemple, lorsqu'un programme demande à sauvegarder un fichier, le SE détermine où le fichier doit être stocké sur le disque, gère les éventuels conflits avec d'autres processus tentant d'accéder au disque, et traite les erreurs éventuelles. Pour le programmeur, cela se résume à une simple fonction, comme `save()`, avec toute la complexité sous-jacente complètement masquée. Cet aspect "machine virtuelle" du SE est crucial pour créer des logiciels capables de fonctionner sur différents types de matériel sans modification, car le SE gère les différences entre les configurations matérielles.

X.2 Le Système d'Exploitation comme Gestionnaire de Ressources

Au-delà de son rôle de machine virtuelle, le système d'exploitation agit également en tant que gestionnaire de ressources. C'est l'une de ses fonctions les plus fondamentales, car il doit allouer efficacement les ressources limitées de l'ordinateur—telles que le temps CPU, la mémoire, le stockage et les périphériques—entre les différents programmes et utilisateurs qui en ont besoin. Cette gestion des ressources est cruciale pour maintenir les performances, la stabilité et la sécurité du système informatique.

Lorsque plusieurs programmes sont exécutés simultanément ou que plusieurs utilisateurs sont connectés à un système, le SE doit garantir que chaque programme et utilisateur reçoive les ressources nécessaires sans interférer les uns avec les autres. Par exemple, le SE gère le temps de traitement du CPU en planifiant les tâches afin que chaque

programme obtienne une part équitable de la puissance du processeur. Cela empêche un programme unique de monopoliser le CPU, ce qui ralentirait ou bloquerait les autres applications.

La gestion de la mémoire est également un aspect critique du rôle du SE en tant que gestionnaire de ressources. Le SE garde une trace des portions de mémoire utilisées et disponibles, alloue la mémoire aux programmes selon les besoins, et veille à ce qu'un programme ne surcharge pas la mémoire utilisée par un autre. Cela prévient la corruption des données et assure un fonctionnement fluide des programmes.

De même, le SE gère l'accès aux dispositifs de stockage, tels que les disques durs et les SSD. Il garde une trace de l'emplacement des fichiers, contrôle l'accès à ces fichiers et coordonne l'écriture et la lecture des données pour éviter les conflits. Par exemple, si deux programmes tentent d'écrire simultanément dans le même fichier, le SE doit s'assurer que les données ne sont pas corrompues en gérant l'ordre des écritures ou en verrouillant le fichier jusqu'à ce que l'opération soit terminée.

Cette gestion des ressources s'étend également à la sécurité. Le SE applique des autorisations et des contrôles d'accès, veillant à ce que les utilisateurs et les programmes ne puissent pas accéder aux ressources auxquelles ils ne sont pas autorisés. Cela est crucial dans les environnements multi-utilisateurs, où le SE doit empêcher les actions d'un utilisateur de nuire aux données ou aux applications d'un autre.

X.3 Le Système d'Exploitation comme Mécanisme de Sécurité

Le rôle du SE peut être comparé aux mécanismes de sécurité dans un véhicule, tels que les limiteurs de vitesse ou les freins antiblocage, qui sont conçus pour prévenir les accidents et assurer un fonctionnement sûr. Tout comme ces dispositifs protègent le véhicule et ses occupants des situations dangereuses, le SE protège le système informatique des erreurs, des conflits et des violations de sécurité.

Par exemple, tout comme un limiteur de vitesse empêche une voiture de dépasser une vitesse sécurisée, le SE empêche un programme d'utiliser plus de ressources que ce qui est autorisé, ce qui pourrait faire planter le système ou ralentir d'autres applications. De même, tout comme les freins antiblocage empêchent les roues d'une voiture de se bloquer et de perdre le contrôle, le SE prévient la corruption des données et veille à ce que les ressources critiques du système soient utilisées de manière sûre et efficace.

En résumé, le système d'exploitation est un composant essentiel de tout système informatique. En agissant à la fois comme machine virtuelle et gestionnaire de ressources, il simplifie la programmation, assure une utilisation efficace des ressources et fournit un

environnement stable et sécurisé pour l'exécution des applications. Son rôle protecteur, semblable aux dispositifs de sécurité dans un véhicule, garantit le bon fonctionnement du système et l'utilisation sécurisée des ressources, permettant ainsi aux utilisateurs et aux développeurs d'interagir avec l'ordinateur de manière fiable et cohérente.

X.4 Définition Simple d'un Système d'Exploitation

Un système d'exploitation (SE) est une couche logicielle cruciale qui gère les ressources matérielles et logicielles de l'ordinateur et fournit des services communs aux programmes informatiques. Il agit comme un intermédiaire entre les utilisateurs et le matériel de l'ordinateur, facilitant l'exécution des applications en offrant des fonctions essentielles telles que la gestion des processus, la gestion de la mémoire, la gestion du système de fichiers et le contrôle des périphériques. Le SE veille à ce que les différents programmes et utilisateurs n'interfèrent pas les uns avec les autres, maintient la stabilité et la sécurité du système, et permet une utilisation efficace des ressources matérielles. En créant un environnement convivial, le SE permet aux utilisateurs d'interagir avec leurs dispositifs et de faire fonctionner des applications sans avoir à gérer directement le matériel sous-jacent.

XI. Objectifs des systèmes exploitations

Les systèmes d'exploitation (SE) ou Operating systems (OS) sont essentiels pour la gestion des ressources matérielles et logicielles des ordinateurs, ainsi que pour fournir une gamme de services aux utilisateurs et aux applications. Leurs principaux objectifs incluent :

- ✓ Gestion des Ressources: Le SE gère efficacement les ressources matérielles de l'ordinateur, telles que le CPU, la mémoire, le stockage et les périphériques d'entrée/sortie. Il alloue ces ressources de manière efficace entre les processus et les applications concurrentes pour garantir un fonctionnement fluide. Cela inclut la planification du temps CPU, la gestion de l'allocation de la mémoire, l'organisation des fichiers sur les dispositifs de stockage et la coordination des interactions avec les périphériques.
- ✓ Gestion des Processus : Le SE s'occupe de l'exécution des processus, ce qui implique la création, la planification et la terminaison des processus. Il veille à ce que les processus soient exécutés de manière efficace et que les ressources système soient partagées de manière appropriée. Cela comprend la gestion des priorités des processus, la gestion du multitâche et l'exécution des changements de contexte pour permettre à plusieurs processus de fonctionner simultanément sans interférence.
- ✓ Gestion du Système de Fichiers : Le SE gère les fichiers et les répertoires, en fournissant une méthode structurée pour le stockage et la récupération des données. Il s'occupe de tâches telles que la création, la suppression et la manipulation des fichiers tout en maintenant l'intégrité et la sécurité des données. Le SE organise les fichiers dans une

structure hiérarchique, contrôle les autorisations d'accès et facilite les opérations sur les fichiers comme la lecture et l'écriture.

- ✓ Interface Utilisateur : Les systèmes d'exploitation fournissent une interface utilisateur qui permet aux utilisateurs d'interagir avec le système informatique. Cela peut être sous la forme d'une interface graphique (GUI) avec des fenêtres, des icônes et des menus, ou d'une interface en ligne de commande (CLI) avec des commandes textuelles. L'interface utilisateur traduit les commandes des utilisateurs en actions que le matériel peut exécuter, facilitant ainsi l'utilisation du système par les utilisateurs.
- ✓ Sécurité et Protection : Assurer la sécurité et la protection du système et de ses données est un objectif crucial du SE. Il met en œuvre des mesures de sécurité telles que l'authentification et le contrôle d'accès pour prévenir les accès non autorisés et protéger les informations sensibles. Le SE gère les autorisations des utilisateurs, applique le chiffrement des données et protège le système contre les menaces potentielles.

Comprendre ces objectifs aide à apprécier comment les systèmes d'exploitation gèrent les ressources, les processus, les fichiers et les interactions avec les utilisateurs tout en garantissant la sécurité et l'efficacité dans les environnements informatiques.

XII. Fonctions du système d'exploitation

Les fonctions d'un système d'exploitation (SE) englobent plusieurs tâches essentielles qui garantissent le fonctionnement efficace et sécurisé d'un système informatique.

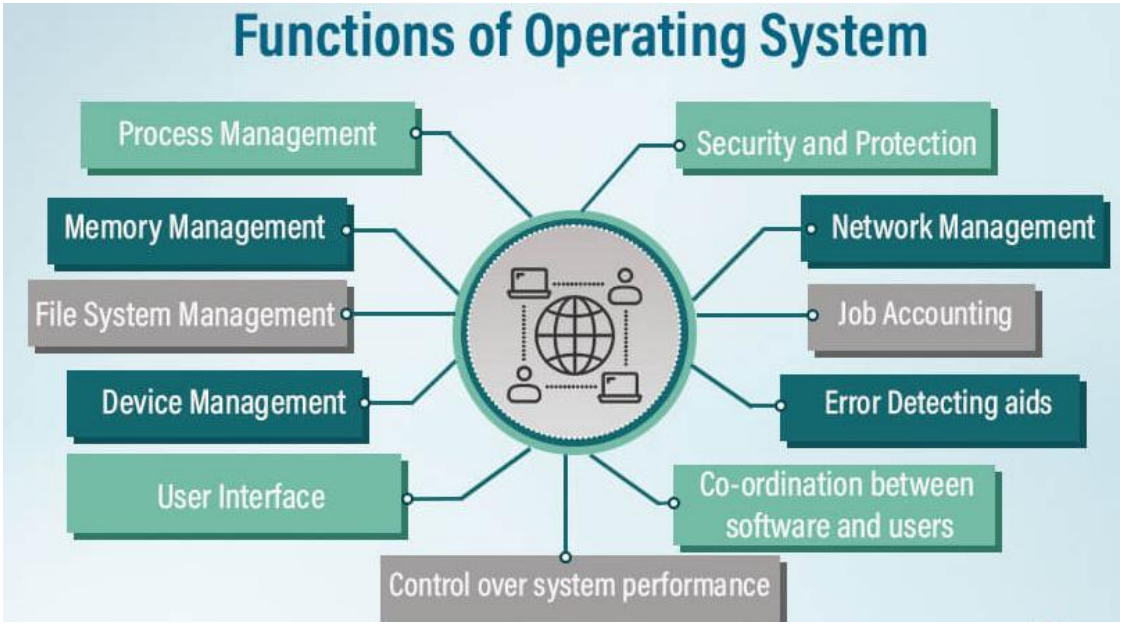


Figure 6 Fonctions d'un système d'exploitation

Ces fonctions comprennent :

- ✓ Gestion des Processus : Le SE est responsable de la création, de la planification et de la terminaison des processus. Il gère l'exécution des processus et s'assure qu'ils partagent le temps du CPU de manière efficace. Cela inclut la gestion du multitâche, la priorisation des processus et la communication entre processus.
- ✓ Gestion de la Mémoire : Le SE gère les ressources mémoire de l'ordinateur, y compris la RAM et la mémoire virtuelle. Il alloue de la mémoire aux processus, gère la protection de la mémoire et optimise l'utilisation de la mémoire à l'aide de la pagination et de la segmentation pour éviter les conflits entre les processus.
- ✓ Gestion du Système de Fichiers : Le SE organise et contrôle l'accès aux fichiers et aux répertoires sur les dispositifs de stockage. Il gère la création, la suppression, la lecture et l'écriture des fichiers, ainsi que les autorisations des fichiers et les structures de répertoires pour garantir l'intégrité des données et une gestion efficace du stockage.
- ✓ Gestion des Périphériques : Le SE gère les périphériques d'entrée et de sortie tels que les claviers, les souris, les imprimantes et les disques durs. Il fournit des pilotes de périphériques qui traduisent les commandes générales en actions spécifiques au périphérique, gère le tamponnage et le spoolage pour réguler le flux de données, et assure une interaction fluide entre le matériel et le logiciel.
- ✓ Interface Utilisateur : Le SE fournit une interface utilisateur qui permet aux utilisateurs d'interagir avec le système informatique. Il peut s'agir d'une interface graphique (GUI) avec des éléments visuels comme des fenêtres et des icônes, ou d'une interface en ligne de commande (CLI) où les utilisateurs saisissent des commandes textuelles. L'interface utilisateur traduit les actions des utilisateurs en commandes système et fournit des retours.
- ✓ Sécurité et Protection : Le SE met en œuvre des mesures de sécurité pour protéger le système contre les accès non autorisés et les menaces potentielles. Il gère l'authentification des utilisateurs, applique des politiques de contrôle d'accès et fournit des mécanismes de chiffrement des données et de protection contre les logiciels malveillants et autres menaces de sécurité.
- ✓ Gestion des reseaux : Le SE gère les connexions réseau et les communications entre les ordinateurs. Il gère les protocoles réseau, les ressources réseau, et fournit des services pour la transmission de données, y compris la gestion des interfaces réseau et la sécurisation des communications sur les réseaux.
- ✓ Surveillance du Système et Gestion des Performances : Le SE surveille les performances du système et l'utilisation des ressources, fournissant des outils pour l'analyse des performances et le diagnostic du système. Il gère les ressources système pour assurer des performances optimales et répond aux événements et erreurs du système.

Ces fonctions permettent au SE de gérer efficacement les ressources matérielles et logicielles, de fournir un environnement de fonctionnement stable et sécurisé, et de faciliter les interactions des utilisateurs avec le système informatique.

XIII. Organisation en couches d'un SE

Le système d'exploitation est organisé en couches hiérarchiques. Chaque couche repose sur les services fournis par les couches situées en dessous d'elle. Cette structuration permet de séparer les différentes fonctions du système d'exploitation, facilitant ainsi sa gestion et son développement. Cette architecture en couches est représentée dans la figure suivante, qui illustre comment chaque niveau du système d'exploitation dépend des services des niveaux inférieurs tout en offrant des services aux niveaux supérieurs. Voici une description de ces couches, en incluant leurs rôles :

- ✓ Noyau (Kernel) : C'est l'interface principale entre le matériel et le logiciel. Il gère les interruptions et les processus, assurant la communication et la synchronisation entre différents programmes.
- ✓ Gestionnaire de la mémoire : Cette couche se charge du partage de la mémoire entre les processus en attente d'exécution. Elle alloue et libère de la mémoire pour optimiser l'utilisation de cette ressource.
- ✓ Gestion des E/S (Entrées/Sorties) : Responsable de la gestion de tous les périphériques tels que le clavier, l'écran, et les disques. Elle traduit les demandes des programmes en opérations spécifiques sur les périphériques.
- ✓ Gestionnaire de fichiers : S'occupe de la gestion de l'espace disque et de la manipulation des fichiers. Cela inclut la création, la suppression, et l'organisation des fichiers sur le disque dur.
- ✓ Allocation de ressources : Assure une utilisation efficace des ressources du système. Il fournit des statistiques sur l'exploitation des ressources, garantit que chaque processus obtient les ressources nécessaires dans des délais raisonnables, et évite les situations de blocage ou d'attente excessive.

Cette organisation permet de simplifier le design du système d'exploitation, de rendre chaque couche plus modulaire et indépendante, et de faciliter la maintenance et les mises à jour.

XIV. Modes de Fonctionnement d'un Système d'Exploitation

Étant donné que le système d'exploitation (SE) et ses utilisateurs partagent les ressources matérielles et logicielles d'un ordinateur, il est crucial qu'un SE bien conçu protège ces ressources. Il doit empêcher un programme défectueux ou malveillant de

perturber le fonctionnement d'autres programmes, ou même du SE lui-même. Pour assurer une exécution fiable et sécurisée, il est nécessaire de différencier l'exécution du code du SE de celle du code utilisateur. La plupart des systèmes informatiques utilisent une approche matérielle pour distinguer ces différents modes d'exécution.

Deux modes de fonctionnement sont généralement utilisés : le mode utilisateur et le mode noyau (également appelé mode superviseur, mode système ou mode privilégié). Un bit, connu sous le nom de bit de mode, est intégré dans le matériel pour indiquer le mode courant : noyau (0) ou utilisateur (1). Ce bit permet de différencier les opérations effectuées au nom du SE de celles effectuées au nom de l'utilisateur.

En mode utilisateur, le système informatique exécute des applications au nom de l'utilisateur. Lorsqu'une application demande un service au SE via un appel système, le système doit passer du mode utilisateur au mode noyau pour traiter la demande.

Au démarrage du système, le matériel fonctionne en mode noyau. Le SE est ensuite chargé et les applications utilisateur sont lancées en mode utilisateur. À chaque interruption, le matériel passe du mode utilisateur au mode noyau (c'est-à-dire que le bit de mode passe à 0). Ainsi, lorsque le SE prend le contrôle, il fonctionne en mode noyau. Avant de transférer le contrôle à un programme utilisateur, le SE passe toujours en mode utilisateur (bit de mode à 1).

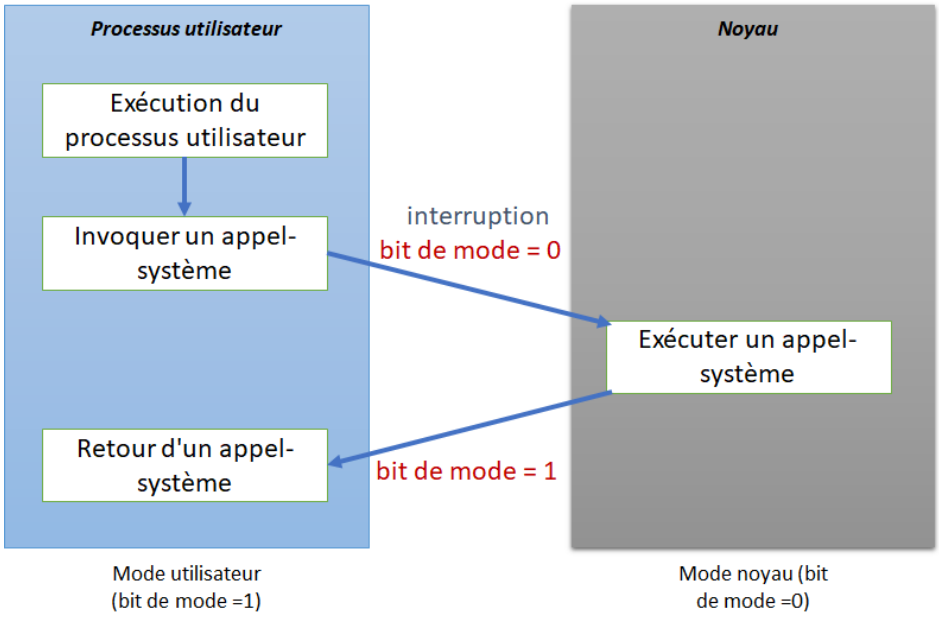


Figure 7 Modes de Fonctionnement d'un Système d'Exploitation

Le fonctionnement en double mode permet de protéger le SE contre les actions non autorisées et les erreurs potentielles des utilisateurs. Cette protection est assurée en désignant certaines instructions comme des instructions privilégiées. Le matériel autorise

l'exécution de ces instructions uniquement en mode noyau. Si une tentative d'exécution d'une instruction privilégiée est faite en mode utilisateur, le matériel rejette l'instruction, la considérant comme illégale, et la transmet au SE.

Par exemple, les instructions pour passer en mode noyau sont des instructions privilégiées, tout comme celles qui contrôlent les E/S, les temporisateurs et les interruptions.

Un appel système est une méthode permettant à un programme informatique de demander un service au noyau du système d'exploitation sur lequel il s'exécute. Les systèmes exécutent des milliers d'appels système par seconde. Les appels système permettent à un programme utilisateur de demander au SE d'exécuter des tâches réservées au SE pour le compte du programme utilisateur. Les appels système fournissent une interface aux services mis à disposition par un système d'exploitation. Ces appels sont généralement disponibles sous forme de fonctions écrites en C et C++, bien que certaines tâches de bas niveau (par exemple, les tâches où le matériel doit être accessible directement) puissent être écrites à l'aide d'instructions en langage assembleur.

Les appels système, qui permettent aux programmes utilisateurs d'interagir avec le système d'exploitation, se classifient généralement en six grandes catégories :

- ✓ Contrôle de Processus : Ces appels gèrent l'exécution des processus, tels que la création, la gestion, et la terminaison des programmes. Par exemple, un appel système peut être utilisé pour lancer un nouveau programme ou pour mettre un processus en pause.
- ✓ Gestion de Fichiers : Cette catégorie concerne la manipulation des fichiers et des répertoires sur les dispositifs de stockage. Les appels système permettent de créer, supprimer, lire ou écrire des fichiers. Par exemple, une application peut utiliser un appel système pour sauvegarder des données dans un fichier.
- ✓ Gestion de Périphériques : Les appels dans cette catégorie gèrent les interactions avec les périphériques matériels, comme les claviers, les souris, et les imprimantes. Par exemple, un appel système peut être utilisé pour envoyer des données à une imprimante pour impression.
- ✓ Maintenance des Informations : Ces appels sont utilisés pour la gestion des informations internes du système, comme les attributs des fichiers ou les paramètres du système. Par exemple, un appel système peut modifier les permissions d'accès à un fichier.
- ✓ Communications : Cette catégorie concerne les mécanismes permettant aux processus de communiquer entre eux ou avec des systèmes externes. Par exemple, un appel système peut être utilisé pour envoyer un message à un autre processus ou pour établir une connexion réseau.
- ✓ Protection : Les appels de cette catégorie assurent la sécurité du système en contrôlant les accès et en protégeant les ressources contre les utilisateurs non autorisés. Par

exemple, un appel système peut vérifier si un utilisateur a les droits nécessaires pour accéder à un fichier ou exécuter une action spécifique.

Ces catégories d'appels système permettent au système d'exploitation de gérer efficacement les ressources et d'assurer une interaction fluide et sécurisée entre les programmes et le matériel de l'ordinateur.

XV. Historique et évolution des systèmes d'exploitation

XX.1 Premiers systèmes d'exploitation :

Dans les années 1950 et 1960, les premiers systèmes d'exploitation étaient conçus pour gérer les premiers ordinateurs tels que les UNIVAC et les IBM 7094. Ces systèmes offraient des fonctionnalités élémentaires comme la gestion des tâches en mode batch, où les programmes étaient exécutés séquentiellement à partir de cartes perforées ou de bandes magnétiques.

- ✓ Nouveautés : L'introduction des premiers concepts de gestion des processus et d'entrée/sortie, bien que rudimentaires, a posé les bases pour les systèmes plus avancés.
- ✓ Avantages : Ils ont permis d'automatiser l'exécution des tâches et d'optimiser l'utilisation des ordinateurs coûteux.
- ✓ Inconvénients : Leur utilisation était limitée à des séquences de tâches fixes, et l'interaction avec l'utilisateur était minimaliste.

XX.2 Systèmes d'exploitation à temps partagé :

Dans les années 1970, avec l'émergence de systèmes comme UNIX, les systèmes d'exploitation ont évolué pour supporter le temps partagé, permettant à plusieurs utilisateurs d'utiliser un même ordinateur en même temps.

- ✓ Nouveautés : Gestion dynamique des processus, support du multitâche et amélioration de l'interaction utilisateur avec la gestion de la mémoire virtuelle.
- ✓ Exemples : UNIX et Multics.
- ✓ Avantages : Introduction du multitâche préemptif, gestion des processus et des fichiers, et interfaces utilisateurs plus flexibles.
- ✓ Inconvénients : La complexité accrue du système peut entraîner des problèmes de sécurité et de gestion des ressources.

XX.3Systèmes d'exploitation graphiques :

Les années 1980 ont vu l'avènement des systèmes d'exploitation graphiques tels que MS-DOS avec Windows 3.1 et Mac OS. Ces systèmes ont introduit des interfaces graphiques utilisateur (GUI) avec des fenêtres, des icônes et des menus.

- ✓ Nouveautés : Introduction des interfaces graphiques qui ont transformé l'interaction avec les ordinateurs, rendant les tâches plus visuelles et interactives.
- ✓ Exemples : Windows 95, Mac OS 8.
- ✓ Avantages : Interfaces plus intuitives et accessibles, facilitant l'utilisation des ordinateurs pour les non-spécialistes.
- ✓ Inconvénients : Exigences matérielles plus élevées et complexité accrue dans la gestion des ressources graphiques.

XX.4Systèmes d'exploitation mobiles et embarqués :

Avec l'avènement des années 2000, les systèmes d'exploitation ont été adaptés pour les appareils mobiles et embarqués.

- ✓ Nouveautés : Développement de systèmes adaptés à des environnements spécifiques avec des interfaces tactiles et une intégration étroite avec des services en ligne.
- ✓ Exemples : Android et iOS pour les smartphones et tablettes, ainsi que des systèmes comme Embedded Linux pour les appareils connectés.
- ✓ Avantages : Optimisation pour les écrans tactiles, faible consommation d'énergie et connectivité sans fil.
- ✓ Inconvénients : Limitation des capacités par rapport aux ordinateurs de bureau et défis de sécurité spécifiques aux appareils mobiles.

XX.5Systèmes d'exploitation modernes et cloud :

Aujourd'hui, les systèmes d'exploitation s'adaptent aux environnements cloud et aux architectures distribuées.

- ✓ Nouveautés : Technologies de virtualisation avancées, gestion des conteneurs, et intégration étroite avec des services cloud pour une flexibilité et une évolutivité accrues.
- ✓ Exemples : Windows Server pour les environnements d'entreprise, Linux pour le cloud.
- ✓ Avantages : Virtualisation, gestion des conteneurs, haute disponibilité, et adaptation aux besoins dynamiques des entreprises.
- ✓ Inconvénients : Complexité accrue dans la gestion des ressources distribuées et défis de sécurité liés à la protection des données dans le cloud.

D’une autre façon, on peut conclure que les systèmes d'exploitation ont évolué en parallèle avec l'architecture des ordinateurs sur lesquels ils fonctionnaient, chaque avancée

dans l'un stimulant des innovations dans l'autre. Cette coévolution a été guidée par plusieurs objectifs clés :

- ✓ Libérer les programmeurs des tâches de programmation répétitives et fastidieuses, afin qu'ils puissent se concentrer sur le développement d'applications plus complexes et innovantes.
- ✓ Protéger le système contre les erreurs des utilisateurs et les manipulations incorrectes, garantissant ainsi la stabilité et la sécurité de l'ensemble du système.
- ✓ Offrir une interface simple, uniforme et cohérente pour interagir avec la machine, facilitant ainsi l'accès aux ressources matérielles et logicielles.

Au fil du temps, quatre grandes générations de systèmes d'exploitation ont émergé, chacune répondant aux besoins technologiques et utilisateur de son époque :

- ✓ Premiers systèmes : Ces systèmes rudimentaires étaient principalement destinés à gérer les opérations de base des premiers ordinateurs, souvent en mode interactif limité, et étaient étroitement liés à l'architecture matérielle spécifique.
- ✓ Systèmes de traitement par lots : Avec l'avènement des ordinateurs plus puissants, les systèmes de traitement par lots ont permis d'exécuter plusieurs programmes en séquence sans intervention humaine constante, optimisant ainsi l'utilisation des ressources.
- ✓ Systèmes multiprogrammés : Ces systèmes ont introduit la capacité d'exécuter plusieurs programmes simultanément en partageant les ressources de l'ordinateur, augmentant l'efficacité et la réactivité.
- ✓ Systèmes à temps partagé : Finalement, les systèmes à temps partagé ont permis à plusieurs utilisateurs d'interagir avec un ordinateur en temps réel, marquant le début de l'informatique interactive et multi-utilisateur telle que nous la connaissons aujourd'hui.

Chaque génération a apporté ses propres innovations, avantages et défis, contribuant à façonner les systèmes d'exploitation modernes qui sont à la fois robustes, polyvalents et capables de gérer une multitude de tâches et d'utilisateurs en parallèle.

XXI. Taxonomie et classification des systèmes d'exploitation

La taxonomie et la classification des systèmes d'exploitation consistent à organiser et catégoriser les différents types de systèmes d'exploitation en fonction de leurs caractéristiques spécifiques, telles que leur architecture, leur mode de fonctionnement, et leurs domaines d'application. Cette classification permet de mieux comprendre les divers types de systèmes d'exploitation, leurs usages, et leurs évolutions technologiques, en les regroupant selon des critères communs.

XXII.1 Taxonomie des systèmes d'exploitation

La taxonomie des systèmes d'exploitation regroupe et catégorise les différents OS en fonction de leurs architectures, de leurs fonctionnalités, et des besoins spécifiques qu'ils adressent. Cette classification offre une vue d'ensemble sur l'évolution des systèmes d'exploitation et leurs applications variées.

- ✓ Systèmes des ordinateurs personnels (PC) : Avec la baisse des coûts matériels, il est devenu possible de disposer de systèmes informatiques dédiés à un seul utilisateur, donnant naissance aux ordinateurs personnels ou PC dans les années 70. Ces systèmes, initialement non multitâches et mono-utilisateur, privilégient la commodité, la rapidité et l'interactivité avec l'utilisateur plutôt que l'optimisation maximale des ressources matérielles. Exemples : MS-DOS de Microsoft et Macintosh d'Apple.
- ✓ Systèmes parallèles : Ces systèmes, ou systèmes fortement couplés, utilisent plusieurs processeurs en communication étroite, partageant bus, horloge, et parfois mémoire et périphériques. L'avantage réside dans l'augmentation de la capacité de traitement et la fiabilité, car la panne d'un processeur ne bloque pas le système, mais le ralentit simplement. Le multitraitement symétrique (SMP) est couramment utilisé, où chaque processeur exécute une copie du système d'exploitation, collaborant au besoin.
- ✓ Systèmes distribués et réseaux : Avec l'essor des réseaux dans les années 1980, les systèmes d'exploitation réseau et distribués sont apparus. Les systèmes réseaux permettent l'accès à des services communs coûteux, comme le stockage et l'impression, via des serveurs accessibles par des clients. Les systèmes distribués, en revanche, consistent en plusieurs ordinateurs connectés coopérant pour exécuter des applications distribuées, donnant l'illusion d'un seul système d'exploitation contrôlant le réseau.
- ✓ Systèmes temps réels : Spécialisés et dédiés à des applications spécifiques, comme le contrôle industriel ou médical, ces systèmes doivent respecter des contraintes de temps strictes pour éviter les pannes. Exemples : contrôle de réacteurs nucléaires, chaînes de production robotisées, systèmes d'imagerie médicale.
- ✓ Systèmes embarqués : Ces systèmes, conçus pour fonctionner sur des appareils autonomes ou de petite taille, tels que les appareils électroménagers, robots, ou smartphones, nécessitent une gestion efficace de l'énergie et des ressources limitées. Exemples : Palm OS pour PDA, Java Card pour cartes à puce, et TinyOS pour réseaux de capteurs.
- ✓ Systèmes multi-cœurs : Ces systèmes combinent plusieurs processeurs sur une seule puce, nécessitant des systèmes d'exploitation capables de gérer efficacement ce parallélisme. Bien que les systèmes actuels ne supportent pas totalement ces architectures, des améliorations ont été apportées, notamment en matière de gestion de l'ordonnancement et des caches.

XXII.2 Classification des systèmes d'exploitation

La classification des systèmes d'exploitation permet de les organiser selon différents critères, ce qui aide à comprendre leurs fonctionnalités et leurs domaines d'application. Voici les principales catégories de cette classification :

XXII.2.1 Selon les contraintes d'utilisation :

- ✓ Mono-utilisateur/mono-tâche: Ce type de système est conçu pour un seul utilisateur à la fois et ne permet l'exécution que d'une tâche unique. Un exemple classique est MS-DOS, qui ne permettait qu'une seule application active à la fois.
- ✓ Mono-utilisateur/multitâche: Ces systèmes permettent à un seul utilisateur de réaliser plusieurs tâches simultanément. Par exemple, Windows permet à un utilisateur de travailler sur plusieurs applications en même temps, comme écouter de la musique tout en naviguant sur Internet.
- ✓ Multi-utilisateurs/multitâches : Ces systèmes permettent à plusieurs utilisateurs d'accéder et d'exécuter des tâches simultanément, tout en partageant les mêmes ressources matérielles. Unix en est un exemple, où plusieurs utilisateurs peuvent se connecter en même temps et exécuter diverses tâches sans interférer les uns avec les autres.

XXII.2.2 Selon les services fournis :

- ✓ Systèmes temps réel : Utilisés dans des environnements où le temps de réponse est critique, comme les systèmes de contrôle industriels ou les dispositifs médicaux. Par exemple, un système de contrôle de réacteur nucléaire doit réagir instantanément aux changements de conditions pour garantir la sécurité.
- ✓ Systèmes transactionnels : Conçus pour gérer de grandes quantités de transactions tout en assurant l'intégrité des données. Les systèmes utilisés dans les banques et les réservations de vols en ligne en sont des exemples, où il est crucial que toutes les transactions soient correctement enregistrées et cohérentes.

XXII.2.3 Selon les architectures matérielles :

- ✓ Systèmes mono-processeur : Ces systèmes fonctionnent avec un seul processeur, capable de gérer plusieurs tâches par le biais du multitâche. Les anciens systèmes comme les premiers PC fonctionnaient ainsi, en simulant des tâches parallèles sur un unique processeur.
- ✓ Systèmes multiprocesseurs : Ces systèmes utilisent plusieurs processeurs pour traiter des tâches en parallèle, ce qui améliore la performance et la fiabilité. Des exemples modernes incluent Solaris et Linux sur des serveurs multiprocesseurs, qui peuvent gérer de lourdes charges de travail et offrir une haute disponibilité en cas de défaillance d'un processeur.

XXIII. Exemples de systèmes d'exploitation

Plusieurs variantes de systèmes d'exploitation ont apparu dès les années 50 jusqu'aujourd'hui, les fameux sont :

XXIV.1 MS-DOS

MS-DOS (Microsoft Disk Operating System) est un système d'exploitation développé par Microsoft, lancé en 1981 pour les premiers PC compatibles IBM. Conçu comme un système mono-utilisateur et mono-tâche, MS-DOS permettait l'exécution d'une seule application à la fois. Il se distingue par sa simplicité et son interface en ligne de commande.

Versions :

- ✓ MS-DOS 1.0 (1981) : Introduction des commandes de base pour la gestion des fichiers et des répertoires. Exemple : `DIR` pour afficher le contenu d'un répertoire.
- ✓ MS-DOS 2.0 (1983) : Ajout du support des sous-répertoires et des commandes améliorées. Exemple : `COPY` pour copier des fichiers d'un répertoire à un autre.
- ✓ MS-DOS 3.0 (1984) : Introduction du support pour les disques durs et améliorations en gestion de mémoire. Exemple : `FORMAT` pour formater un disque dur.
- ✓ MS-DOS 6.22 (1994) : Dernière version majeure avant la transition vers Windows, avec des utilitaires de gestion de disque comme `DEFRAG` pour la défragmentation.

Avantages :

Simplicité : Facile à utiliser pour des tâches simples avec une interface en ligne de commande.

Légèreté : Optimisé pour des machines avec des ressources limitées.

Inconvénients :

- ✓ Monotâche : Ne permet pas l'exécution simultanée de plusieurs programmes.
- ✓ Interface en ligne de commande : Moins intuitive pour les utilisateurs non techniques.

Exemple : MS-DOS était utilisé sur des ordinateurs tels que le IBM PC 5150 . Les utilisateurs entraient des commandes comme `COPY C:\file.txt D:\` pour copier des fichiers d'un disque à un autre.

XXIV.2 Windows

Windows est une famille de systèmes d'exploitation développés par Microsoft, caractérisés par une interface graphique conviviale et une capacité à exécuter plusieurs tâches simultanément. Depuis son lancement, Windows a évolué pour offrir une interface utilisateur graphique avancée et une gestion améliorée des ressources.

Versions :

- ✓ Windows 1.0 (1985) : Première version avec une interface graphique rudimentaire. Exemple : Utilisation de la souris pour interagir avec les fenêtres.
- ✓ Windows 95 (1995) : Introduction du menu Démarrer et de la barre des tâches. Exemple : `Start > Run` pour lancer des programmes et des fichiers.
- ✓ Windows XP (2001) : Connue pour sa stabilité et sa convivialité, avec une interface utilisateur améliorée et un support étendu pour les périphériques. Exemple : `Task Manager` pour surveiller les processus en cours.
- ✓ Windows 10 (2015) : Introduction de fonctionnalités modernes comme les bureaux virtuels et le navigateur Microsoft Edge. Exemple : `Cortana` pour la recherche vocale et le lancement d'applications.

Avantages :

- ✓ Interface graphique : Facilitée d'utilisation avec des éléments visuels intuitifs comme les fenêtres et les icônes.
- ✓ Multitâche : Capacité à exécuter plusieurs applications simultanément et à gérer les ressources efficacement.

Inconvénients :

- ✓ Consommation de ressources : Peut nécessiter des ressources matérielles importantes, ce qui peut affecter les performances sur des machines plus anciennes.
- ✓ Vulnérabilités de sécurité : Les versions populaires peuvent être des cibles pour les logiciels malveillants.

Exemple : Windows 7, lancé en 2009, est souvent loué pour sa stabilité et sa compatibilité avec divers logiciels et matériels, facilitant l'exécution d'applications telles que Microsoft Office et les jeux vidéo.

XXIV.3 Unix

Unix est un système d'exploitation multitâche et multi-utilisateur, développé dans les années 1960 et 1970 aux laboratoires Bell. Connu pour sa robustesse, sa sécurité, et sa flexibilité, Unix est la base de nombreux systèmes modernes, avec une architecture modulaire et une interface en ligne de commande puissante.

Versions :

- ✓ UNIX Version 7 (1979) : Version influente introduisant de nombreuses fonctionnalités standard. Exemple : Commande `ls` pour lister les fichiers d'un répertoire.

- ✓ BSD (Berkeley Software Distribution) : Développée à l'Université de Californie à Berkeley, influente pour des systèmes modernes comme FreeBSD. Exemple : Commande `ps` pour afficher les processus en cours.
- ✓ AIX (1986) : Version d'Unix développée par IBM pour ses serveurs, connue pour sa fiabilité. Exemple : Commande `smit` pour une interface de gestion système.
- ✓ Solaris (1992) : Développé par Sun Microsystems, célèbre pour sa gestion avancée de la mémoire et la sécurité. Exemple : Commande `dmesg` pour afficher les messages du noyau.

Avantages :

- ✓ Sécurité : Modèle de permissions strict qui protège les ressources système.
- ✓ Flexibilité : Architecture modulaire qui permet une personnalisation étendue.

Inconvénients :

- ✓ Complexité : L'interface en ligne de commande peut être difficile à maîtriser pour les utilisateurs débutants.
- ✓ Configuration : Peut nécessiter une expertise technique pour une gestion et une configuration efficaces.

Exemple: Linux, un système basé sur Unix, est utilisé dans des environnements de serveurs et de réseaux, avec des commandes comme `grep` pour rechercher du texte dans les fichiers, illustrant l'héritage de Unix dans les systèmes modernes. MacOS, également basé sur Unix, offre une interface graphique raffinée tout en bénéficiant des solides fondations de sécurité et de performance de Unix.

Système	Codage	Mono-utilisateur	Multi-utilisateur	Mono-tâche	Multitâche
DOS	16 bits	X		X	
Windows3.1	16/32 bits	X			non préemptif
Windows95/98/Me	32 bits	X			coopératif
WindowsNT/2000	32 bits		X		préemptif
WindowsXP	32/64 bits		X		préemptif
Windows7	32/64 bits		X		préemptif
Unix / Linux	32/64 bits		X		préemptif
MAC/OS X	32 bits		X		préemptif
VMS	32 bits		X		préemptif

Tableau 1 Comparaison entre différents systèmes exploitations

XXV. Conclusion

Ce chapitre introductif a posé les bases essentielles pour comprendre les systèmes d'exploitation, en explorant leur structure, leur organisation, et leur rôle fondamental dans les systèmes informatiques. Nous avons abordé les différentes perspectives d'un système, tant du point de vue de l'utilisateur que du système lui-même, ainsi que les architectures matérielles qui les soutiennent. Les systèmes d'exploitation ont été présentés comme des machines virtuelles, des gestionnaires de ressources, et des mécanismes de sécurité, tout en retraçant leur évolution historique, depuis les premiers systèmes jusqu'aux environnements modernes et cloud.

La taxonomie et la classification des systèmes d'exploitation ont fourni un aperçu des diverses approches et architectures existantes, tandis que des exemples concrets tels que MS-DOS, Windows et Unix ont illustré l'application pratique de ces concepts à travers les décennies.

Pour renforcer cette compréhension, des exercices corrigés sont proposés à la fin du chapitre, permettant d'appliquer les concepts théoriques à des situations pratiques. Ces exercices serviront de transition vers les chapitres suivants, qui approfondiront la gestion des processus et la gestion de la mémoire, des aspects cruciaux pour le fonctionnement efficace et sécurisé des systèmes d'exploitation.

XXVI. Exercices proposés

Questions de Cours

1. Qu'est-ce qu'un système informatique ?

2. Quelle est la différence entre la vue utilisateur et la vue système d'un système informatique ?

3. Quels sont les principaux types de stockage utilisés dans un système informatique?

4. Qu'est-ce que la structure des entrées/sorties (E/S) dans un système informatique ?

5. Comment fonctionnent les interruptions dans un système informatique ?

6. Quelle est la différence entre un système à processeur unique et un système multiprocesseur ?

7. Qu'est-ce qu'un système d'exploitation (SE) et à quoi sert-il ?

8. Pourquoi le système d'exploitation est-il considéré comme une machine virtuelle ?

9. Quels sont les principaux objectifs d'un système d'exploitation ?

10. Qu'est-ce que l'organisation en couches d'un système d'exploitation ?

11. Quels sont les modes de fonctionnement d'un système d'exploitation et pourquoi sont-ils importants ?

12. Comment les systèmes d'exploitation ont-ils évolué au fil du temps ?

13. Comment les systèmes d'exploitation sont-ils classifiés ?

14. Donnez trois exemples de systèmes d'exploitation et décrivez leurs caractéristiques principales.

15. Qu'est-ce que la taxonomie des systèmes d'exploitation et pourquoi est-elle importante ?

16. Quelles sont les fonctions principales d'un système d'exploitation ?

17. Comment un système d'exploitation gère-t-il la mémoire virtuelle et quels en sont les avantages ?

18. Expliquez le concept de "gestion des processus" dans un système d'exploitation.

19. Quels sont les défis de la gestion de la concurrence dans les systèmes multiprocesseurs et comment le système d'exploitation les gère-t-il ?

20. Comment le système d'exploitation assure-t-il la sécurité et la protection des données et des processus ?

QCM : Choisir la bonne réponse

1. Quelle caractéristique distincte de MS-DOS le différencie des systèmes d'exploitation modernes comme Windows ?
   - A. MS-DOS utilise une interface graphique pour l'interaction utilisateur.
   - B. MS-DOS est un système d'exploitation mono-utilisateur et monotâche.

- C. MS-DOS prend en charge le multitâche préemptif pour les applications.

2. Quel est l'un des principaux avantages de l'architecture des systèmes d'exploitation Windows par rapport à MS-DOS ?

- A. Windows prend en charge la gestion avancée de la mémoire virtuelle.
- B. Windows utilise une interface en ligne de commande uniquement.
- C. Windows élimine complètement la nécessité de pilotes matériels.

3. En quoi Unix se distingue-t-il principalement des systèmes d'exploitation comme MS-DOS et Windows ?

- A. Unix est conçu principalement pour les architectures matérielles spécifiques et ne supporte pas le réseau.
- B. Unix est un système multi-utilisateur et multitâche avec une forte orientation vers la sécurité et la programmation.
- C. Unix offre une compatibilité complète avec tous les logiciels Windows sans émulation.

4. Quelle structure de stockage est principalement utilisée pour améliorer les performances d'accès aux données ?

- A. Mémoire cache pour minimiser les accès aux disques durs.
- B. Mémoire vive (RAM) pour un accès direct aux données des applications.
- C. Mémoire tampon (buffer) pour synchroniser les transferts entre périphériques rapides et lents.

5. Quelle caractéristique des systèmes d'exploitation en temps partagé les distingue des premiers systèmes d'exploitation ?

- A. L'utilisation d'un traitement par lots pour exécuter des tâches en file d'attente.
- B. La capacité de permettre à plusieurs utilisateurs d'interagir avec le système simultanément.
- C. L'optimisation pour des applications graphiques avec des interfaces utilisateur conviviales.

6. Quel est l'avantage principal des systèmes d'exploitation organisés en couches ?

- A. Simplification de la communication entre le matériel et le logiciel.
- B. Meilleure isolation des erreurs et maintenance simplifiée du système.
- C. Accélération des processus de démarrage et d'arrêt du système.

7. Dans quel cas un système d'exploitation est-il considéré comme un mécanisme de sécurité ?

- A. Lorsqu'il protège uniquement les données des accès non autorisés.

- B. Lorsqu'il contrôle l'accès aux ressources système et protège contre les pannes matérielles.

- C. Lorsqu'il gère les droits des utilisateurs et empêche les exécutions de programmes non autorisés.

8. Quelle est la différence clé entre les systèmes d'exploitation mobiles et les systèmes d'exploitation embarqués ?

- A. Les systèmes mobiles sont optimisés pour les performances graphiques, tandis que les systèmes embarqués sont conçus pour des fonctions spécifiques.

- B. Les systèmes mobiles supportent la connectivité réseau, contrairement aux systèmes embarqués.

- C. Les systèmes embarqués offrent une interface utilisateur, alors que les systèmes mobiles n'en ont pas.

9. Quel est le rôle principal du gestionnaire de ressources dans un système d'exploitation ?

- A. Allouer les ressources mémoire uniquement lors du démarrage des applications.

- B. Assurer une utilisation équitable et efficace des ressources matérielles par les processus.

- C. Contrôler l'accès aux fichiers du système pour les utilisateurs autorisés.

10. Quel facteur différencie principalement les systèmes d'exploitation modernes des premiers systèmes à temps partagé ?

- A. L'intégration de la virtualisation et du cloud computing.

- B. L'utilisation d'un langage de programmation orienté objet pour la gestion des processus.

- C. La mise en œuvre de méthodes de sécurité basées sur l'IA et l'apprentissage machine.

Exercices

Exercice 1 : Pour appliquer les concepts de systèmes d'exploitation à des scénarios pratiques.

Instructions :

1. **Scénarios :** Considérez trois scénarios différents : (1) un utilisateur de bureau utilisant un ordinateur pour le travail quotidien, (2) un administrateur système configurant un serveur, et (3) un développeur travaillant sur un projet logiciel.
2. **Exercice :**
    - Décrivez le rôle du système d'exploitation dans chaque scénario.

- Identifiez les besoins spécifiques et les fonctionnalités du système d'exploitation qui sont particulièrement importantes dans chaque cas.

Questions :

- Quels aspects du système d'exploitation sont les plus critiques pour chaque scénario ?
- Comment le choix du système d'exploitation peut-il affecter l'efficacité et la productivité dans ces différents contextes ?

Exercice 2 : Pour étudier l'évolution des systèmes d'exploitation et comprendre les principales étapes de leur développement.

Instructions :

- Créez une chronologie des systèmes d'exploitation en mettant en évidence les grandes étapes de leur développement.
- Pour chaque étape, décrivez les caractéristiques clés et les innovations introduites.

Questions :

- Quelles sont les principales innovations qui ont marqué l'évolution des systèmes d'exploitation ?
- Comment ces innovations ont-elles influencé les capacités et les fonctionnalités des systèmes d'exploitation modernes ?

XXVII. Correction des exercices

Questions de Cours

1) Qu'est-ce qu'un système informatique ?

Correction : Un système informatique est un ensemble d'éléments matériels (hardware) et logiciels (software) qui travaillent ensemble pour effectuer des opérations de traitement de données. Le matériel inclut des composants comme le processeur (CPU), la mémoire (RAM), les dispositifs de stockage (disques durs, SSD), et les périphériques d'entrée/sortie (claviers, écrans, imprimantes). Le logiciel comprend le système d'exploitation et les applications qui permettent d'exécuter des tâches spécifiques. Un système informatique combine ces éléments pour effectuer des calculs, traiter des données, et fournir des résultats sous une forme utilisable par l'utilisateur.

2) Quelle est la différence entre la vue utilisateur et la vue système d'un système informatique?

Correction : La "vue utilisateur" concerne l'interaction de l'utilisateur avec le système informatique à travers des applications et interfaces conviviales, comme des interfaces graphiques (GUI). L'utilisateur se concentre principalement sur les applications et les services qu'il utilise sans se soucier de la gestion des ressources sous-jacentes. La "vue système", en revanche, est plus technique et s'intéresse à la manière dont le système d'exploitation gère le matériel, les ressources, et les processus pour assurer le bon fonctionnement du système. Cette vue inclut des détails tels que la gestion de la mémoire, la gestion des processus, et les interruptions.

3) Quels sont les principaux types de stockage utilisés dans un système informatique ?

Correction : Les systèmes informatiques utilisent deux principaux types de stockage : la mémoire primaire et la mémoire secondaire. La mémoire primaire, ou mémoire vive (RAM), est rapide et volatile, utilisée pour stocker temporairement les données et instructions nécessaires à l'exécution des processus en cours. La mémoire secondaire, comme les disques durs (HDD) ou les disques SSD, est non volatile et utilisée pour le stockage permanent des données et programmes. La mémoire secondaire est plus lente que la RAM, mais elle est essentielle pour conserver les données après l'arrêt du système.

4) Qu'est-ce que la structure des entrées/sorties (E/S) dans un système informatique ?

Correction : La structure des entrées/sorties (E/S) désigne les mécanismes et processus utilisés par le système informatique pour interagir avec les périphériques externes, comme les claviers, les souris, les imprimantes et les disques externes. Cette structure comprend des composants tels que les pilotes de périphériques, qui permettent la communication entre le matériel et le système d'exploitation, et les buffers, qui sont des zones de mémoire temporaires utilisées pour stocker les données en transit entre le système et les périphériques.

5) Comment fonctionnent les interruptions dans un système informatique ?

Correction : Les interruptions sont des signaux envoyés par des périphériques ou des processus pour attirer l'attention du processeur sur un événement nécessitant une action immédiate. Lorsqu'une interruption se produit, le processeur interrompt le processus en cours, sauvegarde son état, et exécute une routine de traitement d'interruption (ISR) pour répondre à l'événement. Une fois l'interruption gérée, le processeur reprend son activité précédente. Ce mécanisme est crucial pour gérer efficacement les événements asynchrones, comme les entrées de clavier ou les notifications de fin de tâche.

6) Quelle est la différence entre un système à processeur unique et un système multiprocesseur?

Correction : Un système à processeur unique dispose d'un seul CPU qui exécute toutes les tâches, une à la fois ou de manière multitâche par partage de temps. Un système multiprocesseur, quant à lui, utilise plusieurs CPU qui travaillent en parallèle pour exécuter plusieurs tâches simultanément. Cela améliore la performance globale, permet le traitement parallèle, et offre une plus grande tolérance aux pannes, car si un processeur échoue, les autres peuvent continuer à fonctionner.

7) Qu'est-ce qu'un système d'exploitation (SE) et à quoi sert-il ?

Correction : Un système d'exploitation est un logiciel essentiel qui gère les ressources matérielles et logicielles d'un ordinateur. Il sert d'interface entre le matériel de l'ordinateur et les logiciels d'application. Ses principales fonctions incluent la gestion des processus, la gestion de la mémoire, la gestion des fichiers, la gestion des périphériques, et la gestion de la sécurité et des utilisateurs. Sans un SE, il serait presque impossible pour les utilisateurs d'exploiter efficacement le matériel informatique.

8) Pourquoi le système d'exploitation est-il considéré comme une machine virtuelle ?

Correction : Le système d'exploitation est considéré comme une machine virtuelle car il crée une abstraction du matériel physique sous-jacent, offrant une interface standardisée pour les applications. Cela masque la complexité et les différences du matériel, permettant aux logiciels de fonctionner sans avoir à être modifiés pour chaque type de matériel. Cette abstraction facilite également la gestion des ressources et l'isolation des processus pour éviter les interférences.

9) Quels sont les principaux objectifs d'un système d'exploitation ?

Correction : Les principaux objectifs d'un système d'exploitation sont : (1) fournir un environnement convivial pour les utilisateurs afin de faciliter l'utilisation de l'ordinateur ; (2) gérer efficacement les ressources matérielles et logicielles pour maximiser la performance et l'efficacité ; (3) garantir la sécurité et la stabilité du système en protégeant les données et les processus contre les accès non autorisés ; (4) permettre la communication entre les différentes parties du système, y compris les utilisateurs, les processus, et les périphériques.

10) Qu'est-ce que l'organisation en couches d'un système d'exploitation ?

Correction : L'organisation en couches d'un système d'exploitation consiste à diviser ses fonctions en plusieurs niveaux, où chaque couche a une responsabilité spécifique. La couche la plus basse interagit directement avec le matériel, tandis que la couche la plus haute interagit avec les utilisateurs. Par exemple, la couche de gestion de la mémoire gère l'allocation et la libération de la mémoire, tandis que la couche de gestion des processus

contrôle l'exécution des programmes. Chaque couche ne communique qu'avec les couches adjacentes, ce qui permet une conception modulaire et une maintenance simplifiée.

11) Quels sont les modes de fonctionnement d'un système d'exploitation et pourquoi sont-ils importants ?

Correction : Les deux principaux modes de fonctionnement d'un système d'exploitation sont le mode utilisateur et le mode noyau. En mode utilisateur, les applications n'ont pas d'accès direct au matériel, ce qui empêche les applications de modifier directement les données critiques du système, réduisant ainsi les risques d'erreurs et de pannes. En mode noyau, le SE a un accès complet aux ressources matérielles, ce qui est nécessaire pour exécuter des tâches de gestion système, comme la gestion de la mémoire et des processus. Ces modes assurent une meilleure sécurité et stabilité du système.

12) Comment les systèmes d'exploitation ont-ils évolué au fil du temps ?

Correction : Les systèmes d'exploitation ont évolué pour répondre aux besoins changeants des utilisateurs et des technologies. Les premiers systèmes d'exploitation étaient simples et ne pouvaient exécuter qu'une seule tâche à la fois. Les systèmes à temps partagé ont permis l'exécution simultanée de plusieurs tâches par plusieurs utilisateurs. Avec l'avènement des interfaces graphiques, l'utilisation des ordinateurs est devenue plus accessible. Les systèmes d'exploitation modernes incluent des fonctionnalités pour les environnements mobiles, embarqués, et cloud, permettant des applications plus complexes et des services en ligne robustes.

13) Comment les systèmes d'exploitation sont-ils classifiés ?

Correction : Les systèmes d'exploitation sont classifiés selon plusieurs critères : (1) les contraintes d'utilisation, comme les systèmes en temps réel, interactifs, ou batch ; (2) les services fournis, comme la gestion des fichiers, la gestion des utilisateurs, et la sécurité ; (3) les architectures matérielles supportées, comme les systèmes à processeur unique ou multiprocesseur. Cette classification aide à comprendre les capacités et les domaines d'application de chaque type de système.

14) Donnez trois exemples de systèmes d'exploitation et décrivez leurs caractéristiques principales.

Correction : Trois exemples de systèmes d'exploitation sont : MS-DOS (système simple à ligne de commande, utilisé dans les premiers ordinateurs personnels), Windows (interface graphique conviviale, multitâche, très utilisé dans les environnements personnels et professionnels), et Unix (conçu pour les serveurs et les stations de travail, multi-utilisateurs, multitâche, hautement sécurisé). Chaque SE a évolué pour répondre à des besoins spécifiques et à des environnements différents.

15) Qu'est-ce que la taxonomie des systèmes d'exploitation et pourquoi est-elle importante ?

Correction : La taxonomie des systèmes d'exploitation est la classification des systèmes en différentes catégories basées sur leur architecture, leur mode de fonctionnement, et les types de services qu'ils offrent. Cette taxonomie est importante car elle aide à comprendre les différences fondamentales entre les systèmes d'exploitation et à choisir le SE le plus approprié pour des applications spécifiques.

16) Quelles sont les fonctions principales d'un système d'exploitation ?

Correction : Les fonctions principales d'un système d'exploitation incluent : (1) la gestion des processus (création, planification, terminaison des processus), (2) la gestion de la mémoire (allocation et libération de mémoire pour les processus), (3) la gestion des fichiers (création, suppression, modification des fichiers et répertoires), (4) la gestion des périphériques (communication avec les périphériques d'entrée/sortie), et (5) la sécurité (gestion des utilisateurs, permissions, et sécurité des données).

17) Comment un système d'exploitation gère-t-il la mémoire virtuelle et quels en sont les avantages ?

Correction : La mémoire virtuelle est une technique utilisée par le système d'exploitation pour étendre la mémoire physique disponible en utilisant une partie de l'espace disque comme mémoire supplémentaire. Le système d'exploitation gère la mémoire virtuelle en créant des pages de données qui peuvent être déplacées entre la RAM et le disque selon les besoins. Les avantages incluent la possibilité d'exécuter des programmes plus volumineux que la mémoire physique, une utilisation plus efficace de la mémoire, et une meilleure isolation des processus.

18) Expliquez le concept de "gestion des processus" dans un système d'exploitation.

Correction : La gestion des processus concerne la création, l'exécution, la planification, et la terminaison des processus dans un système. Un processus est une instance d'un programme en cours d'exécution, et le système d'exploitation doit gérer les ressources allouées à chaque processus (comme la mémoire et le temps CPU). Les mécanismes incluent la planification des processus (ordonnancement), le changement de contexte entre les processus, et la synchronisation des processus pour éviter les conflits et assurer une exécution cohérente.

19) Quels sont les défis de la gestion de la concurrence dans les systèmes multiprocesseurs et comment le système d'exploitation les gère-t-il ?

Correction : Dans les systèmes multiprocesseurs, la gestion de la concurrence est un défi majeur car plusieurs processus peuvent s'exécuter simultanément sur différents

processeurs, entraînant des problèmes tels que les conditions de course, les interblocages, et les incohérences de cache. Le système d'exploitation gère ces défis à l'aide de mécanismes comme les verrous, les sémaphores, les moniteurs, et les algorithmes de gestion de cache. Ces outils assurent une synchronisation appropriée et un partage équitable des ressources.

20) Comment le système d'exploitation assure-t-il la sécurité et la protection des données et des processus ?

Correction : La sécurité et la protection dans un système d'exploitation sont assurées par plusieurs mécanismes, y compris le contrôle d'accès basé sur les utilisateurs (permissions et autorisations), les pare-feu, les listes de contrôle d'accès (ACL), et les politiques de sécurité. Le SE utilise également la séparation des modes utilisateur et noyau pour limiter l'accès aux ressources critiques, et des techniques telles que la gestion des identités, le chiffrement des données, et la journalisation des événements pour prévenir les accès non autorisés et garantir l'intégrité du système.

QCM

1) Quelle caractéristique distincte de MS-DOS le différencie des systèmes d'exploitation modernes comme Windows ?

Réponse : B. MS-DOS est un système d'exploitation mono-utilisateur et monotâche.

2) Quel est l'un des principaux avantages de l'architecture des systèmes d'exploitation Windows par rapport à MS-DOS ?

Réponse : A. Windows prend en charge la gestion avancée de la mémoire virtuelle.

3) En quoi Unix se distingue-t-il principalement des systèmes d'exploitation comme MS-DOS et Windows ?

Réponse : B. Unix est un système multi-utilisateur et multitâche avec une forte orientation vers la sécurité et la programmation.

4) Quelle structure de stockage est principalement utilisée pour améliorer les performances d'accès aux données ?

Réponse : A. Mémoire cache pour minimiser les accès aux disques durs.

5) Quelle caractéristique des systèmes d'exploitation en temps partagé les distingue des premiers systèmes d'exploitation ?

Réponse : B. La capacité de permettre à plusieurs utilisateurs d'interagir avec le système simultanément.

6) Quel est l'avantage principal des systèmes d'exploitation organisés en couches ?

Réponse : B. Meilleure isolation des erreurs et maintenance simplifiée du système.

7) Dans quel cas un système d'exploitation est-il considéré comme un mécanisme de sécurité ?

Réponse : C. Lorsqu'il gère les droits des utilisateurs et empêche les exécutions de programmes non autorisés.

8) Quelle est la différence clé entre les systèmes d'exploitation mobiles et les systèmes d'exploitation embarqués ?

Réponse : A. Les systèmes mobiles sont optimisés pour les performances graphiques, tandis que les systèmes embarqués sont conçus pour des fonctions spécifiques.

9) Quel est le rôle principal du gestionnaire de ressources dans un système d'exploitation ?

Réponse : B. Assurer une utilisation équitable et efficace des ressources matérielles par les processus.

10) Quel facteur différencie principalement les systèmes d'exploitation modernes des premiers systèmes à temps partagé ?

Réponse : A. L'intégration de la virtualisation et du cloud computing.

Corrigés des Exercices

Exercice 1 : Scénarios d'Utilisation des Systèmes d'Exploitation

Scénarios :

1. Un utilisateur de bureau utilisant un ordinateur pour le travail quotidien.
2. Un administrateur système configurant un serveur.
3. Un développeur travaillant sur un projet logiciel.

Exercice :

1. Décrivez le rôle du système d'exploitation dans chaque scénario.
2. Identifiez les besoins spécifiques et les fonctionnalités du système d'exploitation qui sont particulièrement importantes dans chaque cas.

Réponses :

1. Scénario 1 : Utilisateur de Bureau

- Rôle du Système d'Exploitation : Le système d'exploitation (SE) permet à l'utilisateur d'exécuter des applications de bureau (traitement de texte, tableurs, navigateurs web), de gérer les fichiers et les dossiers, de configurer les périphériques (imprimantes, souris, claviers) et d'assurer la sécurité et la stabilité du système.
- Besoins Spécifiques :
    - Interface Graphique : Interface conviviale et facile à utiliser.
    - Gestion de Fichiers : Organisation et accès rapide aux documents.
    - Compatibilité : Support des logiciels courants de bureau.
    - Sécurité : Protection contre les logiciels malveillants et les attaques.

2. Scénario 2 : Administrateur Système

- Rôle du Système d'Exploitation : Le SE permet à l'administrateur de configurer et de maintenir un serveur, de gérer les utilisateurs et les permissions, de superviser les performances du serveur, et de garantir la sécurité et la fiabilité du système.
- Besoins Spécifiques :
    - Gestion des Utilisateurs : Création, modification et gestion des comptes utilisateurs et des permissions.
    - Réseau : Configuration et gestion des services réseau, comme les serveurs web, les bases de données et les services de fichiers.
    - Sécurité : Mise en œuvre des politiques de sécurité, gestion des pare-feu et des mises à jour.
    - Performances : Surveillance des performances du serveur et gestion des ressources système.

3. Scénario 3 : Développeur Logiciel

- Rôle du Système d'Exploitation : Le SE fournit un environnement de développement pour la création, le test, et le déploiement des applications. Il permet l'exécution d'outils de développement, la gestion des versions du code, et l'intégration avec d'autres systèmes et services.
- Besoins Spécifiques :
    - Support de Développement : Outils et environnements de développement (IDEs, compilateurs, débogueurs).
    - Gestion des Versions : Intégration avec des systèmes de gestion de versions (comme Git).
    - Performance : Capacité à exécuter des processus lourds tels que la compilation et les tests.
    - Compatibilité : Support des bibliothèques et des frameworks nécessaires pour le développement.

Questions :

1. Quels aspects du système d'exploitation sont les plus critiques pour chaque scénario ?

- Utilisateur de Bureau : Interface utilisateur, gestion de fichiers, sécurité, compatibilité des logiciels.
- Administrateur Système : Gestion des utilisateurs et des permissions, sécurité, gestion des ressources réseau et performances.
- Développeur Logiciel : Support de développement, outils de gestion de versions, performance, compatibilité des bibliothèques.

2. Comment le choix du système d'exploitation peut-il affecter l'efficacité et la productivité dans ces différents contextes ?

- Utilisateur de Bureau : Un SE avec une interface utilisateur intuitive et des fonctionnalités de sécurité robustes augmentera la productivité en facilitant l'utilisation des applications et la protection des données.
- Administrateur Système : Un SE offrant des outils de gestion avancés, une bonne sécurité et des capacités de surveillance améliorera l'efficacité dans la gestion des serveurs et des réseaux.
- Développeur Logiciel : Un SE avec des outils de développement appropriés, des performances élevées, et une bonne compatibilité avec les bibliothèques et frameworks accélérera le développement et le déploiement des logiciels.

Exercice 2 :

Chronologie des Systèmes d'Exploitation

1. Premiers Systèmes d'Exploitation (Années 1950-1960)

- Caractéristiques Clés :
- Systèmes de Traitement par Lots (Batch Processing) : Les premiers systèmes d'exploitation étaient conçus pour traiter des tâches en lots, où les tâches étaient collectées et traitées séquentiellement sans interaction de l'utilisateur.
- Exemples : IBM OS/360.
- Innovations :
- Introduction des concepts de gestion des tâches et des fichiers, bien que de manière très basique.
- Premiers efforts pour automatiser le traitement des données et la gestion des ressources.

2. Systèmes à Temps Partagé (Années 1960-1970)

- Caractéristiques Clés :

- Temps Partagé (Time Sharing) : Permet à plusieurs utilisateurs d'accéder simultanément au même système informatique, améliorant ainsi l'efficacité de l'utilisation du CPU.

- Exemples : UNIX, CTSS (Compatible Time-Sharing System).

- Innovations :

- Introduction du concept de multitâche, où plusieurs processus peuvent être exécutés simultanément.

- Interfaces utilisateur interactives permettant une meilleure gestion des ressources.

3. Systèmes d'Exploitation Graphiques (Années 1980)

- Caractéristiques Clés :

- Interface Graphique (GUI) : Introduction des interfaces graphiques avec des éléments visuels comme des fenêtres, des icônes, et des menus.

- Exemples : Microsoft Windows 1.0, Macintosh System 1.

- Innovations :

- Facilitation de l'interaction utilisateur avec le système via des éléments visuels plutôt que par ligne de commande.

- Meilleure gestion des applications multiples et de l'environnement utilisateur.

4. Systèmes d'Exploitation Mobiles et Embarqués (Années 2000-2010)

- Caractéristiques Clés :

- Mobilité et Connectivité : Développement de systèmes d'exploitation optimisés pour les dispositifs mobiles et embarqués avec des caractéristiques adaptées à la mobilité et à l'utilisation de réseaux sans fil.

- Exemples : Android, iOS.

- Innovations :

- Introduction de systèmes d'exploitation spécifiques aux smartphones et tablettes, avec une intégration étroite des services en ligne et des capacités multimédia.

- Gestion optimisée de la consommation d'énergie et des ressources pour les appareils portables.

5. Systèmes d'Exploitation Modernes et Cloud (Années 2010-2020)

- Caractéristiques Clés :

- Cloud Computing : Systèmes d'exploitation conçus pour le cloud computing avec une gestion centralisée des ressources et une flexibilité accrue.

- Exemples : Windows Server, Ubuntu Server, Amazon AWS.

- Innovations :

- Développement de capacités pour la virtualisation et la gestion des ressources à grande échelle.
- Intégration de fonctionnalités avancées pour le déploiement d'applications et la gestion des données en ligne.

Réponses aux Questions

1. Quelles sont les principales innovations qui ont marqué l'évolution des systèmes d'exploitation ?

- Introduction du Multitâche et du Temps Partagé : Permet aux systèmes de gérer plusieurs processus simultanément, améliorant ainsi l'efficacité des ressources.
- Interfaces Graphiques Utilisateur (GUI) : Rendent les systèmes plus accessibles et faciles à utiliser pour les utilisateurs non techniques.
- Systèmes d'Exploitation Mobiles et Embarqués : Optimisent les systèmes pour des environnements mobiles et des appareils avec des contraintes spécifiques.
- Cloud Computing et Virtualisation : Permettent la gestion des ressources et des applications à grande échelle, avec une flexibilité accrue pour le déploiement et la maintenance.

2. Comment ces innovations ont-elles influencé les capacités et les fonctionnalités des systèmes d'exploitation modernes ?

- Amélioration de l'Efficacité : Les systèmes modernes utilisent des techniques de multitâche et de temps partagé pour optimiser l'utilisation des ressources processeur et mémoire.
- Accessibilité Accrue : Les interfaces graphiques ont facilité l'interaction avec les systèmes, rendant les technologies informatiques accessibles à un plus large public.
- Mobilité et Connectivité : Les systèmes d'exploitation mobiles et cloud offrent des fonctionnalités adaptées à des environnements dynamiques et connectés, permettant une utilisation flexible des ressources.
- Gestion des Ressources à Grande Échelle : Les innovations comme la virtualisation et les services cloud permettent une gestion efficace des ressources sur des infrastructures distribuées et évolutives.

CHAPITRE 2
Gestion des Processus

I. Introduction

L'exécution des programmes dans un système informatique repose sur une série de mécanismes complexes, essentiels pour garantir un fonctionnement fluide et efficace. Ce chapitre débute par une exploration des mécanismes de base d'exécution des programmes, en abordant l'architecture des ordinateurs et les technologies sous-jacentes. Ces fondations techniques sont cruciales pour comprendre comment un programme progresse depuis son état initial jusqu'à son exécution complète au sein d'un système.

Nous examinerons ensuite les concepts de processus, une notion clé dans l'exécution des programmes. Cette section couvrira les définitions, les caractéristiques des processus, ainsi que les divers états par lesquels un processus peut passer. Un accent particulier sera mis sur le descripteur de processus (PCB), qui joue un rôle central dans la gestion et le suivi de l'exécution des processus. Les mécanismes de commutation de contexte seront également abordés, illustrant comment un système gère l'exécution concurrente de multiples processus.

Les interruptions constituent un autre aspect fondamental du fonctionnement des systèmes informatiques. Ce chapitre analysera en détail les différentes causes et types d'interruptions, ainsi que les mécanismes de gestion associés. La hiérarchie des priorités d'interruption sera discutée, soulignant l'importance de gérer efficacement les interruptions pour maintenir la stabilité et la performance du système.

Enfin, ce chapitre se concentre sur l'ordonnancement des processus, un sujet crucial pour la gestion des ressources dans un système d'exploitation. L'ordonnancement détermine l'ordre et la durée d'exécution des processus sur le processeur, influençant directement les performances globales du système. Nous explorerons les différentes politiques et algorithmes d'ordonnancement, tels que le Premier Arrivé Premier Servi (FCFS), le Plus Court d'Abord (SJF), le Round Robin, et le Scheduling avec Priorité, en analysant leurs avantages, inconvénients et applications spécifiques. Ces concepts seront illustrés à travers des exemples concrets, facilitant ainsi la compréhension des défis et des solutions liés à l'ordonnancement dans les systèmes d'exploitation modernes.

II. Mécanismes de Base d'Exécution des Programmes

L'homme a recours à l'ordinateur pour résoudre ses problèmes de manière automatisée. Ces problèmes sont traduits en programmes, qui sont ensuite exécutés par la machine matérielle. Pour qu'un programme soit interprété, exécuté, et fournisse les résultats attendus par le processeur, il doit être formulé dans un langage que la machine peut comprendre, et son exécution doit être gérée par le système d'exploitation. On commence par l'écriture structurée du problème sous forme d'algorithme, rédigé en langage humain. Pour le rendre exécutable par l'ordinateur, le programme passe par plusieurs étapes, en utilisant des outils de programmation et des outils systèmes. Pour bien comprendre les mécanismes de base employés lors de l'exécution d'un programme, il est essentiel de rappeler quelques concepts clés de l'architecture matérielle d'une machine. Ensuite, nous aborderons les différentes étapes de traitement d'un programme au sein du système, ainsi que les mécanismes utilisés par les systèmes d'interruption.

II.1 Architecture et technologie des ordinateurs

L'architecture de Von Neumann est la plus couramment utilisée, notamment dans les ordinateurs personnels que nous utilisons au quotidien. En revanche, l'architecture Harvard, qui se distingue par sa séparation des mémoires pour les instructions et les données, est souvent employée dans des appareils spécialisés, notamment dans le domaine des calculs intensifs. Dans le cadre de nos études, nous nous concentrerons principalement sur l'architecture de Von Neumann.

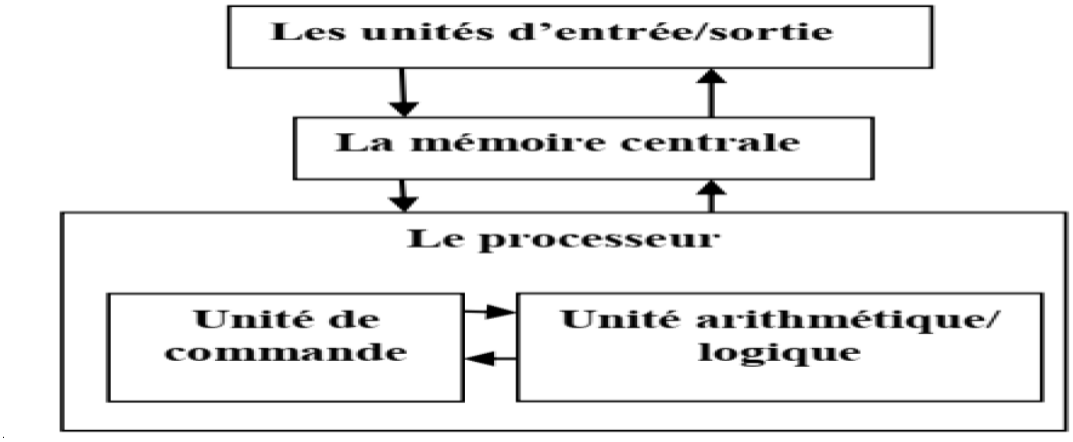


Figure 8 Machine de Von-Neumann

Une machine de Von Neumann est un type de calculateur électronique qui repose sur une mémoire centrale pour stocker à la fois les données et les instructions. Ses principaux composants sont :

- ✓ Mémoire Centrale (MC): Elle stocke les programmes et les données nécessaires à leur exécution.

- ✓ Processeur ou Unité Centrale (UC) : Chargée d'effectuer les calculs et d'exécuter les instructions stockées en mémoire. Le processeur est le cœur du système, gérant toutes les opérations logiques et arithmétiques.
- ✓ Unités périphériques ou d'Entrée/Sortie (E/S) : Elles permettent la communication entre la machine et l'extérieur, telles que les claviers, écrans, imprimantes, etc.

Dans ce modèle, la mémoire centrale est essentielle car elle contient tout ce dont le processeur a besoin pour fonctionner, tandis que les unités d'E/S facilitent l'interaction avec l'utilisateur et les autres systèmes.

L'Unité Centrale de Traitement (CPU, Central Processing Unit), souvent appelée "Processeur Central", est le cœur d'un ordinateur, responsable de l'exécution des instructions et du traitement des données. Elle se compose de plusieurs éléments essentiels, chacun jouant un rôle crucial :

L'Unité de Contrôle et de Commande (UC) : Cette unité assure le bon déroulement du traitement des instructions et contrôle l'ensemble des opérations au sein du CPU. Elle comprend plusieurs composants :

- ✓ Compteur Ordinal (CO, Program Counter (PC), Instruction Pointer (IP)) : Ce registre contient l'adresse mémoire de la prochaine instruction à exécuter, permettant ainsi le séquencement des opérations.
- ✓ Registre d'Instruction (RI, Instruction Register (IR)) : Il conserve l'instruction en cours pendant son interprétation, garantissant que le CPU traite les bonnes informations.
- ✓ Décodeur d'Instruction (DI) : Ce composant analyse le code opération de l'instruction, décomposant celle-ci en commandes élémentaires à exécuter par les différentes unités du CPU.
- ✓ Séquenceur : Il coordonne l'exécution des microcommandes en ordonnant aux différentes unités (comme la mémoire ou l'UAL) de réaliser les opérations nécessaires.
- ✓ Registre Mot d'État (PSW, Program Status Word) : Ce registre contient diverses informations de contrôle et d'état du programme en cours d'exécution, telles que les indicateurs de condition ou l'état des interruptions.

L'Unité Arithmétique et Logique (UAL) : Responsable de toutes les opérations arithmétiques (comme l'addition, la soustraction) et logiques (comparaisons, opérations booléennes). Elle est le centre de calcul du CPU, réalisant les opérations nécessaires à l'exécution des programmes.

Les Registres : Ce sont des mémoires internes rapides utilisées pour stocker temporairement des données, des adresses, des instructions, et des signaux de contrôle. Ils incluent :

- ✓ Registre de Données : Pour les données en cours de traitement.
- ✓ Registre d'Adresses : Pour les adresses mémoire en cours d'utilisation.
- ✓ Registre de Contrôle : Pour les signaux de contrôle du CPU.
- ✓ Registre d'Instruction : Pour l'instruction actuellement traitée.

Bus Interne : Il s'agit d'un réseau de communication au sein du CPU qui transporte les données, les adresses et les signaux de commande entre les différentes unités. Le bus interne permet une communication synchronisée entre les éléments du CPU, assurant une exécution fluide des instructions.

Ensemble, ces composants permettent au CPU de traiter efficacement les instructions, de gérer les données et de contrôler les opérations au sein de l'ordinateur, assurant ainsi son bon fonctionnement.

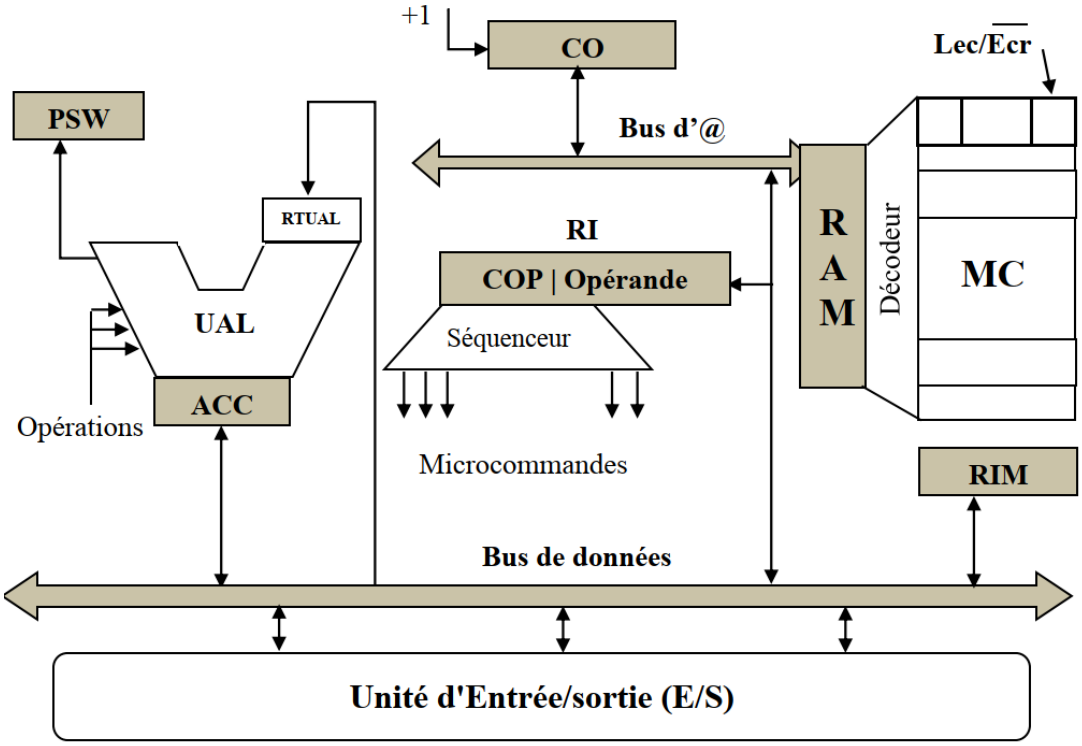


Figure 9 Cycle d'exécution d'une instruction

Le cycle d'exécution d'une instruction commence par l'initialisation du compteur ordinal (CO), qui contient l'adresse de la première instruction du programme stocké en mémoire centrale. L'ordinateur charge cette instruction dans le registre d'instruction (RI) en récupérant son contenu depuis l'adresse pointée par le CO. Ensuite, le CO est incrémenté pour pointer vers l'instruction suivante. L'instruction chargée est ensuite décodée par l'unité de commande pour déterminer l'opération à effectuer. Une fois décodée, l'instruction est exécutée, ce qui peut impliquer des opérations arithmétiques, des manipulations de données en mémoire, ou la modification de registres. Ce cycle se répète pour chaque instruction du programme jusqu'à l'achèvement de l'exécution complète.

II.2 Cheminement d'un programme dans un système

Le développement d'un programme commence par une phase d'analyse, où le problème à résoudre est défini et décomposé en sous-problèmes. Cette étape est cruciale car elle détermine la structure générale du programme et les fonctionnalités qu'il devra implémenter. Une fois cette analyse réalisée, le programmeur passe à l'écriture du code source dans un langage de programmation de haut niveau, comme Pascal, C, VB, Java, etc. Ces langages sont conçus pour être compréhensibles par les humains, offrant une syntaxe et des structures logiques qui facilitent l'expression des solutions aux problèmes identifiés. Cependant, bien que ce code soit lisible pour un programmeur, il ne peut pas être directement exécuté par un ordinateur, qui ne comprend que le langage machine, composé de 0 et de 1.

Pour transformer ce code source en un programme exécutable, une série d'étapes de transformation est nécessaire, souvent regroupées sous le terme de "chaîne de production de programme." La première de ces étapes est la compilation, où le code source est traduit en code intermédiaire ou en code objet. Cette étape est réalisée par un compilateur, un outil logiciel spécifique qui analyse le code source, détecte les erreurs de syntaxe, et génère un code qui se rapproche du langage machine, tout en restant partiellement lisible. Ensuite, le code objet est passé à un assembleur, qui le convertit en code machine. Ce code machine n'est pas encore prêt à être exécuté jusqu'à ce qu'il passe par l'étape du lien ou "linking," où différents modules de code, bibliothèques, et ressources externes sont assemblés en un seul fichier exécutable.

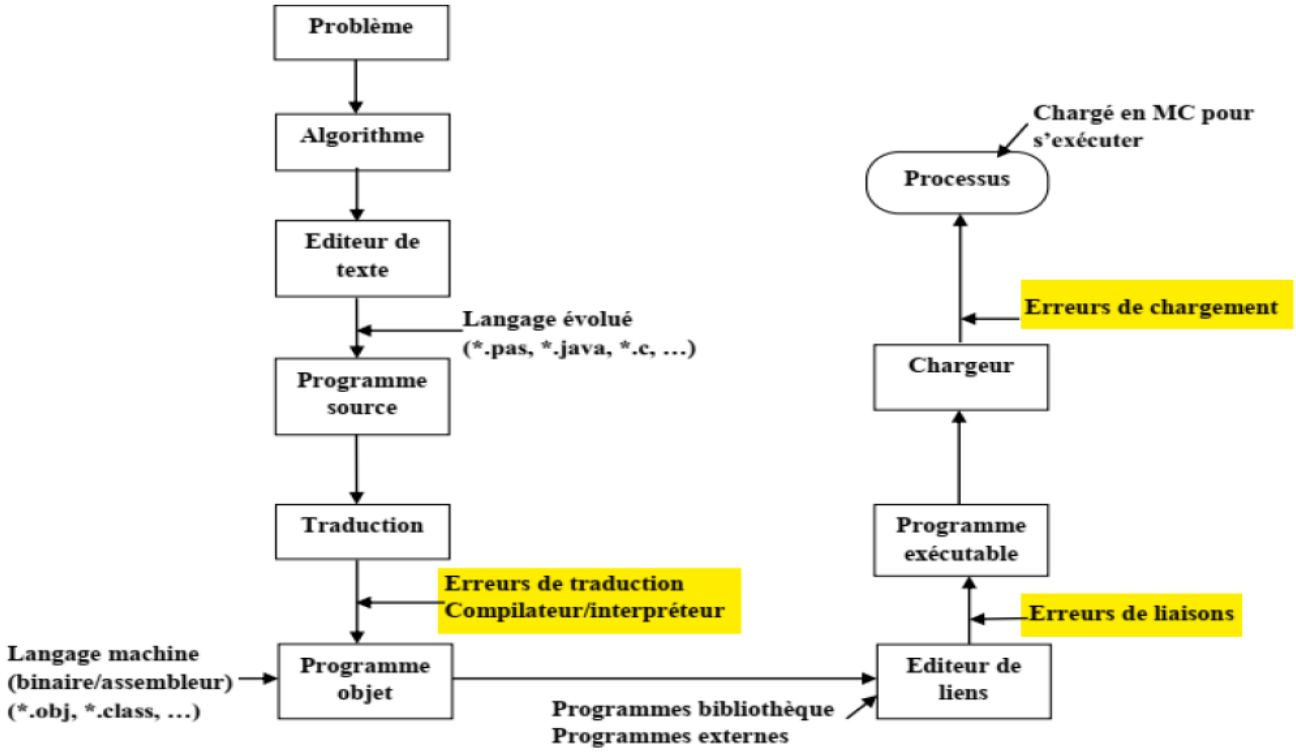


Figure 10 Cheminement d'un programme

Une fois le programme exécutable généré, il peut être lancé sur l'ordinateur en tant que processus. Un processus est une instance en cours d'exécution d'un programme, qui comprend le code du programme, ainsi que les données qu'il manipule, et les ressources qu'il utilise (comme la mémoire et les fichiers). Le système d'exploitation joue un rôle clé dans la gestion de ces processus, en fournissant les services nécessaires pour leur exécution. Il alloue de la mémoire, programme l'accès au processeur, et gère les entrées et sorties pour chaque processus en cours. L'environnement de programmation, soutenu par les services du système d'exploitation, veille à ce que chaque processus fonctionne de manière efficace, permettant au programme de résoudre le problème pour lequel il a été conçu, tout en garantissant une gestion optimale des ressources de l'ordinateur.

II.3 Concepts de processus

II.3.1 Définitions

Un processus est une instance d'un programme en cours d'exécution sur un ordinateur. Chaque processus possède des caractéristiques spécifiques, telles qu'un numéro d'identification unique, et il utilise diverses ressources, comme des fichiers ouverts. En outre, un processus peut se trouver dans différents états à tout moment, par exemple en cours d'exécution, en attente ou suspendu. On peut citer les définitions d'un processus comme suit :

1. Un processus est un programme en cours d'exécution qui est exécuté par un processeur. Aussi, plusieurs processus peuvent-ils être associés à un programme. Chaque processus possède un espace de travail en mémoire, son compteur ordinal et ses registres.
2. On peut définir un processus (process) comme un programme en exécution. Autrement dit, un programme par lui-même n'est pas un processus. Un programme est une entité passive, comme le contenu d'un fichier stocké sur disque, tandis qu'un processus est une entité active, avec un compteur d'instructions spécifiant l'instruction suivante à exécuter et un ensemble de ressources associées.

Lorsqu'un utilisateur lance l'exécution d'un programme sur une machine qui utilise la mémoire virtuelle et le multitâche, le programme suit plusieurs étapes :

1. Création du Job : Le système commence par créer un job dans la mémoire virtuelle en chargeant le programme.

2. Admission dans le Système : Ensuite, selon certains critères, ce job est admis dans le système et chargé en totalité ou en partie dans la mémoire centrale. Ce chargement transforme le job en un processus.

3. Exécution du Processus : Le processus est alors dirigé vers le processeur pour exécution. Pendant cette phase, le processus peut changer d'état plusieurs fois jusqu'à ce qu'il se termine.

II.3.2 Caractéristiques du processus

- ✓ Process Identifier : Un processus est identifié de manière unique par un numéro, généralement un entier incrémental (par exemple, 1 pour le premier processus, 2 pour le deuxième, etc.), connu sous le nom de PID (Process Identifier) . Ce numéro permet de distinguer de manière exclusive chaque processus dans le système.
- ✓ Les instructions à exécuter sont stockées dans une pile de données contenant les valeurs des variables du programme
- ✓ Un espace d'adressage (code, données, piles d'exécution);
- ✓ Un état principal (prêt, en cours d'exécution (élu), bloqué, …);
- ✓ Les valeurs des registres lors de la dernière suspension (CO, sommet de Pile...);
- ✓ Une priorité;
- ✓ Les ressources allouées (fichiers ouverts, mémoires, périphériques …);
- ✓ Les signaux à capter, à masquer, à ignorer, en attente et les actions associées;
- ✓ Autres informations indiquant le processus père, les processus fils, le groupe, les variables d'environnement, les statistiques et les limites d'utilisation des ressources...

II.3.3 Contexte d'exécution d'un processus

Le noyau du système d'exploitation maintient une table pour gérer tous les processus, où chaque processus est identifié par un PID (Process Identifier), un entier compris entre 0 et 215, attribué par le noyau. Le contexte d'un processus regroupe toutes les données nécessaires pour reprendre l'exécution d'un processus après une interruption. Ce contexte inclut :

- ✓ Son état actuel.
- ✓ Son mot d'état, qui comprend :
- ✓ La valeur des registres actifs.
- ✓ Le compteur ordinal.
- ✓ Les valeurs des variables globales, qu'elles soient statiques ou dynamiques.
- ✓ Son entrée dans la table des processus.
- ✓ La pile.
- ✓ Les zones de code et de données.

Lorsqu'un changement de processus se produit, une commutation du mot d'état et un changement de contexte sont effectués. Le noyau s'exécute alors dans le nouveau contexte.

II.3.4 Descripteur de Processus (PCB)

Chaque processus est représenté dans le système d'exploitation par un Bloc de Contrôle de Processus (PCB) . Le contenu du PCB peut varier en fonction de la complexité du système d'exploitation, mais il contient généralement les éléments suivants:

- ✓ Identité du processus : Chaque processus a deux noms pour son identification : Un nom externe, sous forme de chaîne de caractères, fourni par l'utilisateur (correspondant au

nom du fichier exécutable). Un nom interne, sous forme d'un entier attribué par le système. Ce nom interne est utilisé pour référencer le processus à l'intérieur du système pour faciliter la gestion.

- ✓ État du processus : L'état du processus peut être nouveau, prêt, en cours d'exécution, en attente, arrêté, etc.
- ✓ Contexte du processus : Il comprend le compteur ordinal, le mot d'état, et les registres associés.
- ✓ Informations sur le scheduling du processeur : Cela inclut la priorité du processus, ainsi que des pointeurs vers les files d'attente de scheduling.
- ✓ Informations sur la gestion de la mémoire : Ces informations peuvent inclure les valeurs des registres de base et de limite, ainsi que les tables de pages ou de segments.
- ✓ Informations sur l'état des E/S : Cette section couvre la liste des périphériques d'entrée/sortie alloués au processus, ainsi que la liste des fichiers ouverts.
- ✓ Informations de comptabilisation : Elles concernent l'utilisation des ressources par le processus, souvent pour la facturation du travail effectué par la machine.

Le système d'exploitation maintient une table des processus qui stocke les informations de tous les processus créés, chaque entrée étant représentée par un PCB. Cette table permet au système d'exploitation de localiser et de gérer efficacement tous les processus.

Les PCBs sont placés dans différentes files d'attente en fonction de l'état du processus :

- Le PCB d'un nouveau processus est placé dans la file d'attente des processus prêts.
- Lorsque le processus est sélectionné pour entrer dans le processeur, il devient actif, et un pointeur vers son PCB est utilisé en mémoire.
- Si le processus attend une ressource ou un événement, son PCB est déplacé vers la file d'attente des processus bloqués.
- Si le processus est suspendu, son PCB est placé dans la file d'attente des processus suspendus.

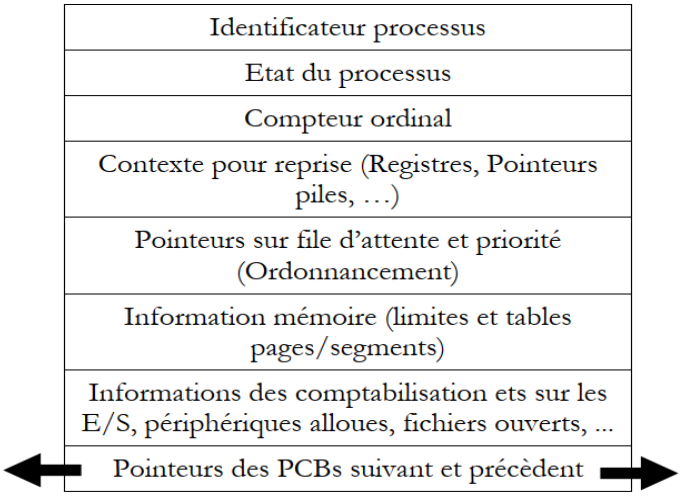


Figure 11 Bloc de Contrôle de Processus (PCB)

II.3.5 États de Processus

Les processus se partagent le processeur et passent d'un état à un autre selon qu'ils détiennent ou non le processeur. Au cours de son cycle de vie, un processus peut se trouver dans l'un des états suivants :

- ✓ Nouveau : Le processus est en cours de création.
- ✓ Actif (Élu) : Le processus dispose de toutes les ressources nécessaires, y compris le processeur, et s'exécute actuellement.
- ✓ Prêt (Éligible) : Le processus dispose de toutes les ressources nécessaires sauf le processeur, et attend donc que celui-ci soit disponible pour commencer son exécution.
- ✓ Bloqué : Le processus est en attente d'une ressource autre que le processeur ou d'un événement particulier avant de pouvoir poursuivre son exécution.
- ✓ Terminé : Le processus a achevé son exécution.

Pour les états de Prêt , Bloqué , et éventuellement Suspendu , chaque processus est associé à une file d'attente spécifique. Ces files d'attente sont gérées par des outils du système d'exploitation.

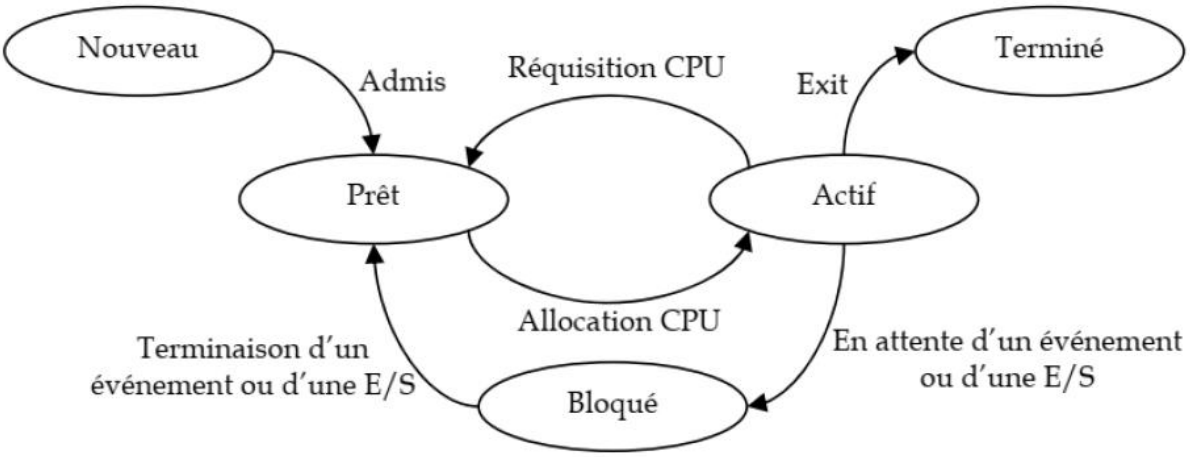


Figure 12 Diagramme des transitions d'états d'un processus

Les transitions entre ces états sont déclenchées par divers événements :

- ✓ La transition de Actif à Prêt se produit lorsque le processeur est retiré du processus, soit parce que son quantum (période d'exécution allouée) a expiré, soit en raison d'un appel système ou d'une interruption.
- ✓ La transition de Actif à Bloqué survient lorsque le processus attend une ressource autre que le processeur, comme un événement externe.
- ✓ La transition de Bloqué à Prêt a lieu lorsque le processus a acquis toutes les ressources nécessaires (à l'exception du processeur) et est prêt à reprendre l'exécution.
- ✓ La transition de Prêt à Actif se produit lorsque le processeur est alloué au processus, lui permettant de commencer ou de reprendre son exécution.

II.3.6 Processus en cours exécution.

Un processus en cours d'exécution englobe plusieurs composantes et interactions cruciales pour son fonctionnement. Le code du programme, stocké dans le segment de code, contient les instructions que le processeur doit exécuter, tandis que le compteur ordinal (CO) pointe vers la prochaine instruction à exécuter. Les registres de processus stockent des informations temporaires nécessaires à l'exécution, et la pile d'exécution gère les appels de fonctions, les adresses de retour, et les variables locales. Le processus utilise également un segment de données pour les variables globales et un segment de tas (heap) pour l'allocation dynamique de mémoire. Pendant son cycle de vie, le processus peut être dans différents états, tels qu'actif, prêt, bloqué, ou terminé, en fonction des ressources disponibles et des opérations en cours. Les opérations d'entrée/sortie (E/S) sont gérées par le système d'exploitation, et le descripteur de processus (PCB) conserve toutes les informations nécessaires pour gérer et suivre l'évolution du processus, telles que son état, son identifiant (PID), et ses ressources allouées. Ce schéma montre l'interaction complexe entre le code, les données, le processeur, et le système d'exploitation pour permettre l'exécution ordonnée et efficace d'un programme.

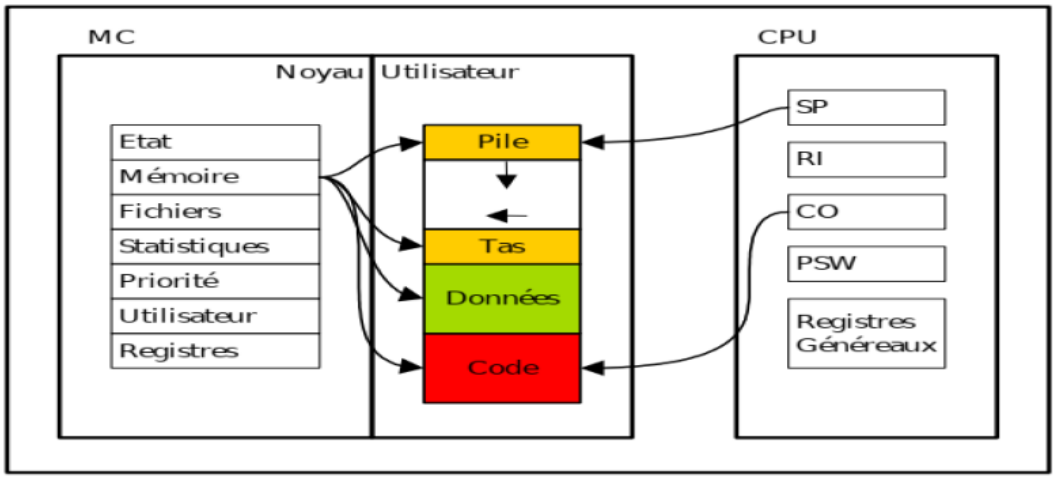


Figure 13 Schéma général d'un processus en cours exécution.

II.3.7 Mécanisme de commutation de contexte

Dans un système multiprogrammé, le processeur doit gérer l'exécution de plusieurs processus de manière quasi simultanée, ce qu'on appelle le pseudo-parallélisme. Pour passer de l'exécution d'un processus à un autre, le système doit d'abord sauvegarder l'état actuel du processus en cours, un ensemble d'informations appelé "contexte", avant de charger le contexte du nouveau processus à exécuter. Ce processus de transition est connu sous le nom de commutation de contexte (Context Switch).

La commutation de contexte est une opération clé qui permet au système d'exploitation de suspendre temporairement l'exécution d'un processus (le processus "élu") pour en sélectionner un autre, prêt à être exécuté (le processus "éligible"). Elle se produit lors de deux événements principaux : soit l'exécution d'un processus est interrompue pour diverses

raisons (comme une interruption matérielle ou une expiration de son temps d'exécution), soit un processus précédemment suspendu reprend son exécution.

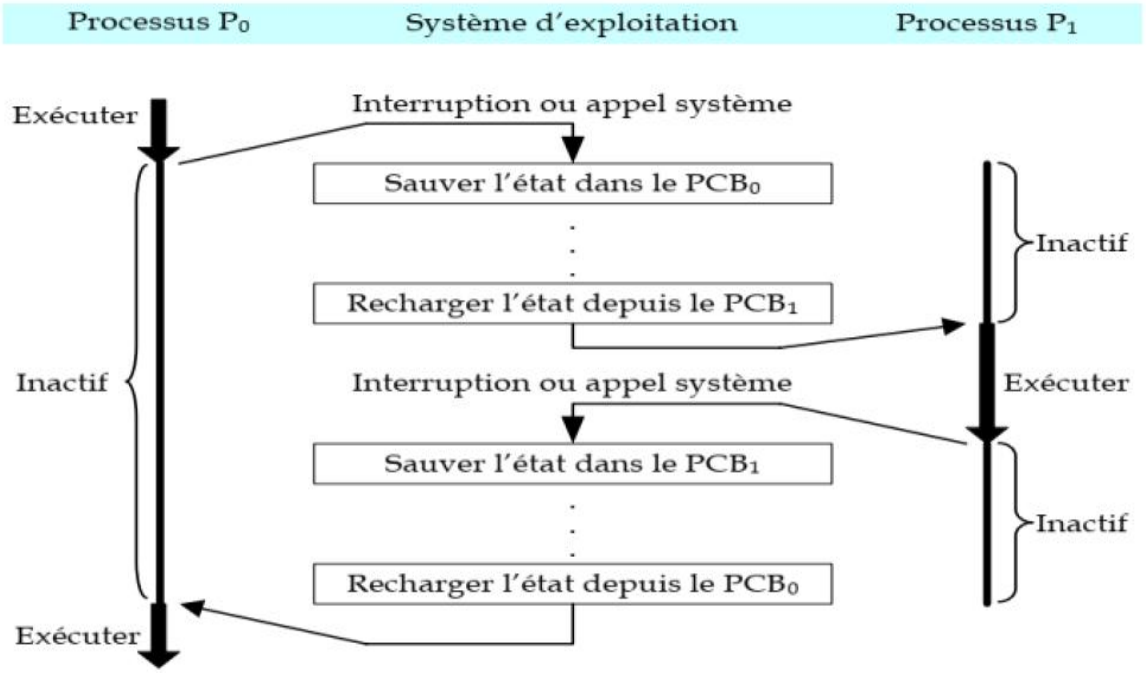


Figure 14 Mécanisme de commutation de contexte

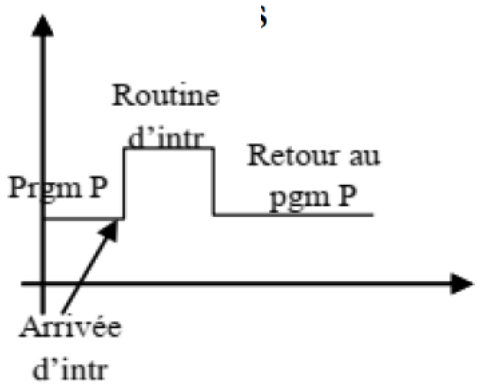
Lors de cette commutation, le système d'exploitation remplace les informations stockées dans les registres du processeur central par les données de contexte du nouveau processus, assurant ainsi que le nouveau processus reprend exactement là où il avait été interrompu. Cette capacité à changer rapidement et efficacement entre différents processus est essentielle pour maintenir l'illusion que plusieurs programmes s'exécutent simultanément sur un même processeur, optimisant ainsi l'utilisation des ressources du système.

II.4 Les interruptions

II.4.1 Définitions et Causes d'Interruptions

Les interruptions sont des signaux envoyés au processeur pour indiquer qu'un événement externe ou interne nécessite son attention immédiate.

Définition : Une interruption est un signal déclenché par un événement interne à la machine ou externe, provoquant l'arrêt d'un programme en cours d'exécution à la fin de l'opération courante, au profit d'un programme plus prioritaire appelé programme d'interruption. Ensuite, le programme interrompu reprend son exécution à l'endroit où il avait été interrompu ou un autre programme.



Lorsqu'une interruption se produit, le processeur suspend temporairement l'exécution de son programme courant pour exécuter un gestionnaire d'interruption (ISR, Interrupt Service Routine). Les causes des interruptions peuvent être variées, incluant des événements matériels tels qu'un périphérique signalant la fin d'une opération d'E/S (entrée/sortie), des erreurs matérielles comme une division par zéro, ou encore des interruptions logicielles déclenchées par des instructions spécifiques comme `INT` en assembleur.

II.4.2 Types et Mécanismes de Gestion d'Interruption

Il existe principalement deux types d'interruptions : les interruptions matérielles et les interruptions logicielles. Les interruptions matérielles sont générées par des périphériques externes (comme un clavier ou une carte réseau) tandis que les interruptions logicielles sont générées par le logiciel en cours d'exécution, souvent pour effectuer des appels système.

Quand un processus utilisateur P est en cours d'exécution, dans un temps T, et le processeur reçoit une interruption quelconque plus prioritaire, alors :

1) Suspendre le process P en cours et sauvegarder son contexte dans une pile pouvoir reprendre son exécution ultérieurement. Son contexte contient :

i. Adresse de la prochaine instruction à exécuter dans le programme interrompu,
ii. Contenu des registres qui seront modifiés par le programme d'interruption,
iii. Contenu du mot d'état (registre de drapeaux) rassemblant les indicateurs (tout cela forme le contexte sauvegardé)

2) Identifier la cause d'interruption : pour détecter la source émettrice du signal d'IT.

3) Chargement du contexte du programme d'interruption (contexte actif) et passage en mode système (ou superviseur)

4) Exécuter le programme de traitement de cette IT (RIT).

5) Restaurer le contexte du programme P suspendu et en repassant en mode utilisateur ou le choix d'un autre, et continuer son exécution.

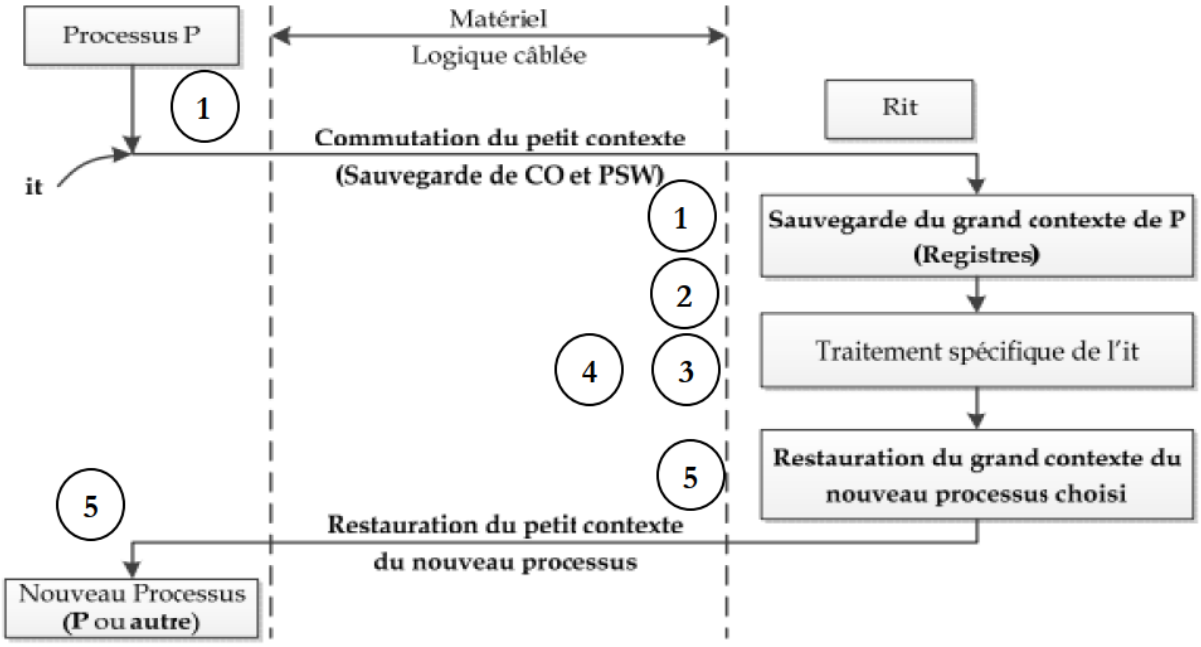


Figure 15 Mécanisme de gestion des interruptions

Ces étapes permettent au système de réagir rapidement aux événements tout en assurant que les programmes en cours ne sont pas perturbés de manière indésirable. Ce mécanisme garantit non seulement la réactivité du système, mais aussi la stabilité et l'intégrité des processus en cours d'exécution.

II.4.3 Priorité et Hiérarchie

Dans un système où plusieurs interruptions peuvent se produire simultanément, il est crucial de gérer la priorité et la hiérarchie des interruptions. Chaque interruption est généralement associée à un niveau de priorité, permettant au processeur de déterminer laquelle doit être traitée en premier. Par exemple, une interruption critique comme une panne matérielle peut avoir une priorité plus élevée qu'une interruption signalant la réception d'une donnée réseau. La hiérarchie des interruptions peut aussi inclure des mécanismes de masquage, où des interruptions de priorité inférieure peuvent être temporairement ignorées pendant le traitement d'une interruption plus urgente. Cette gestion de la priorité et de la hiérarchie garantit que les événements critiques sont traités en temps voulu, minimisant les risques d'erreurs ou de perte de données.

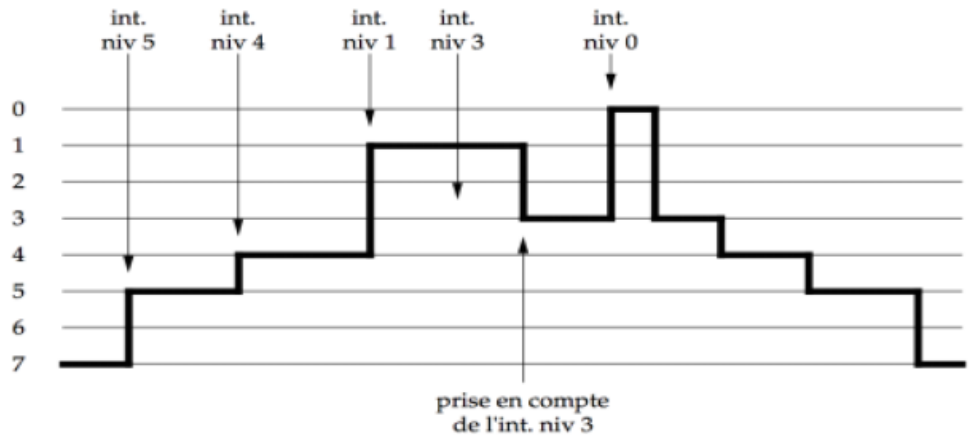


Figure 16 Hiérarchie des interruptions

III. Gestion des processus

III.1 Introduction

Dans un système multiprocesseur, plusieurs processus peuvent être en concurrence pour accéder au temps processeur. Cette situation se produit lorsque deux ou plusieurs processus sont simultanément dans l'état « prêt », c'est-à-dire qu'ils sont prêts à être exécutés dès que le processeur devient disponible. La figure 9 illustre le fonctionnement d'une telle machine multiprocessus. Plusieurs processus sont chargés en mémoire centrale : le processus P1 est actuellement élu et s'exécute sur le processeur, tandis que les processus P2 et P4 sont dans un état bloqué, car ils attendent la fin d'une opération d'entrée/sortie avec le disque.

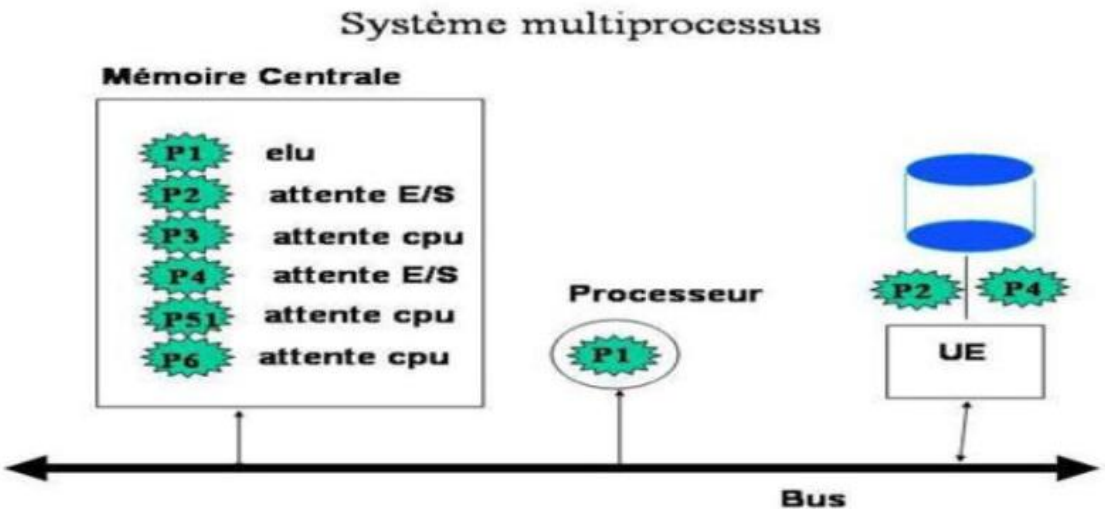


Figure 17 Fonctionnement d'une machine multiprocessus

Pendant ce temps, les processus P3, P5 et P6 sont dans l'état « prêt » : bien qu'ils disposent de toutes les ressources nécessaires pour s'exécuter, ils ne peuvent pas le faire car le processeur est déjà occupé par P1. Lorsque P1 aura terminé son exécution et libérera le

processeur, les processus P3, P5 et P6 seront tous éligibles pour obtenir le processeur. Cependant, puisque le processeur ne peut être alloué qu'à un seul processus à la fois, il sera nécessaire de choisir lequel des trois processus prêts recevra le processeur. C'est ici qu'intervient l'ordonnanceur, ou scheduler, qui jouera son rôle en sélectionnant l'un des processus P3, P5 ou P6 pour prendre le relais de P1 sur le processeur. Ce mécanisme de choix et d'allocation est essentiel pour garantir une exécution ordonnée et efficace des processus dans un environnement multitâche.

III.2 L'ordonnancement (Scheduling)

Définition : La fonction d'ordonnancement est responsable de la gestion du partage du processeur parmi les différents processus qui attendent de s'exécuter, c'est-à-dire ceux qui sont dans l'état « prêt ». L'opération de sélection, ou élection, consiste à attribuer le processeur à l'un de ces processus. Chaque fois que le processeur devient inactif, le système d'exploitation doit choisir un processus de la file d'attente des processus prêts et lui transférer le contrôle du processeur. Concrètement, cette tâche est effectuée par deux routines système spécifiques : le Dispatcheur et le Scheduleur (ou Ordonnanceur).

- ✓ **Le Dispatcheur** : Il s'occupe de l'allocation du processeur à un processus sélectionné par l'Ordonnanceur du processeur. Une fois allouer, le processeur doit réaliser les tâches suivantes:
    - Commutation de contexte : sauvegarder le contexte du processus qui doit relâcher le processeur et charger le contexte de celui qui aura le prochain cycle processeur.
    - Commutation du mode d'exécution : basculer du mode Maître (mode d'exécution du dispatcheur) en mode utilisateur (mode d'exécution du processeur utilisateur).
    - Branchement : se brancher au bon emplacement dans le processus utilisateur pour le faire démarrer.
- ✓ **L'Ordonnanceur** : Certains systèmes d'exploitation utilisent une technique d'ordonnancement à deux niveaux qui intègre deux types d'Ordonnanceurs :
    - Ordonnanceur du processeur : c'est un Ordonnanceur court terme opère sur une ensemble du processus présents en mémoire. Il s'occupe de la sélection du processus qui aura le prochain cycle processeur, à partir de la file d'attente des processus prêts.
    - Ordonnanceur de travail : ou Ordonnanceur long terme, utilisé en cas d'insuffisance de mémoire, son rôle est de sélectionné le sous ensemble de processus stockés sur un disque et qui vont être chargés en mémoire. Ensuite, il retire périodiquement de la mémoire les processus qui sont restés assez longtemps et les remplace par des processus qui sont sur le disque depuis trop de temps.

Lorsqu'il y a plusieurs processus prêts à s'exécuter, comment décide-t-on lequel doit recevoir le processeur ? Un processus en attente doit-il nécessairement attendre la fin de l'exécution du processus actif ? Que se passe-t-il si un processus prioritaire ou urgent demande l'accès au processeur ? Doit-il patienter comme les autres ? Quelles stratégies ou politiques peuvent être employées pour allouer le processeur de manière efficace ? Enfin, comment optimiser l'utilisation du processeur pour maximiser les performances du système informatique ? Toutes ces questions soulignent la nécessité d'un ordonnanceur, un composant essentiel pour gérer l'allocation du CPU de manière équitable et efficace, tout en assurant une performance optimale du système. Donc on peut déduire la définition suivante :

Définition : L'ordonnanceur, appelé Scheduler en anglais, est un programme essentiel du système d'exploitation. Sa fonction principale est de choisir un processus parmi ceux qui sont prêts à s'exécuter, en utilisant une politique ou un algorithme d'ordonnancement spécifique, puis de lui attribuer le processeur. Sur le plan ordonnancement, l'ordonnanceur élit un processus prêt selon un algorithme qui doit garantir :

- ✓ L'Équité :L'algorithme d'ordonnancement doit garantir l'équité en répartissant le temps processeur de manière juste entre les processus. Chaque processus doit avoir l'opportunité de s'exécuter, sans qu'aucun ne soit favorisé ou désavantagé de manière injuste.
- ✓ L'Efficacité : L'ordonnanceur doit maximiser l'efficacité du système en s'assurant que le processeur est utilisé de manière optimale. Le but est de maintenir un taux d'utilisation élevé du processeur, minimisant ainsi les périodes d'inactivité et maximisant les ressources disponibles.
- ✓ Temps Moyen d'Exécution Minimal : L'algorithme d'ordonnancement doit également viser à réduire le temps moyen d'exécution . Cela signifie que le temps que les processus passent en attente avant d'être traités doit être aussi court que possible, ce qui contribue à une gestion plus fluide et plus rapide des tâches.
- ✓ Rendement Maximal :Enfin, l'ordonnanceur doit maximiser le rendement du système, c'est-à-dire le nombre de tâches ou de processus complétés dans un laps de temps donné. Un rendement élevé indique que le système est capable de traiter efficacement un grand nombre de processus, optimisant ainsi les performances globales.

Ces points mettent en lumière l'importance de l'ordonnanceur dans la gestion des ressources du processeur, assurant à la fois une utilisation efficace des ressources et une répartition équitable du temps de calcul entre les processus.

III.3 Les Différentes Politiques d'ordonnanacement (Scheduling)

Les systèmes d'exploitation utilisent principalement deux types de politiques d'ordonnancement pour gérer l'exécution des processus : l'ordonnancement préemptif (avec réquisition) et l'ordonnancement non préemptif (sans réquisition).

III.3.1 Le Scheduling Non Préemptif

Dans un algorithme d'ordonnancement non préemptif, un processus est sélectionné pour s'exécuter et continue à s'exécuter jusqu'à ce qu'il se bloque, par exemple en attendant une opération d'entrée/sortie (E/S) ou un autre processus, ou qu'il libère volontairement le processeur. Même si ce processus s'exécute pendant une longue durée, il ne sera pas interrompu de force. Dans ce type d'ordonnancement, aucune décision n'est prise lors des interruptions d'horloge. Une fois l'interruption traitée, le processus qui était en cours avant l'interruption reprend son exécution. Cela signifie que le processus a un contrôle total sur le processeur tant qu'il ne se bloque pas ou ne termine pas son exécution.

III.3.2 Le Scheduling Préemptif

Dans un algorithme d'ordonnancement préemptif, un processus est sélectionné pour s'exécuter, mais seulement pour un délai prédéterminé. Si le processus n'a pas terminé son exécution à la fin de ce délai, il est suspendu et déplacé dans la file d'attente des processus prêts, permettant ainsi à l'ordonnanceur de choisir un autre processus à exécuter. Ce type d'ordonnancement repose sur des interruptions qui, une fois le délai écoulé, rendent le contrôle de l'unité centrale (UC) à l'ordonnanceur. Cela permet de garantir que le processeur est partagé de manière équitable entre les différents processus, évitant ainsi qu'un processus monopolise le processeur pendant une durée excessive.

III.4 Critères d'évaluation de performances entre les algorithmes d'ordonnancement

L'objectif principal d'un algorithme d'ordonnancement est de sélectionner le processus qui optimisera la performance globale du système. Cette évaluation repose sur différents critères, dont l'importance peut varier en fonction des besoins spécifiques du système. La politique d'ordonnancement définit la priorité de chaque critère, et plusieurs algorithmes éprouvés sont utilisés pour mettre en œuvre ces politiques. Voici une revue des critères d'ordonnancement fréquemment utilisés :

- ✓ Utilisation de l'UC :Ce critère mesure le pourcentage de temps pendant lequel l'unité centrale (UC) exécute un processus. Une utilisation élevée de l'UC indique que le processeur est bien utilisé, ce qui est souvent souhaité pour éviter les périodes d'inactivité. Par exemple, si l'UC est occupée 90% du temps, cela signifie qu'elle est largement utilisée pour traiter les processus.
- ✓ Utilisation Répartie : Ce critère évalue le pourcentage de temps pendant lequel toutes les ressources du système, y compris l'UC, la mémoire, et les périphériques

d'entrée/sortie, sont utilisées. Une bonne utilisation répartie assure que toutes les ressources sont employées efficacement. Par exemple, un système avec une utilisation répartie de 80% utilise non seulement l'UC mais aussi la mémoire et les périphériques de manière équilibrée.

- ✓ Débit : Le débit représente le nombre de processus que le système peut exécuter dans une période de temps donnée. Un débit élevé signifie que le système est capable de traiter de nombreux processus rapidement. Par exemple, un débit de 50 processus par minute indique une capacité élevée à gérer les tâches.
- ✓ Temps de Rotation : Le temps de rotation (de sejour) est la durée moyenne nécessaire pour qu'un processus soit entièrement exécuté, depuis son arrivée jusqu'à sa fin. Ce temps inclut tout le temps que le processus passe dans le système, que ce soit en attente, en exécution ou en blocage. Un temps de rotation court est généralement souhaité pour améliorer l'efficacité du système. Par exemple, un temps de rotation de 20 ms signifie que les processus sont terminés rapidement.

$$\mathbf{TRM} = \sum_{i=1}^{n} TR_i/n \text{ , avec } TR_i = date\ fin - date\ arrivée$$

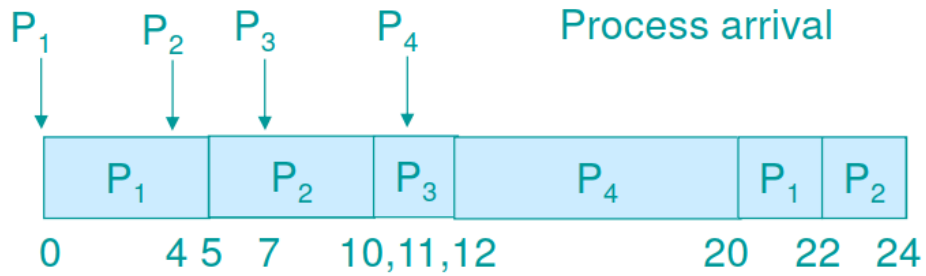
- ✓ Temps d'Attente : Le temps d'attente est la durée moyenne pendant laquelle un processus reste dans la file d'attente des processus prêts avant de commencer son exécution. Contrairement au temps de rotation, le temps d'attente se concentre uniquement sur le délai d'attente. Par exemple, si un processus passe en moyenne 10 ms en attente avant de commencer à s'exécuter, cela peut indiquer une gestion efficace ou inefficace des processus en fonction du contexte.

$$\mathbf{TAM} = \sum_{i=1}^{n} TA_i/n \text{ , avec } TA_i = TR_i - temps\ d'exécution$$

- ✓ Temps de Réponse Le temps de réponse est le temps moyen qu'il faut au système pour commencer à répondre aux entrées de l'utilisateur. Il mesure la réactivité du système. Par exemple, un temps de réponse de 50 ms signifie que le système commence à traiter les entrées de l'utilisateur après 50 millisecondes.
- ✓ Équité : L'équité évalue dans quelle mesure tous les processus reçoivent une chance égale de s'exécuter. Un système équitable assure que chaque processus obtient une part raisonnable du temps processeur, sans favoriser injustement certains processus.
- ✓ Priorités : Ce critère attribue un traitement préférentiel aux processus en fonction de leur niveau de priorité. Les processus ayant une priorité plus élevée sont exécutés avant ceux de priorité inférieure. Par exemple, un processus critique avec une priorité élevée pourrait être exécuté immédiatement, tandis qu'un processus moins urgent attendra.

Ces critères permettent d'évaluer et d'ajuster les politiques d'ordonnancement pour répondre aux besoins spécifiques du système et améliorer la performance globale.

Exemple :



- ✓ Utilisation de l'UCT = 24/24 = 100%
- ✓ Temps de réponse:

P1: (0 – 0) = 0
P2: (5 – 4) = 1
P3: (10 – 7) = 3
P4: (12 – 11) = 3

- ✓ Débit =4/24 = 0.17 processus/ms
- ✓ Temps de rotation (de sejour):

P1: TR1=(22 – 0) = 22
P2: TR2=(24 – 4) = 20
P3: TR3=(12 – 7)= 5
P4: TR4=(20 – 11)= 9
Temps de rotation moyen : TRM= (22+20+5+9)/4 = 14

- ✓ Temps d'attente:

P1: TA1=(22 – 7) = 15
P2: TA2=(20 – 7) = 13
P3: TA3=(5 – 2)= 3
P4: TA4=(9 – 8)= 1
Temps d'attente moyen : TAM= (15+13+3+1)/4 = 8

III.5 Diagrammes de Gantt

Pour illustrer l'évolution temporelle des processus dans un système d'exploitation, les diagrammes de Gantt sont couramment utilisés. Un diagramme de Gantt est un outil graphique qui affiche les processus sur l'axe des abscisses et le temps sur l'axe des ordonnées. Chaque processus est représenté par une barre horizontale dont la longueur correspond à la durée d'exécution du processus, et sa position sur l'axe des abscisses indique le moment précis de début et de fin. Cette représentation visuelle permet de suivre

clairement l'allocation du processeur et d'autres ressources au fil du temps, facilitant ainsi l'analyse des politiques d'ordonnancement et l'évaluation des performances du système, telles que les temps d'attente et de réponse.

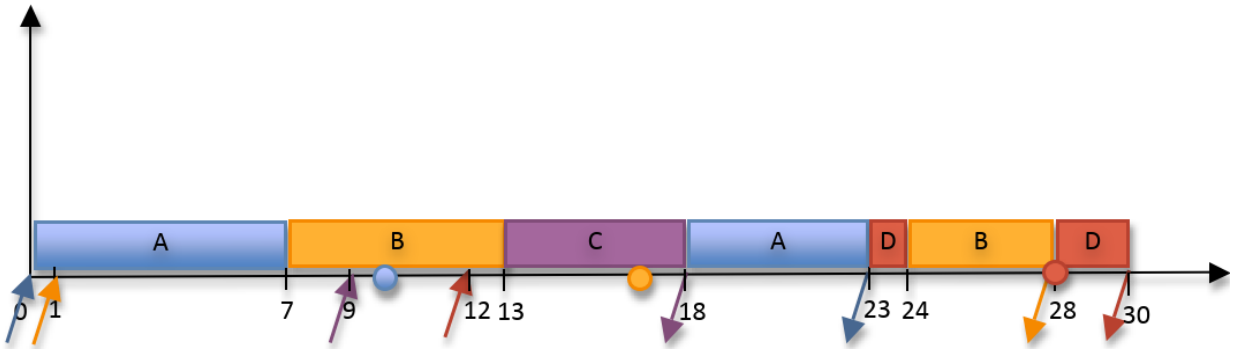


Figure 18 diagrammes de Gantt 1

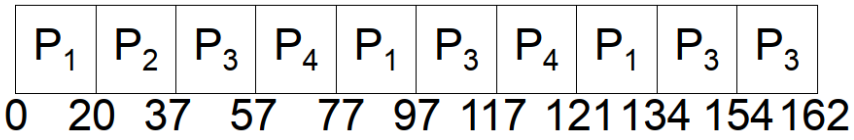


Figure 19 diagrammes de Gantt 2

IV. Les algorithmes d'ordonnancement (scheduling)

IV.1 Algorithme du Premier Arrivé Premier Servi (FCFS)

L'algorithme du Premier Arrivé Premier Servi (First Come First Served, FCFS) est l'un des algorithmes d'ordonnancement les plus élémentaires. Avec cet algorithme, le processeur est attribué au processus qui se présente en premier. La mise en œuvre de FCFS est réalisée à l'aide d'une file d'attente FIFO (First In, First Out). Lorsqu'un processus rejoint la file d'attente des processus prêts, son Bloc de Contrôle de Processus (PCB) est ajouté à la fin de la file. Dès que le processeur se libère, il est accordé au processus qui se trouve en tête de la file d'attente.

Exemple : Considérons trois processus, P1, P2 et P3, qui arrivent dans cet ordre au système avec des durées d'exécution de 24, 3 et 3 unités de temps respectivement. Pour illustrer l'occupation du processeur dans le temps, on utilise un diagramme de Gantt, qui montre clairement comment le processeur est alloué aux processus en fonction de leur ordre d'arrivée.

Le diagramme de Gantt révèle que le processus P1 utilise le processeur de l'instant 0 jusqu'à 24. À ce moment-là, le processeur passe au processus P2, qui s'exécute jusqu'à 27, puis c'est au tour de P3. En termes de temps d'attente, P1 n'attend pas, donc son temps

d'attente est de 0 unités de temps. En revanche, le processus P2 attend 24 unités de temps avant de commencer son exécution, tandis que P3 attend 27 unités de temps. Ainsi, le temps d'attente moyen est calculé comme suit : (0 + 24 + 27) / 3, ce qui donne 17 unités de temps.

Si les processus étaient arrivés dans l'ordre P2, P3, et P1, les temps d'attente et les résultats globaux seraient différents, illustrant ainsi l'impact significatif de l'ordre d'arrivée sur les performances de l'algorithme FCFS.

Avec une politique FCFS, si les processus arrivent dans l'ordre P2, P3, puis P1, le temps d'attente serait calculé comme suit : P2 commence immédiatement à 0 et termine à 3, donc son temps d'attente est de 0 unités de temps. P3 commence après P2, à 3, et termine à 6, donc son temps d'attente est de 3 unités de temps (temps d'attente = temps de début - temps d'arrivée). P1 commence après P3, à 6, et termine à 30, donc son temps d'attente est de 6 unités de temps.

Le temps d'attente moyen est donc : (0 + 3 + 6) /3 = 3 unités de temps. Ainsi, le temps moyen d'attente avec l'algorithme FCFS peut ne pas être optimal et peut varier considérablement en fonction des durées d'exécution des processus et de leur ordre d'arrivée.

Critique de la Méthode FCFS : L'algorithme du Premier Arrivé Premier Servi (FCFS) présente des inconvénients notables, surtout pour les systèmes où les durées d'exécution des processus varient largement. L'une des principales critiques de FCFS est qu'il tend à pénaliser les travaux courts. En effet, FCFS ne permet pas de réquisition ; une fois qu'un processus a pris possession du processeur, il le conserve jusqu'à ce qu'il soit terminé ou qu'il demande une opération d'entrée/sortie (E/S). Ce comportement peut conduire à une situation où un processus long bloque le processeur pendant une période prolongée, empêchant ainsi les processus plus courts d'accéder rapidement au processeur. Cette caractéristique est particulièrement problématique dans les systèmes à temps partagé, où il est crucial que les utilisateurs obtiennent un accès au processeur à intervalles réguliers. Le fait de laisser un processus long monopoliser le processeur peut réduire l'efficacité globale du système et provoquer des retards significatifs pour les processus plus courts, rendant l'algorithme FCFS inadapté pour des environnements où une répartition équitable et réactive du temps processeur est essentielle.

IV.2 Algorithme du Plus Court d'abord (SJF)

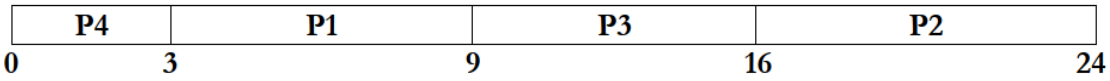
L'algorithme du Plus Court d'abord (Shortest Job First, SJF) est une méthode d'ordonnancement qui attribue le processeur au processus ayant la durée d'exécution la plus courte. Si plusieurs processus ont des durées d'exécution identiques, une politique FIFO (First In, First Out) est utilisée pour les départager. Ce critère d'ordonnancement vise à

minimiser le temps d'attente moyen en traitant d'abord les tâches les plus courtes, ce qui est souvent plus efficace dans des environnements où les processus varient considérablement en durée.

Exemple : Considérons un système avec quatre processus, P1, P2, P3, et P4, ayant les durées d'exécution suivantes :

Processus	Durée d'exécution
P1	6
P2	8
P3	7
P4	3

Les processus seront exécutés selon la durée d'exécution la plus courte en premier. L'algorithme du travail le plus court donnera alors le résultat suivant :



Le temps moyen d'attente est = (0+3+9+16)/4=7. Alors que si on avait choisi une politique FCFS, le temps moyen serait de : 10.25 unités de temps. Ce calcul démontre que l'algorithme SJF réduit le temps moyen d'attente par rapport à FCFS, offrant une meilleure performance en termes de temps d'attente moyen lorsqu'il s'agit de traiter des processus avec des durées d'exécution variées.

Critique de l'Algorithme SJF : L'algorithme du Plus Court d'abord (SJF) est reconnu comme optimal en termes de minimisation du temps d'attente moyen pour un ensemble de processus donné. En effet, en attribuant le processeur aux processus les plus courts en premier, SJF garantit que le temps moyen d'attente est minimisé, ce qui est un avantage significatif en termes d'efficacité. Cependant, la principale difficulté de l'algorithme SJF réside dans son implémentation pratique. Le défi majeur est la nécessité de connaître à l'avance le temps d'exécution de chaque processus. En réalité, il est souvent impossible de prévoir avec précision combien de temps un processus prendra pour s'exécuter, surtout si le processus est dynamique ou dépend de facteurs externes comme les entrées/sorties ou les variations dans la charge de travail. Cette imprévisibilité rend l'application de SJF difficile, car l'algorithme repose sur une connaissance préalable qui n'est généralement pas disponible.

IV.3 Scheduling avec Priorité

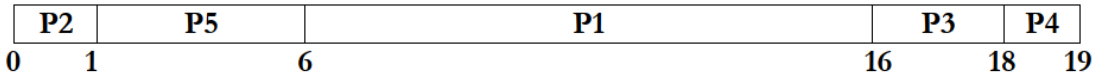
L'algorithme de scheduling avec priorité attribue un niveau de priorité à chaque processus, et le processeur est alloué en priorité au processus ayant la plus haute priorité. Dans cet algorithme, une priorité plus élevée est généralement représentée par une valeur numérique plus élevée, bien que les conventions puissent varier selon les systèmes. Les

priorités peuvent être définies en fonction de divers critères tels que le type de processus, les contraintes de temps, et les besoins en mémoire.

Exemple : Considérons un système avec les processus suivants, chacun ayant une durée d'exécution et une priorité associée :

Processus	Durée d'exécution	Priorité
P1	10	2
P2	1	4
P3	2	2
P4	1	1
P5	5	3

Diagramme de Gantt :



Le temps moyen d'attente est = (0+1+6+16+18)/5=8.2 unités de temps.

Critique de la Méthode : L'algorithme de scheduling avec priorité est efficace pour garantir que les processus les plus importants obtiennent le processeur en premier. Cependant, L'algorithme de scheduling avec priorité peut rencontrer des problèmes notables, notamment la famine des processus de faible priorité. Cette situation se produit lorsque les processus ayant une faible priorité attendent indéfiniment le processeur, car de nouveaux processus de haute priorité continuent d'arriver et de recevoir l'accès au processeur en priorité. En conséquence, les processus de basse priorité peuvent rester bloqués dans la file d'attente sans jamais avoir l'opportunité de s'exécuter. Pour mitiger ces problèmes, des techniques supplémentaires comme l'inversion de priorité, le vieillissement des priorités, ou des ajustements dynamiques des priorités peuvent être mises en place pour assurer une allocation équitable des ressources.

La technique du vieillissement est souvent employée. Cette méthode consiste à augmenter progressivement la priorité des processus qui attendent depuis longtemps. Par exemple, on pourrait décider d'incrémenter la priorité d'un processus en attente toutes les 15 minutes. Ainsi, un processus dont la priorité est initialement basse verra sa priorité augmenter avec le temps. Finalement, même un processus ayant commencé avec une priorité très faible, comme 0, pourra atteindre la plus haute priorité après une période suffisamment longue et sera ainsi exécuté. Cette approche aide à garantir que tous les processus, indépendamment de leur priorité initiale, auront une chance d'obtenir du temps processeur, réduisant ainsi les risques de famine et améliorant l'équité du système.

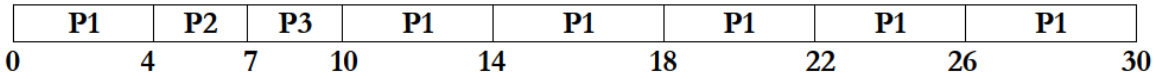
IV.4 Algorithme de Round Robin (Tourniquet)

L'algorithme de scheduling Round Robin, également connu sous le nom de tourniquet, est particulièrement adapté aux systèmes à temps partagé. Il fonctionne en allouant le

processeur aux processus de manière cyclique, chaque processus recevant une tranche de temps fixe appelée *quantum*. Ce quantum est généralement compris entre 10 et 100 millisecondes. L'idée est que chaque processus obtient une opportunité d'exécution pour une période déterminée avant de passer au processus suivant dans la file d'attente.

Exemple : On dispose de 3 processus P1, P2 et P3 ayant comme durée d'exécution, respectivement 24, 3 et 3 ms. En utilisant un algorithme Round Robin, avec un quantum de 4 ms, Le diagramme de Gantt pour ce système avec l'algorithme Round Robin serait le suivant :

Diagramme de Gantt :



Le temps moyen d'attente est de : (6+4+7)/3 = 17/3 = 5.66 ms

Critique de l'Algorithme Round Robin : La performance de l'algorithme Round Robin est fortement influencée par la taille du quantum. Voici comment le choix du quantum affecte la méthode :

- ✓ Quantum Très Grand : Lorsque le quantum est très grand, l'algorithme Round Robin se rapproche du comportement de l'algorithme FCFS (Premier Arrivé, Premier Servi). En effet, le processus qui obtient le processeur reste en exécution pendant une période prolongée, ce qui peut réduire l'avantage de l'équité offerte par Round Robin et ne permet pas un partage efficace du processeur entre plusieurs processus.
- ✓ Quantum Très Petit : À l'inverse, un quantum très petit peut offrir une sensation de réactivité accrue, donnant à chaque utilisateur l'impression de disposer d'un processeur dédié. Cependant, un quantum trop court entraîne une augmentation du nombre de commutations de contexte. Cela peut engendrer une surcharge du système, car les processus passent plus souvent par le processus de sauvegarde et de restauration de l'état, ce qui réduit l'efficacité globale du système.

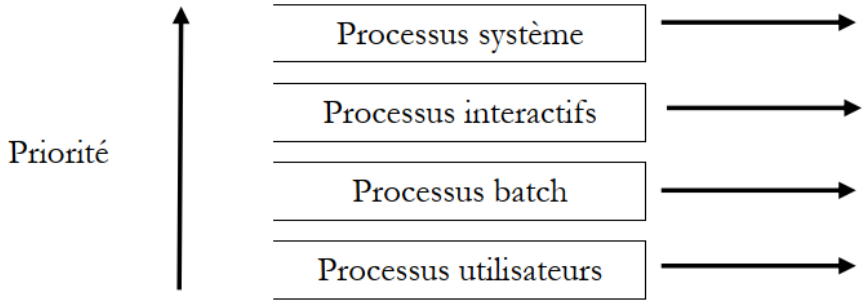
Il est crucial de choisir un quantum approprié pour éviter une surcharge du système due aux commutations fréquentes de contexte. Un quantum trop court augmentera les coûts liés aux commutations, tandis qu'un quantum trop long pourrait diminuer la réactivité du système. En conclusion, bien que l'algorithme Round Robin favorise un partage équitable du processeur, le choix du quantum doit être équilibré pour optimiser à la fois la réactivité et l'efficacité du système.

IV.5 Scheduling avec Files d'Attente Multiniveaux

Une autre classe d'algorithmes de scheduling est conçue pour des situations où les processus peuvent être facilement regroupés en différentes catégories. Par exemple, il peut

être utile de distinguer les processus de premier plan (interactifs) des processus d'arrière-plan (traitement par lot). Ces deux types de processus ont des besoins différents en termes de temps de réponse, ce qui peut nécessiter des stratégies de scheduling distinctes. Les processus de premier plan, par exemple, pourraient avoir une priorité plus élevée que ceux d'arrière-plan.

L'algorithme de scheduling avec files d'attente multiniveaux divise ainsi la file d'attente des processus prêts en plusieurs files d'attente distinctes. Chaque file peut être gérée selon une politique de scheduling propre, en fonction des caractéristiques des processus qu'elle contient. La figure suivante donné un exemple de découpage de ces files d'attente :

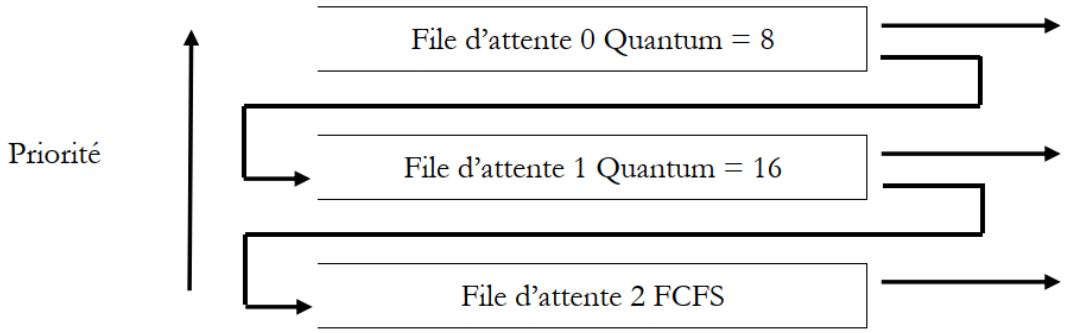


Les processus sont en permanence assignés à une file d'attente spécifique en fonction de certaines caractéristiques comme le type de processus, la taille de la mémoire ou la priorité. Chaque file d'attente est gérée par son propre algorithme de scheduling : par exemple, la file des processus systèmes peut être gérée par un algorithme de plus haute priorité, celle des processus interactifs par l'algorithme Round Robin, et celle des processus batch par l'algorithme FCFS. Il doit également y avoir un scheduling entre les différentes files d'attente. Par exemple, dans un ensemble de files d'attente multiniveaux, les processus systèmes ont la priorité absolue, suivis par les processus interactifs, les processus batch, et enfin les processus utilisateurs. Ainsi, aucun processus batch ne peut s'exécuter tant que les files d'attente des processus systèmes et interactifs ne sont pas vides. En cas d'arrivée d'un processus interactif, un processus batch en cours d'exécution sera interrompu. Une autre approche consiste à allouer des tranches de temps spécifiques à chaque file d'attente, par exemple, 80 % du temps processeur aux processus de premier plan et 20 % aux processus d'arrière-plan.

IV.6 Algorithme de scheduling avec files d'attente multiniveaux

Dans un algorithme de scheduling avec files d'attente multiniveaux, les processus sont habituellement assignés de manière permanente à une file d'attente dès leur entrée dans le système, sans possibilité de déplacement entre les files. Cette approche minimise la surcharge due au scheduling, mais manque de flexibilité. Le scheduling avec feedback multiniveaux introduit davantage de souplesse en permettant aux processus de se déplacer

entre les files d'attente en fonction de l'évolution de leurs caractéristiques. Par exemple, dans un système doté de trois files d'attente multiniveaux (File 0, File 1, et File 2), où la File 0 est la plus prioritaire, les Files 0 et 1 sont gérées selon la politique Round Robin, tandis que la File 2 utilise la technique FCFS. Un processus entrant est initialement placé dans la File 0 et reçoit une tranche de temps de 8 ms. S'il ne termine pas dans ce délai, il est déplacé vers la File 1. Si la File 0 est vide, le processus en tête de la File 1 reçoit une tranche de 16 ms. S'il n'achève toujours pas son exécution, il est déplacé vers la File 2, qui est exécutée uniquement lorsque les Files 0 et 1 sont vides.



V. Conclusion

En conclusion, l'exécution efficace des programmes au sein d'un système informatique repose sur une série de mécanismes bien orchestrés. Nous avons exploré les fondements de l'architecture des ordinateurs et le cheminement des programmes, en soulignant l'importance des processus et des interruptions dans la gestion des ressources du système. La compréhension des concepts de processus, y compris leur cycle de vie et les transitions d'état, est essentielle pour saisir la complexité de l'exécution simultanée de multiples tâches.

L'étude des interruptions a révélé leur rôle crucial dans la gestion des événements asynchrones, tandis que l'exploration des algorithmes d'ordonnancement a mis en lumière les différentes stratégies employées pour maximiser l'efficacité du processeur. Chaque algorithme, qu'il soit non préemptif comme le FCFS ou préemptif comme le Round Robin, présente des avantages et des inconvénients spécifiques qui doivent être évalués en fonction des besoins du système.

Le chapitre a également abordé les critères de performance qui permettent d'évaluer ces algorithmes, soulignant l'importance d'un ordonnancement adapté aux caractéristiques des processus en cours. En fin de compte, l'ordonnancement est un élément central dans la gestion des systèmes d'exploitation, influençant directement la réactivité, l'équité, et la performance globale du système.

L'intégration de ces connaissances permet non seulement de mieux comprendre les défis auxquels sont confrontés les concepteurs de systèmes, mais aussi de mieux apprécier les solutions développées pour optimiser l'utilisation des ressources tout en garantissant une performance fiable et équitable pour tous les processus en cours d'exécution.

VI. Exercices proposés

Questions de cours

1. Qu'est-ce que le cheminement d'un programme dans un système informatique ?
   - Expliquez les différentes étapes par lesquelles un programme passe, depuis sa soumission jusqu'à son exécution.

2. Quels sont les principaux états d'un processus et comment se transforment-ils ?
   - Décrivez les états d'un processus tels que "prêt", "en cours d'exécution", et "bloqué", ainsi que les transitions entre ces états.

3. Quels sont les mécanismes de commutation de contexte et pourquoi sont-ils importants ?
   - Décrivez comment et pourquoi le système d'exploitation effectue une commutation de contexte entre processus.

4. Comment les interruptions sont-elles traitées dans un système informatique ?
   - Expliquez les mécanismes de gestion des interruptions, y compris les types d'interruptions et leur priorité.

5. Quels sont les critères d'évaluation de performances utilisés pour comparer les algorithmes d'ordonnancement ?
   - Discutez des critères tels que le temps d'attente, le temps de réponse, le taux d'occupation du processeur, etc.

QCM

1. Quel est le rôle du descripteur de processus (PCB) dans un système d'exploitation?
   - A) Stocker les données utilisateur
   - B) Contenir les informations de gestion du processus
   - C) Gérer les périphériques externes

2. Quel est le but principal du mécanisme de commutation de contexte ?
   - A) Réduire la consommation d'énergie
   - B) Permettre le multitâche en sauvegardant et en restaurant l'état des processus
   - C) Optimiser la gestion des fichiers

3. Quel est le type de scheduling où un processus en cours peut être interrompu et remplacé par un autre processus ?

- A) Scheduling non préemptif
- B) Scheduling préemptif
- C) Scheduling en premier arrivé, premier servi

4. Quel algorithme d'ordonnancement est basé sur le principe de traiter les processus en fonction de leur durée d'exécution estimée ?

- A) FCFS (Premier Arrivé, Premier Servi)
- B) SJF (Plus Court d'abord)
- C) Round Robin (Tourniquet)

5. Quel est le critère de performance qui mesure le temps écoulé entre l'arrivée d'un processus et sa fin d'exécution ?

- A) Temps de réponse
- B) Temps d'attente
- C) Temps de séjour

6. Quel est l'objectif principal des algorithmes de scheduling avec files d'attente multiniveaux ?

- A) Simplifier la gestion des fichiers
- B) Améliorer la gestion de la mémoire
- C) Optimiser l'allocation des ressources processeur en utilisant plusieurs niveaux de priorité

7. Quel type d'interruption est généré par une erreur matérielle ?

- A) Interruption logicielle
- B) Interruption matérielle
- C) Interruption par événement

8. Quelle politique d'ordonnancement est souvent utilisée pour les systèmes interactifs afin de garantir que chaque processus reçoit une part équitable du temps processeur ?

- A) Scheduling avec Priorité
- B) Scheduling en Premier Arrivé, Premier Servi
- C) Round Robin (Tourniquet)

9. Quel est l'impact principal de la politique de scheduling préemptif sur les processus en cours d'exécution ?

- A) Les processus sont exécutés en séquence sans interruption

- B) Les processus peuvent être interrompus et remplacés par d'autres processus
- C) Les processus sont traités en priorité par leur durée d'exécution

10. Quel algorithme d'ordonnancement est le plus simple à mettre en œuvre mais peut entraîner une mauvaise utilisation du processeur dans des environnements multitâches ?
- A) FCFS (Premier Arrivé, Premier Servi)
- B) SJF (Plus Court d'abord)
- C) Scheduling avec Priorité

Exercices

Exercice 1 : Comprendre les principes fondamentaux de l'architecture des ordinateurs.
- Instructions : Décrivez les principaux composants d'un ordinateur et leur rôle dans l'exécution des programmes. Faites un schéma simple illustrant l'architecture de base d'un ordinateur, en incluant le processeur, la mémoire, et les périphériques d'entrée/sortie.
- Questions :
  - Quels sont les principaux rôles du processeur et de la mémoire dans l'exécution des programmes ?
  - Comment les périphériques d'entrée/sortie interagissent-ils avec le reste du système?

Exercice 2 :Analyser et comprendre les différents états d'un processus.
- Instructions : Créez un diagramme montrant les différents états d'un processus et les transitions possibles entre ces états. Décrivez une situation où un processus passe de l'état "prêt" à l'état "en cours d'exécution" et vice versa.
- Questions :
  - Quelle est la différence entre les états "prêt" et "bloqué" ?
  - Comment un processus passe-t-il de l'état "en cours d'exécution" à l'état "bloqué"?

Exercice 3 : Comprendre le mécanisme de commutation de contexte.
- Instructions : Expliquez en détail les étapes de la commutation de contexte d'un processus à un autre. Décrivez les informations sauvegardées et restaurées lors de ce processus.
- Questions :
  - Quelles informations doivent être sauvegardées lors de la commutation de contexte?
  - Pourquoi est-il important de sauvegarder l'état des registres et de la mémoire ?

Exercice 4 : Comparer différents algorithmes d'ordonnancement.

- Instructions : Choisissez deux algorithmes d'ordonnancement (par exemple, FCFS et Round Robin) et comparez-les en termes de temps d'attente, de temps de réponse et d'efficacité globale. Utilisez des exemples de processus pour illustrer vos points.
- Questions :
  - Quel algorithme d'ordonnancement est le plus adapté pour les systèmes interactifs et pourquoi ?
  - Quels sont les avantages et les inconvénients de chaque algorithme par rapport aux autres ?

Exercice 5 : Soient les processus suivants, dont les données sont précisées par le tableau suivant :

Processus	Temps arrivé	Temps d'exécution	Priorité
P1	0	10	3
P2	5	6	3
P3	5	7	2
P4	8	4	1
P5	12	10	4

1 : On applique l'algorithme de scheduling 1 : "premier arrivé, premier servi". Dessiner le digramme de Gantt
correspondant.
2 : On applique l'algorithme de scheduling 2 : "Round Robin" avec un quantum égal à 7. Dessiner le digramme
de Gantt correspondant.
3 : On applique l'algorithme de scheduling 3 : "Priorité (preemptif)". Dessinez le diagramme de Gantt correspondant.
4 : Donnez les temps d'attente et le temps de réponse de chaque processus pour chacun des 3 algorithmes précédents.
5 : calculer le temps moyen de réponse de chaque algorithme, quel est le meilleur algo.

Exercice 6 : Cinq travaux (processus) A, B, C, D et E arrivent pratiquement en même temps dans un centre de calcul. Leur temps d'exécution respectif est estimé à 10, 6, 2, 4 et 8 secondes.
Tracez le digramme de Gantt et déterminez le temps moyen de rotation (de reponse) pour chacun des algorithmes d'ordonnancement suivants. Ne tenez pas compte du temps perdu lors de la commutation des processus.

- Premier arrivé, premier servi FCFS ou FIFO (exécution dans l'ordre 10, 6, 2, 4, 8) ;
- Plus court d'abord SJF ;
- Tourniquet (quantum q = 4 s).

Quel est le meilleur algo.

Exercice 7 :

On considère 4 programmes P1, P2, P3 et P4 dont le comportement est définit comme suit :

Programme	Comportement
P1	Calcul pendant 40 ms Lecture disque pendant 50 ms Calcul pendant 30 ms Lecture disque pendant 40 ms Calcul pendant 20 ms
P2	Calcul pendant 30 ms Lecture disque pendant 80 ms Calcul pendant 80 ms Lecture disque pendant 20 ms Calcul pendant 10 ms
P3	Calcul pendant 40 ms Lecture disque pendant 40 ms Calcul pendant 10 ms
P4	Calcul pendant 80 ms

Les 4 programmes sont lancés en même temps dans un système de multiprogrammation.
On considère que l'ordonnancement sur le processeur se fait selon une stratégie FCFS (FIFO).

1. Remplir le diagramme d'exécution de ces 4 programmes selon le modèle ci-joint (Pour vous aider on a coché les 4 premières cases du programme P1).
2. Calculer les temps suivants : Temps d'exécution de chaque programme, Temps moyen d'exécution , Temps d'attente pour chaque programme , Temps moyen d'attente

P1

E/S																																								
Attente																																								
Prêt																																								
Actif	×	×	×	×																																				

Note : Chaque case de ces diagrammes correspond à 10 ms. Créer le même tableau pour les autres processeurs P2, P3 , P4

Exercice 8 : On considère 4 processus, A, B, C, D. On suppose que l'exécution des processus nécessite :

- Pour A : 7 unités de temps CPU, 3 unités de temps d'E/S et 5 unités de temps CPU.
- Pour B : 6 unités de temps CPU, 4 unités de temps d'E/S, 4 unités de temps CPU.
- Pour C : 5 unités de temps CPU.
- Pour D : 1 unité de temps CPU, 4 unités de temps d'E/S et 2 unités de temps CPU.

On suppose que : A se présente en premier, à l'instant 0. B se présente à l'instant 1, C se présente à l'instant 9, D se présente à l'instant 12.
Montrez comment les 4 processus vont utiliser le processeur dans chacun des cas suivants (diagramme de gantt) :

1) Chaque processus a son propre périphérique d'E/S et l'ordonnanceur fonctionne selon Premier Arrivée Premier Servi FCFS (sans préemption).

2) Chaque processus a son propre périphérique d'E/S et l'ordonnanceur utilise l'algorithme du tourniquet, avec un quantum de 5. Le temps de commutation est égal à 0. Donnez, dans ce cas, les temps de rotation des processus A, B, C et D.

3) Les trois processus utilisent le même périphérique d'E/S dont la file d'attente est gérée par l'algorithme FCFS. L'ordonnanceur du processeur utilise l'algorithme du tourniquet, avec un quantum de 5. Le temps de commutation est supposé égal à 0.

Exercice 9 :

On considère un système possédant deux processeurs et une seule file d'attente pour les processus prêts.

Soit le scénario d'arrivée des processus suivants : A, B, C, et D, ayant les caractéristiques suivantes (la priorité 1 correspond à la plus faible priorité). Le CPU1 a la priorité d'accès à la file des processus prêts par rapport au CPU2.

Processus	Priorité	Instant d'arrivée	Durée d'exécution
A	2	0	4
B	4	2	5
C	3	0	6
D	1	0	7

Q1) Complétez les diagrammes, pour chacun des algorithmes de scheduling suivants :

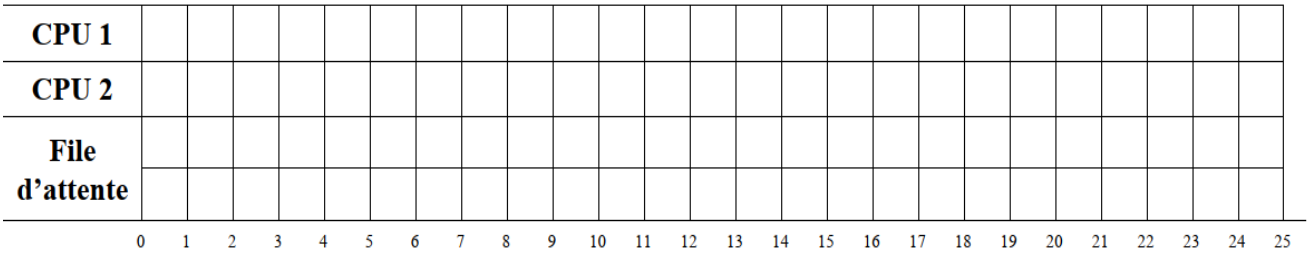
a) FCFS (First Come First Served),

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
CPU 1																										
CPU 2																										
File d'attente																										

b) Plus haute priorité préemptif (avec réquisition),

CPU 1																										
CPU 2																										
File d'attente																										

c) Round Robin (RR avec quantum = 2)



Q2) Donnez les temps d'attente (TA) et de réponse (TR) des processus.

VII. Correction des exercices

Questions de Cours

1. Qu'est-ce que le cheminement d'un programme dans un système informatique ?
   - Réponse :

Le cheminement d'un programme dans un système informatique comprend plusieurs étapes clés :

- Chargement du programme : Le programme est chargé en mémoire à partir du disque dur.
- Allocation des ressources : Le système d'exploitation alloue les ressources nécessaires (mémoire, CPU).
- Exécution : Le programme est exécuté par le processeur en passant par différents états (prêt, en cours d'exécution, bloqué).
- Gestion des entrées/sorties : Le système gère les appels d'E/S pour interagir avec les périphériques.
- Terminaison : Le programme termine son exécution, et les ressources sont libérées.

2. Quels sont les principaux états d'un processus et comment se transforment-ils ?
   - Réponse :
     - Prêt : Le processus est chargé en mémoire et prêt à être exécuté par le CPU.
     - En cours d'exécution : Le processus est en cours d'exécution par le processeur.
     - Bloqué : Le processus attend une ressource (par exemple, une E/S).
     - Transitions :
       - De Prêt à En cours d'exécution lorsque le CPU devient disponible.
       - De En cours d'exécution à Bloqué lorsque le processus attend une ressource.
       - De Bloqué à Prêt lorsque la ressource devient disponible.
3. Quels sont les mécanismes de commutation de contexte et pourquoi sont-ils importants ?
   - Réponse :

- La commutation de contexte est le processus par lequel le système d'exploitation enregistre l'état d'un processus en cours pour pouvoir en exécuter un autre.

- Importance : Elle permet le multitâche en interrompant un processus et en permettant à un autre d'utiliser le CPU, maximisant ainsi l'utilisation des ressources du système.

4. Comment les interruptions sont-elles traitées dans un système informatique ?
   - Réponse :
   - Les interruptions sont traitées par le système d'exploitation via un gestionnaire d'interruptions. Les types d'interruptions incluent les interruptions matérielles (générées par les périphériques), logicielles (générées par des instructions), et de minuterie (pour le scheduling).
   - Gestion : Lorsque l'interruption se produit, le CPU arrête son travail actuel, sauvegarde l'état, exécute le gestionnaire d'interruptions, puis reprend l'exécution précédente.

5. Quels sont les critères d'évaluation de performances utilisés pour comparer les algorithmes d'ordonnancement ?
   - Réponse :
   - Temps d'attente : Durée pendant laquelle un processus reste en file d'attente.
   - Temps de réponse : Temps écoulé entre la soumission d'un processus et le début de sa première exécution.
   - Taux d'occupation du processeur : Pourcentage du temps durant lequel le CPU est actif.
   - Temps de séjour (de rotation) : Temps total passé dans le système, de l'arrivée à la terminaison.

QCM

1. Quel est le rôle du descripteur de processus (PCB) dans un système d'exploitation?
   - Réponse : B) Contenir les informations de gestion du processus.

2. Quel est le but principal du mécanisme de commutation de contexte ?
   - Réponse : B) Permettre le multitâche en sauvegardant et en restaurant l'état des processus.

3. Quel est le type de scheduling où un processus en cours peut être interrompu et remplacé par un autre processus ?
   - Réponse : B) Scheduling préemptif.

4. Quel algorithme d'ordonnancement est basé sur le principe de traiter les processus en fonction de leur durée d'exécution estimée ?
   - Réponse : B) SJF (Plus Court d'abord).

5. Quel est le critère de performance qui mesure le temps écoulé entre l'arrivée d'un processus et sa fin d'exécution ?

- Réponse : C) Temps de séjour.

6. Quel est l'objectif principal des algorithmes de scheduling avec files d'attente multiniveaux ?

- Réponse : C) Optimiser l'allocation des ressources processeur en utilisant plusieurs niveaux de priorité.

7. Quel type d'interruption est généré par une erreur matérielle ?

- Réponse : B) Interruption matérielle.

8. Quelle politique d'ordonnancement est souvent utilisée pour les systèmes interactifs afin de garantir que chaque processus reçoit une part équitable du temps processeur ?

- Réponse : C) Round Robin (Tourniquet).

9. Quel est l'impact principal de la politique de scheduling préemptif sur les processus en cours d'exécution ?

- Réponse : B) Les processus peuvent être interrompus et remplacés par d'autres processus.

10. Quel algorithme d'ordonnancement est le plus simple à mettre en œuvre mais peut entraîner une mauvaise utilisation du processeur dans des environnements multitâches ?

- Réponse : A) FCFS (Premier Arrivé, Premier Servi).

Exercice

Exercice 1 : Comprendre les principes fondamentaux de l'architecture des ordinateurs

- Réponse :
  - Composants principaux :
    - Processeur (CPU) : Exécute les instructions des programmes, effectue les calculs et les opérations logiques.
    - Mémoire (RAM) : Stocke les données et instructions temporairement pour un accès rapide par le CPU.
    - Périphériques d'entrée/sortie (E/S) : Permettent l'interaction entre l'utilisateur et l'ordinateur (clavier, écran, etc.).
    - Schéma : Un schéma simple montrera le CPU connecté à la mémoire, avec des liens vers les périphériques E/S.
  - Questions :
    - Rôle du processeur et de la mémoire : Le processeur exécute les instructions, tandis que la mémoire stocke temporairement les données nécessaires.

- Interaction des périphériques E/S : Ils permettent l'envoi de données vers le CPU et reçoivent les sorties du système.

Exercice 2 : Analyser et comprendre les différents états d'un processus

- Réponse :
- Diagramme des états : Un diagramme montrera les transitions entre les états Prêt, En cours d'exécution, et Bloqué.
- Transitions :
- Prêt à En cours d'exécution : Le processus obtient le CPU.
- En cours d'exécution à Bloqué : Le processus attend une ressource (E/S).
- Bloqué à Prêt : La ressource est maintenant disponible.
- Questions :
- Différence Prêt/Bloqué : Un processus prêt attend le CPU, tandis qu'un processus bloqué attend une ressource.
- Transition Exécution → Bloqué : Lorsqu'un processus demande une ressource indisponible.

Exercice 3 : Comprendre le mécanisme de commutation de contexte

- Réponse :
- Étapes de la commutation :
- Sauvegarde : L'état du processus (registres, compteur de programme) est enregistré dans le PCB.
- Chargement : L'état du prochain processus à exécuter est restauré.
- Exécution : Le CPU reprend l'exécution du nouveau processus.
- Questions :
- Informations sauvegardées : Registres, état de la pile, compteur de programme, etc.
- Importance : Sauvegarder l'état est crucial pour reprendre l'exécution d'un processus exactement où il s'est arrêté.

Exercice 4 : Comparer différents algorithmes d'ordonnancement

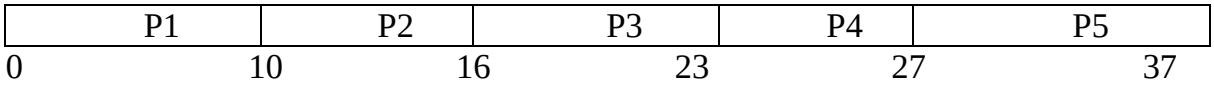
- Réponse :
- Comparaison FCFS vs. Round Robin :
- FCFS : Simplicité mais peut entraîner un long temps d'attente pour les processus courts.
- Round Robin : Meilleure équité, idéal pour les systèmes interactifs grâce à un temps de partage équitable du CPU.
- Exemple : Illustrer avec un Gantt chart pour montrer les différences de temps d'attente et de réponse.
- Questions :

- Algorithme pour systèmes interactifs : Round Robin, car il assure que chaque processus obtient du temps CPU régulièrement.

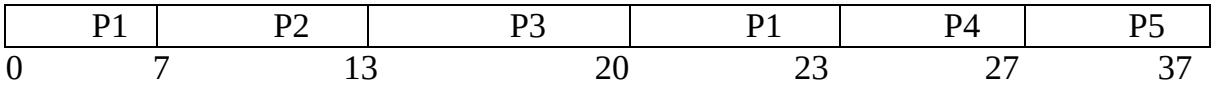
- Avantages/inconvénients : FCFS est simple mais pas optimal pour le multitâche; Round Robin est équitable mais nécessite des commutations fréquentes.

Exercice 5 :

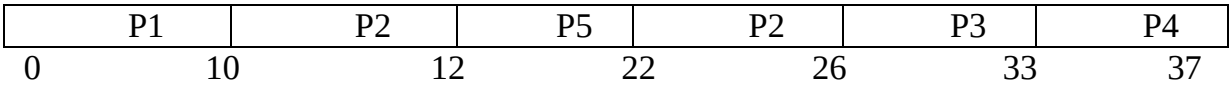
1 :



2 :



3 :



4 : temps d'attente

	1	2	3
P1	00	13	00
P2	05	02	17
P3	11	08	21
P4	15	15	25
P5	15	15	00

temps de réponse

	1	2	3
P1	10	23	10
P2	11	8	21
P3	18	15	28
P4	19	19	29
P5	25	25	10

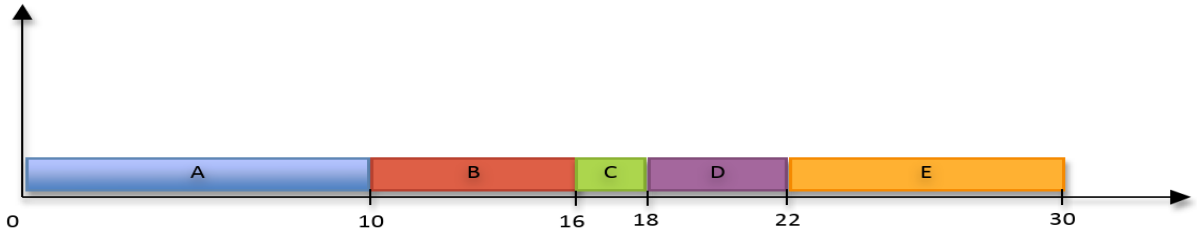
5 :

1	2	3
16.6	18	19.6

Le premier algo FIFO est le meilleur dans ce cas

Exercice 6 :

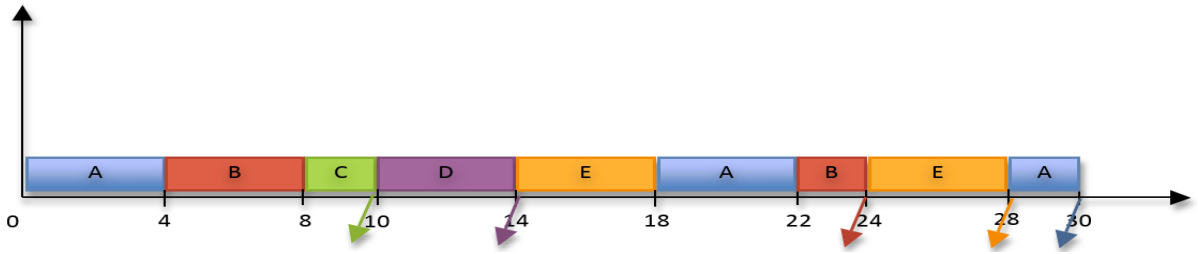
Algorithme FCFS → T_{RM} = ((10-0)+(16-0)+(18-0)+(22-0)+(30-0)) / 5 = **19,2**



Algorithme SJF → T_{RM} = ((2-0)+(6-0)+(12-0)+(20-0)+(30-0)) / 5 = **14**

Algorithme RR (quantum=4)

→ T_{RM} = ((10-0)+(14-0)+(24-0)+(28-0)+(30-0)) / 5 = **21,2**



Exercice 7 :

P1

E/S					■	■	■	■	■													■	■	■	■											
Attente																																			
Pret									■	■	■	■	■	■	■	■	■							■	■	■	■	■							
Actif	■	■	■	■														■	■	■									■	■					

P2

E/S									■	■	■	■	■	■	■	■													■	■						
Attente							■	■																												
Pret	■	■	■	■													■	■	■	■	■										■					
Actif					■	■	■															■	■	■	■	■	■	■	■				■				

P3

E/S																																			
Attente																																		
Pret																																		
Actif																																		

P4

E/S																																			
Attente																																		
Pret																																		
Actif																																		

Temps d'exécution :

P1 : 90 ms
P2 : 120 ms
P3 : 50 ms
P4 : 80 ms

⇨ Temps moyen d'exec : (90 + 120 + 50 + 80) / 4 = 85 ms

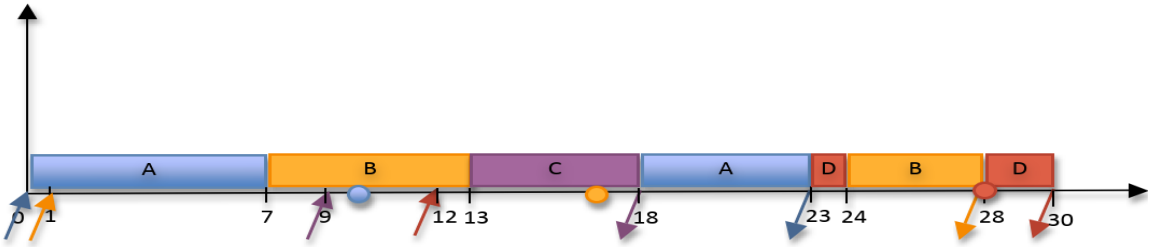
Temps d'attente

P1 : 330 – 90 = 240
P2 : 340 – 120 = 220
P3 : 310 – 50 = 260
P4 : 190 – 80 = 110

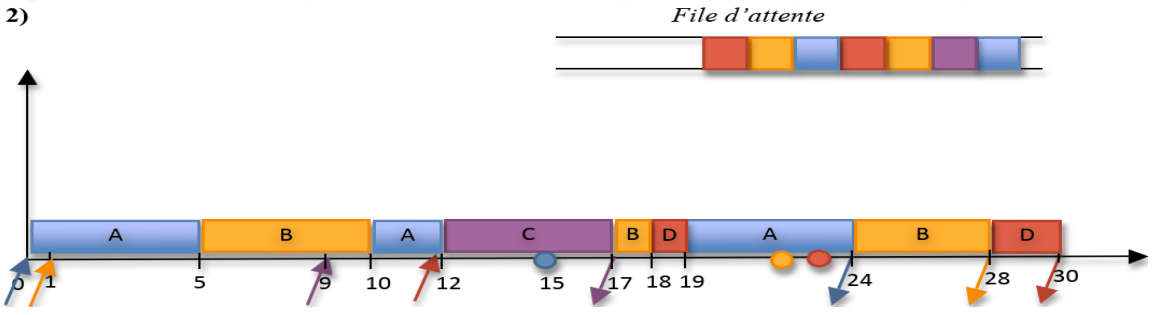
⇨ Temps moyen d'attente : (240 + 220 + 260 + 110) / 4 = 207.5 ms

Exercice 8 :

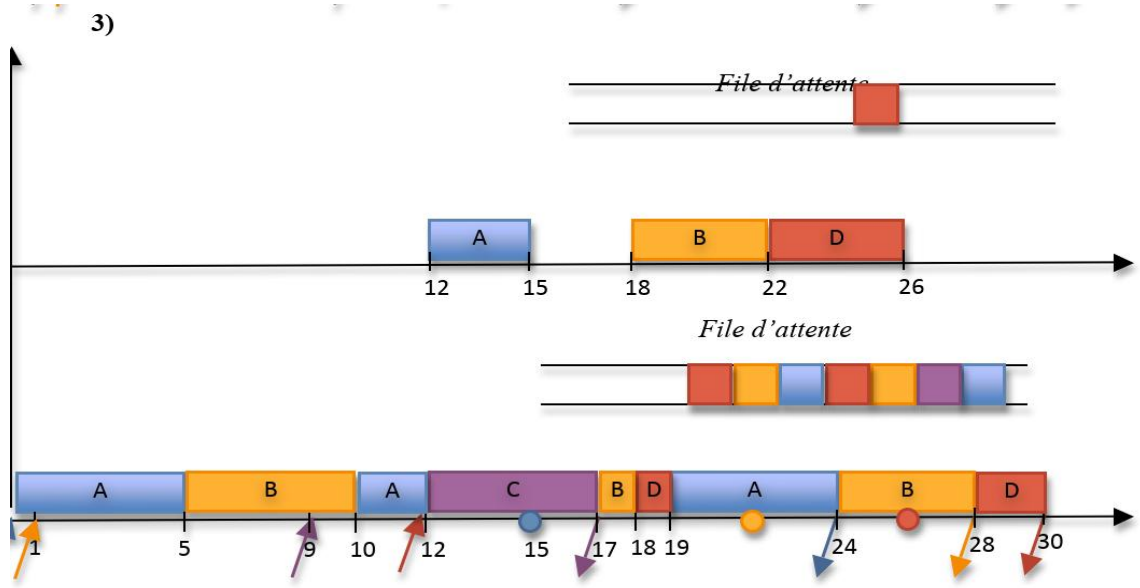
1)



2)



3)



Exercice 9 :

Q1) Complétez les diagrammes, pour chacun des algorithmes de scheduling suivants :

a) FCFS (First Come First Served),

CPU 1	A	A	A	A	D	D	D	D	D	D	D														
CPU 2	C	C	C	C	C	C	B	B	B	B	B														
File d'attente	D	D	D	D	B	B																			
			B	B																					
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

b) Plus haute priorité préemptif (avec réquisition),

CPU 1	C	C	C	C	C	C	A	A																	
CPU 2	A	A	B	B	B	B	B	D	D	D	D	D	D	D											
File d'attente	D	D	A	A	A	A	D																		
			D	D	D	D																			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

c) Round Robin (RR avec quantum = 2)

CPU 1	A	A	D	D	A	A	D	D	C	C	B														
CPU 2	C	C	B	B	C	C	B	B	D	D	D														
File d'attente	D	D	A	A	D	D	C	C	B	B															
			C	C	B	B																			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Q2) Donnez les temps d'attente (TA) et de réponse (TR) des processus.

	Algorithme FCFS		Algorithme Plus haute priorité		Algorithme RR avec Q = 2	
Processus	**Temps d'attente**	**Temps de Réponse**	**Temps d'attente**	**Temps de Réponse**	**Temps d'attente**	**Temps de Réponse**
A	0	4	4	8	2	6
B	4	9	0	5	4	9
C	0	6	0	6	4	10
D	4	11	7	14	4	11

CHAPITRE 3
Gestion de la Mémoire Centrale

I. Introduction

La mémoire centrale, également connue sous le nom de mémoire vive (RAM), est une ressource cruciale dans les systèmes d'exploitation, jouant un rôle déterminant dans l'exécution efficace des programmes. Elle permet aux processus de stocker temporairement les données et instructions nécessaires à leur fonctionnement. La gestion de cette mémoire est un défi fondamental pour les systèmes d'exploitation, car elle impacte directement la performance, la stabilité et l'efficacité globale du système.

Ce chapitre explore en détail les mécanismes de gestion de la mémoire centrale, en commençant par une vue d'ensemble des concepts clés tels que l'espace d'adressage et les objectifs du gestionnaire de mémoire. Nous étudierons les différentes techniques utilisées pour allouer la mémoire, notamment dans les systèmes monoprogrammés et multiprogrammés. Le swapping, les techniques de recouvrement, et les méthodes d'allocation de mémoire, qu'elles soient contiguës ou non contiguës, seront examinées en profondeur.

Nous aborderons également des concepts avancés tels que la pagination, la segmentation, et la mémoire virtuelle, qui permettent de surmonter les limitations physiques de la mémoire centrale et d'améliorer l'isolation et la sécurité des processus. Ces techniques jouent un rôle essentiel dans les systèmes modernes où la gestion optimisée de la mémoire est nécessaire pour répondre aux exigences de performances des applications contemporaines.

L'objectif de ce chapitre est de fournir une compréhension complète des différentes stratégies de gestion de la mémoire centrale et de leurs impacts sur les systèmes d'exploitation, en illustrant comment ces méthodes contribuent à l'efficacité et à la flexibilité des environnements informatiques modernes.

II. Gestion de la mémoire centrale

II.1 Généralités

Une mémoire dans un ordinateur est un système conçu pour acquérir, stocker et restituer des informations binaires. La capacité de la mémoire correspond à la quantité d'informations

qu'elle peut contenir, exprimée en bits ou en mots de 2^n bits. Le temps d'accès désigne l'intervalle entre la demande d'une information et le moment où elle devient disponible.

On distingue quatre types de mémoire en fonction de leur utilisation, capacité, temps d'accès, et coût :

- ✓ Les registres : Ce sont les plus petites mémoires, offrant un temps d'accès extrêmement court, mais elles sont également les plus coûteuses.
- ✓ La mémoire cache : Elle dispose d'une capacité limitée, mais elle est très rapide, avec un coût relativement élevé.
- ✓ La mémoire centrale : Sa capacité varie entre 128 Mo et 2 Go, avec un accès rapide et un coût plus abordable.
- ✓ Les mémoires auxiliaires : Elles offrent une capacité de stockage très importante, mais avec un temps d'accès plus long et sont les moins chères.

II.2 Définition de la mémoire centrale :

La mémoire centrale est constituée d'un ensemble d'emplacements ou de mots mémoire, chacun étant identifié par une adresse unique. Avec une adresse de n bits il est possible de référencer au plus 2^n cases mémoire. Elle prend en charge les opérations de lecture et d'écriture. La mémoire centrale est utilisée pour stocker les données et les programmes en cours d'exécution, servant d'intermédiaire entre le processeur et les autres composants de l'ordinateur. Le système qui gère et contrôle l'utilisation de la mémoire centrale est appelé le système de gestion de la mémoire centrale.

II.3 Objectifs du gestionnaire de mémoire :

Les principaux objectifs du gestionnaire de mémoire centrale sont les suivants :

- ✓ Partager et contrôler l'espace mémoire entre les différents processus.
- ✓ Protéger l'espace alloué à chaque processus.
- ✓ Réallouer et réorganiser la mémoire afin d'optimiser son utilisation.
- ✓ Organiser logiquement l'espace mémoire en partitions, pages ou segments.

Pour atteindre ces objectifs, le gestionnaire de mémoire centrale :

- ✓ Alloue l'espace mémoire aux processus.
- ✓ Libère l'espace mémoire occupé.
- ✓ Maintient un suivi des zones de mémoire utilisées et disponibles.

II.4 L'espace d'adressage

L'espace d'adressage logique et l'espace d'adressage physique sont deux concepts clés en informatique. Une adresse générée par la CPU est appelée « adresse logique » ou « adresse virtuelle », représentant la taille du processus et pouvant être modifiée. En revanche, une adresse vue par l'unité mémoire, appelée « adresse physique » ou « adresse réelle », correspond aux adresses effectivement utilisées par la mémoire. L'ensemble de toutes les adresses physiques liées aux adresses logiques constitue l'espace d'adressage physique. Le mappage des adresses virtuelles aux adresses physiques est effectué par l'unité de gestion de la mémoire (MMU) du matériel, et l'adresse physique reste constante. Pendant l'exécution, il est nécessaire de convertir les adresses logiques en adresses physiques. Par exemple, dans un système donné, une adresse physique peut être obtenue en ajoutant l'adresse de base contenue dans un registre à chaque adresse logique.

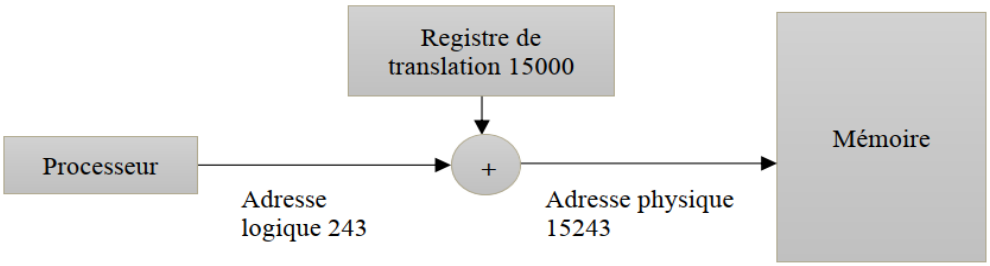


Figure 20 Conversion d'adresses logiques en adresses physiques par translation

III. Gestion de la Mémoire centrale dans les systèmes monoprogrammés

La gestion de la mémoire centrale dans les systèmes monoprogrammés se caractérise par la présence d'un seul processus utilisateur prêt à s'exécuter en mémoire. La mémoire centrale est divisée en deux zones distinctes :

- ✓ La première est réservée au système d'exploitation ou noyau (kernel), qui inclut ses différents modules et pilotes, et reste résidente pendant toute la durée d'activité de la machine;
- ✓ La seconde zone est entièrement dédiée à l'utilisateur actif, lui offrant une disponibilité totale.

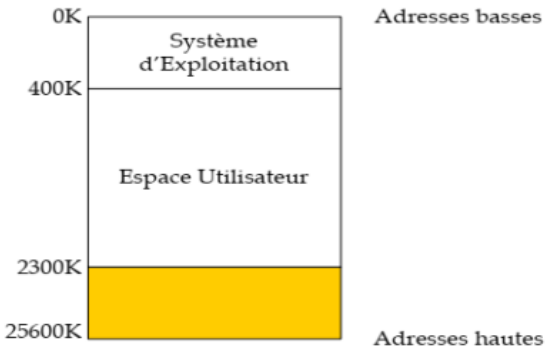


Figure 21 Organisation de la mémoire dans un système mono-programmé

Dans les systèmes monoprogrammés, un processus occupe l'espace mémoire jusqu'à sa terminaison, ce qui peut entraîner une utilisation inefficace des ressources. Il devient donc essentiel de trouver des méthodes pour gérer plus efficacement la mémoire, afin de permettre à plusieurs processus de partager les ressources limitées de manière optimale. Deux techniques couramment utilisées pour résoudre ces problèmes sont le swapping et le recouvrement.

III.1 Swapping (ou méthode de va-et-vient)

Dans un système monoprogrammé, un processus qui occupe de la mémoire la conserve jusqu'à sa fin, ce qui peut empêcher d'autres processus d'accéder à la mémoire centrale. Pour contourner cette limitation, la méthode du swapping permet de libérer temporairement l'espace mémoire occupé par un processus en le transférant sur le disque. Cela donne la possibilité à un autre processus en attente de charger en mémoire centrale et de s'exécuter. Par exemple, si le processus P1 se bloque en attendant une opération d'entrée/sortie (E/S), il peut être "swappé" sur le disque, et un autre processus P2 peut être chargé en mémoire pour continuer l'exécution.

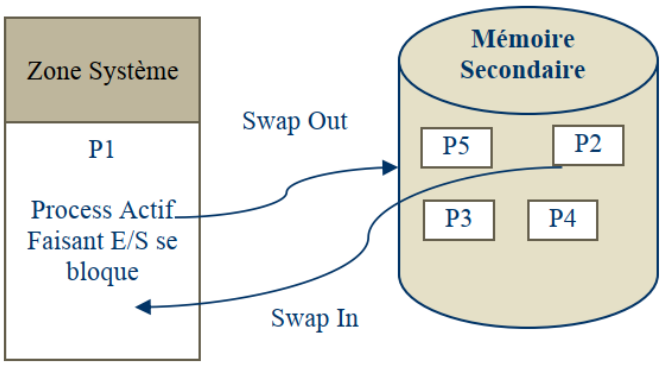


Figure 22 Technique de Swapping

III.2 Technique de recouvrement

Lorsque la taille d'un programme dépasse la capacité de la mémoire centrale, il devient nécessaire de gérer cette limitation de manière dynamique. Dans ce cas, la technique de recouvrement consiste à diviser le programme en plusieurs modules lors de sa conception. À l'exécution, seuls les modules effectivement nécessaires sont chargés en mémoire centrale, tandis que les modules non utilisés sont déchargés pour libérer de l'espace. Chaque nouveau module chargé remplace un module déjà présent en mémoire, permettant ainsi une utilisation optimale de la mémoire malgré ses limitations physiques.

IV. Gestion de la Mémoire centrale dans les systèmes multiprogrammés

IV.1 Introduction

La multiprogrammation permet l'exécution simultanée de plusieurs processus, optimisant ainsi l'utilisation du processeur en réduisant les temps d'attente liés aux opérations d'entrée/sortie. Cette technique repose sur la présence de plusieurs processus en mémoire, qui est alors partagée entre le système d'exploitation et ces processus. Un défi majeur se pose : comment organiser la mémoire pour faire cohabiter efficacement plusieurs processus tout en garantissant leur protection ? Deux approches principales se distinguent pour répondre à cette question.

- ✓ **Approche d'allocation contiguë** : Cette méthode consiste à allouer une section continue de mémoire à un processus ou un fichier. Autrement dit, la mémoire disponible pour un processus est regroupée en un seul bloc, ce qui évite la fragmentation de l'espace mémoire. La mémoire principale est divisée en deux parties : une pour le système d'exploitation et l'autre pour les programmes utilisateurs. Un programme, dans ce contexte, est vu comme un ensemble inséparable de mots mémoire contigus. Deux techniques d'allocation se rattachent à cette approche :
    - Allocation par partitions fixes : La mémoire est divisée en partitions de taille fixe, et chaque processus est chargé dans une partition de taille correspondante.
    - Allocation par partitions variables : Les partitions sont de taille variable, et chaque processus reçoit un bloc de mémoire correspondant à ses besoins spécifiques.
- ✓ **Approche d'allocation non contiguë** : Contrairement à l'approche contiguë, cette méthode alloue la mémoire à un processus en différents emplacements selon ses besoins, ce qui entraîne une répartition dispersée de l'espace mémoire disponible. Un programme est ici représenté par un ensemble de mots mémoire non contigus et sécables, permettant une plus grande flexibilité. Dans cette méthode, un programme peut être divisé en plusieurs segments ou pages, chacun contenant un ensemble de mots contigus, qui peuvent être traités indépendamment les uns des autres. Les principales techniques associées à cette approche sont :
    - Pagination : La mémoire est divisée en pages de taille fixe, et un processus est réparti sur ces pages non contiguës.
    - Segmentation : La mémoire est divisée en segments de taille variable, en fonction de la structure logique du programme.
    - Segmentation paginée : Combine les deux techniques précédentes en segmentant d'abord la mémoire, puis en paginant chaque segment.

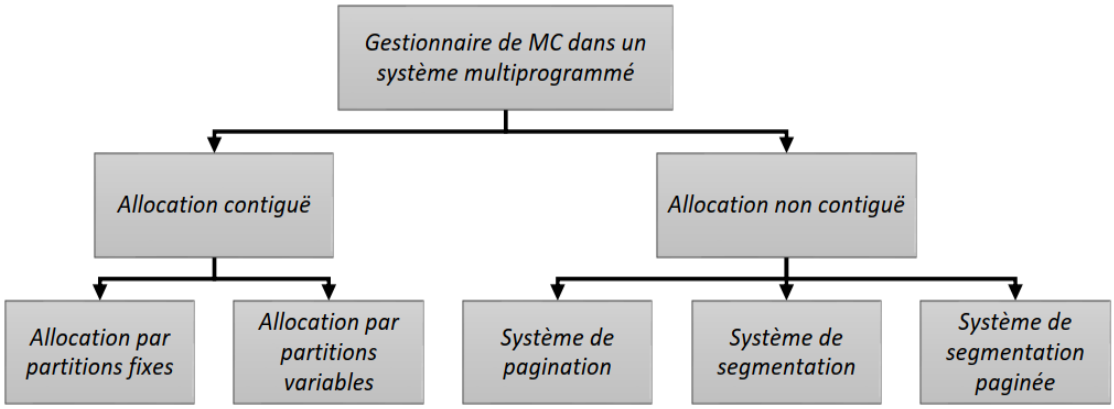


Figure 23 Les approches de gestion de mémoire centrale dans un système multiprogrammé

IV.2 Allocation contiguë

L'allocation contiguë (Contiguous Allocation) est une stratégie simple et efficace pour implémenter la multiprogrammation. Elle consiste à diviser la mémoire principale en régions distinctes, appelées partitions mémoires, où chaque partition possède son propre espace d'adressage. Le partitionnement de la mémoire peut être réalisé de manière statique, avec des partitions de taille fixe, ou de manière dynamique, avec des partitions de taille variable.

Dans cette approche, chaque processus est entièrement chargé en mémoire, et l'exécutable utilise des adresses relatives. Les adresses réelles sont ensuite déterminées au moment du chargement du processus en mémoire, permettant ainsi une organisation directe et cohérente de l'espace mémoire.

IV.2.1 Allocation contiguë par partitions fixes

L'allocation contiguë par partitions fixes est une méthode qui divise la mémoire en partitions de **taille fixe** lors de l'initialisation du système. Ces partitions, dont les tailles et le nombre sont déterminés à l'avance (soit par le système, soit par le constructeur), ne sont pas nécessairement égales. Un des principaux défis de cette approche réside dans la détermination du nombre optimal de partitions et de leurs tailles respectives.

Le système d'exploitation gère ces partitions à l'aide d'une table de description des partitions (PDT, Partition Description Table), qui indique quelles portions de la mémoire sont disponibles et lesquelles sont occupées. Les programmes qui ne trouvent pas immédiatement de place en mémoire sont placés dans une file d'attente. Cette file peut être unique pour toutes les partitions, ou distincte pour chaque partition.

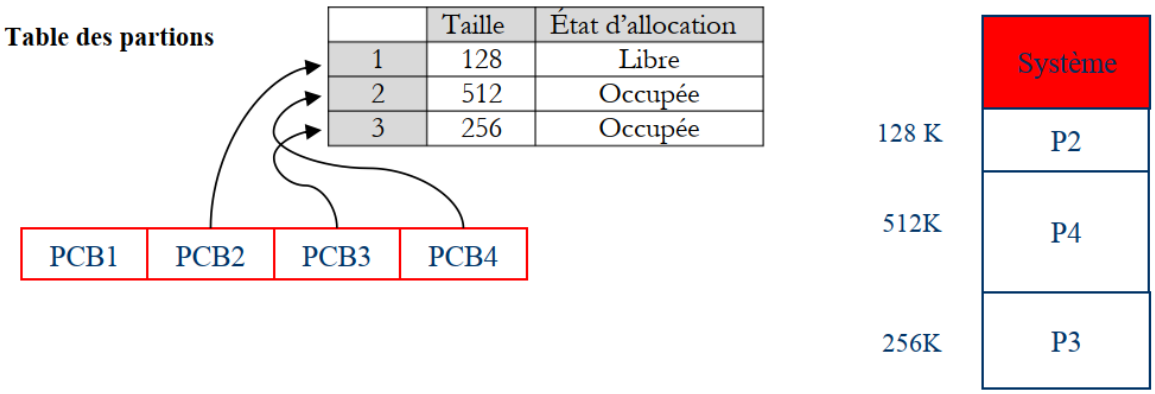


Figure 24 Allocation contiguë à partitions multiples de tailles fixes

Dans le cas de files d'attente par partition, chaque nouveau processus est placé dans la file d'attente de la plus petite partition capable de le contenir. Cependant, cette approche peut entraîner des inefficacités, où un processus attend dans une file alors qu'une autre partition, compatible et libre, pourrait l'accueillir.

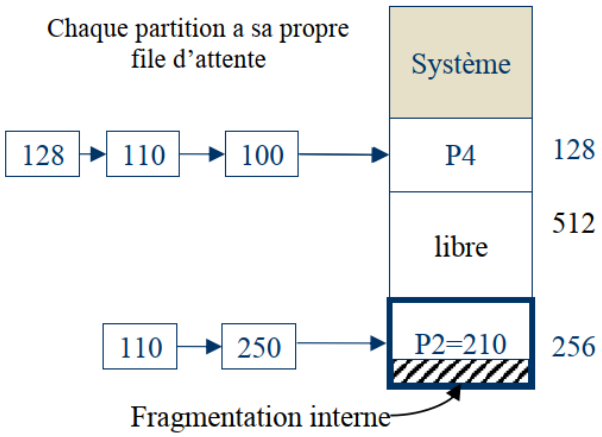


Figure 25 Partitions fixes avec files multiples

Une alternative consiste à utiliser une seule file d'attente globale, où le premier processus qui peut tenir dans une partition libre est chargé dès qu'une partition se libère. Cette méthode présente plusieurs stratégies :

- ✓ Première adéquation : La première partition disponible est attribuée au premier processus capable de l'occuper. Toutefois, cela peut gaspiller de la mémoire en allouant une grande partition à un petit processus.
- ✓ Plus grande adéquation : La partition est attribuée au plus grand processus possible dans la file d'attente, mais cela peut pénaliser les petits processus qui attendent plus longtemps.

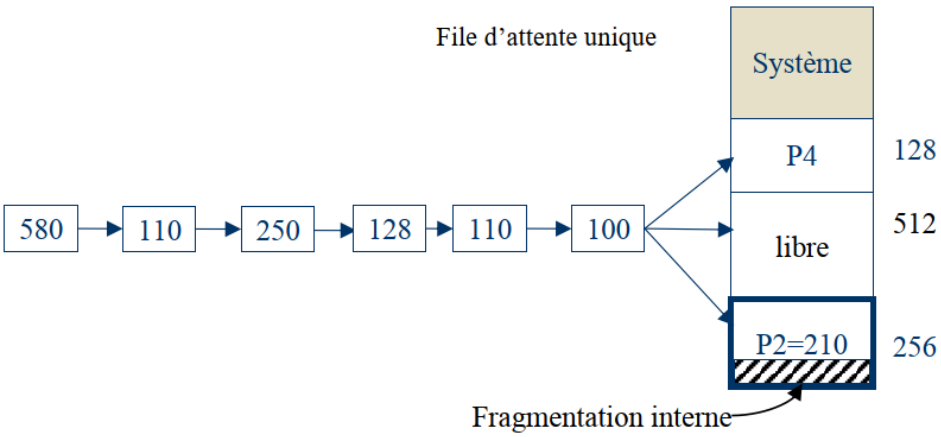


Figure 26 Partitions fixes avec une seule file

Les inconvénients de l'allocation par partitions fixes incluent :

- ✓ Fragmentation interne : Lorsque le processus n'occupe pas toute la partition, l'espace inutilisé reste gaspillé.
- ✓ Gestion complexe des files d'attente : Une mauvaise gestion peut entraîner une surcharge et un gaspillage de la mémoire.
- ✓ Limitation du degré de multiprogrammation : Le nombre de partitions fixe impose une limite au nombre de processus pouvant être exécutés simultanément.

IV.2.2 Allocation contiguë par partitions variables

IV.2.2.1 Principe d'allocation contiguë par partitions variables

Le gaspillage de mémoire et la fragmentation associés aux systèmes à partitions fixes ont conduit à l'évolution vers des partitions variables. Avec cette approche, la mémoire est découpée dynamiquement en partitions dont la taille est déterminée en fonction des besoins spécifiques des processus au moment de leur chargement. Ainsi, chaque programme se voit attribuer une partition dont la taille correspond exactement à celle requise.

Lorsque un programme termine son exécution, sa partition est récupérée par le système et peut être réattribuée à un autre programme, entièrement ou en partie, selon la demande. Cette méthode élimine les problèmes de partitions trop grandes ou trop petites, typiques des systèmes à partitions fixes, et améliore l'utilisation de la mémoire centrale (MC). Cependant, cette flexibilité accrue requiert un mécanisme de gestion plus complexe pour l'allocation et la libération des partitions. Il est nécessaire de maintenir une liste des espaces mémoire disponibles, afin de gérer efficacement les partitions variables et minimiser les problèmes de fragmentation interne.

Fragmentation Externe : Ce phénomène se produit lorsque la mémoire disponible est suffisamment grande pour accueillir un processus, mais est répartie en plusieurs petites partitions libres non contiguës. Autrement dit, même si la somme totale des espaces libres

pourrait satisfaire les besoins d'un processus, ce dernier ne peut pas être chargé car il ne trouve pas un espace mémoire contigu de taille adéquate. Cette fragmentation externe rend difficile la gestion efficace de la mémoire et peut conduire à un sous-optimisation de l'espace disponible.

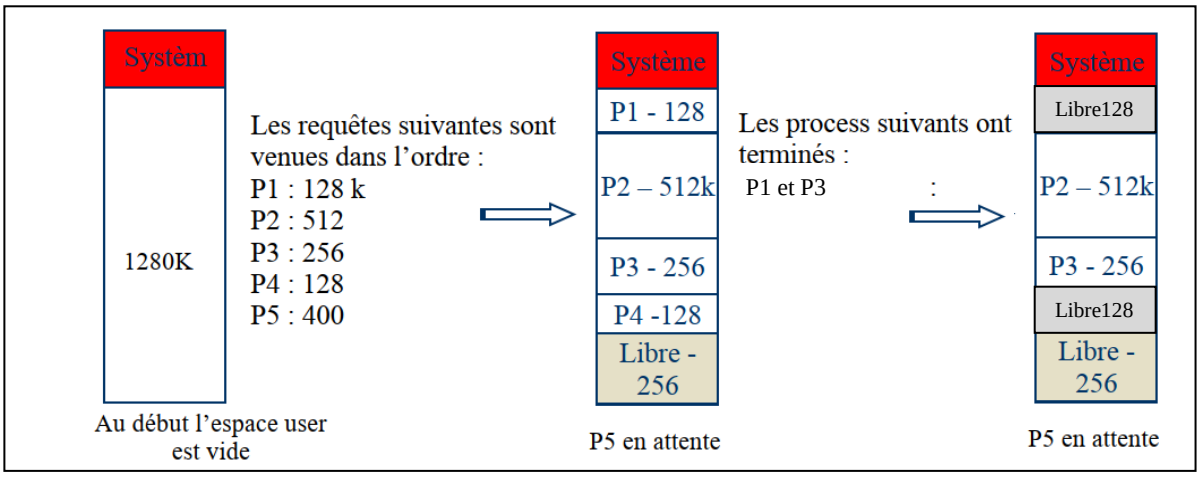


Figure 27 Exemple d'allocation des partitions variables et fragmentation externe(zones en gris)

IV.2.2.2 Stratégies de placement et d'allocation

Dans le contexte de l'allocation à partitions variables, où l'espace mémoire est divisé en différentes partitions de tailles variées, il est crucial de choisir la bonne stratégie pour optimiser l'utilisation de la mémoire. Ces stratégies visent à minimiser les situations où un processus ne peut pas être servi malgré la disponibilité suffisante de mémoire. Voici les trois principales stratégies d'allocation :

- ✓ **First Fit** : Cette stratégie consiste à parcourir la liste des partitions libres et à sélectionner la première zone dont la taille est suffisante pour satisfaire la demande du processus. Par exemple, si un processus nécessite 130 unités de mémoire et la première partition disponible a une taille de 140 unités, cette partition sera allouée au processus, laissant un résidu de 10 unités. Cette approche permet une recherche rapide et une allocation rapide, mais elle peut parfois allouer une partition qui aurait pu mieux servir un autre processus. Pour optimiser cette stratégie, il est souvent utile de trier la liste des partitions par adresse croissante.

- ✓ **Best Fit** : Avec cette méthode, on cherche la partition dont la taille est la plus proche de la demande du processus. Cela nécessite de parcourir l'ensemble de la liste des partitions libres pour trouver la meilleure adéquation. Par exemple, si un processus a besoin de 100 unités et il y a une partition de 105 unités disponibles, cette partition sera allouée, laissant un résidu de 5 unités. Bien que cette méthode assure une allocation précise, elle peut entraîner une fragmentation externe plus fine,

avec des petits espaces de mémoire souvent inutilisables. La recherche est plus lente, mais le tri des partitions par taille croissante peut aider à accélérer le processus d'allocation, bien que cela ralentisse l'insertion.

- ✓ **Worst Fit** : Cette stratégie consiste à choisir la partition la plus grande parmi toutes les partitions disponibles pour satisfaire la demande du processus. Par exemple, si une partition de 220 unités est disponible et qu'un processus demande 100 unités, cette partition sera allouée, laissant un résidu de 120 unités. Cette approche vise à minimiser la fragmentation externe en utilisant les plus grands espaces libres, qui peuvent être plus facilement réutilisés pour d'autres processus. Cependant, cette méthode est généralement plus lente car elle nécessite un parcours complet de la liste des partitions. Pour améliorer l'efficacité, il peut être avantageux de trier les partitions par taille décroissante, bien que cela ralentisse le processus de libération des partitions.

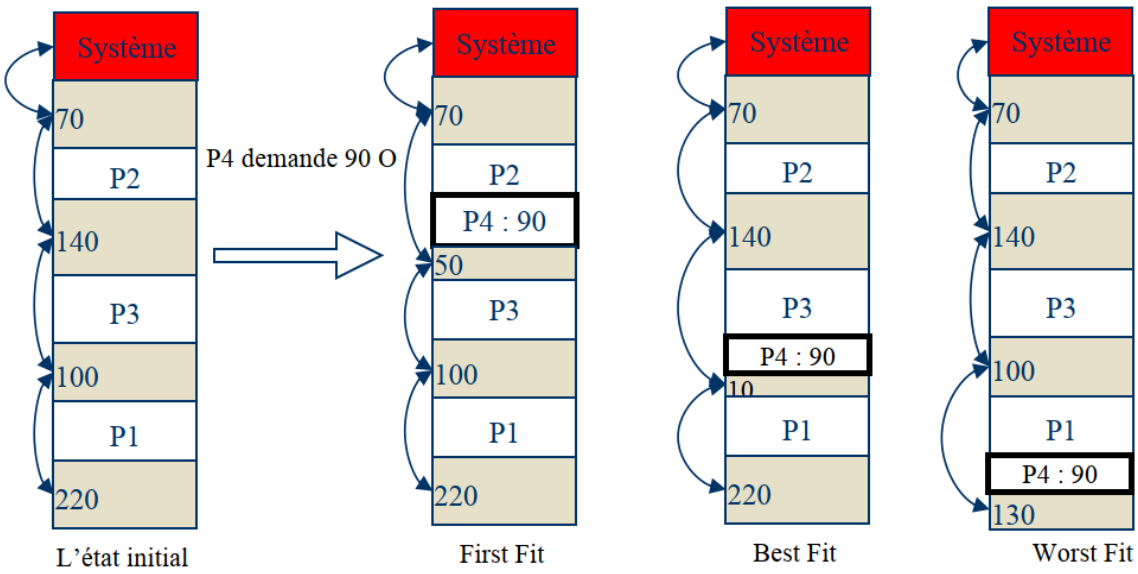


Figure 28 Exemple illustrant les algorithmes de placement

Chacune de ces stratégies a ses avantages et inconvénients, et le choix entre elles dépend des objectifs spécifiques de gestion de la mémoire et des caractéristiques du système.

Les stratégies d'allocation de mémoire précédemment discutées, telles que First Fit , Best Fit , et Worst Fit , cherchent à optimiser l'utilisation de la mémoire dans les systèmes à partitions variables. Toutefois, d'autres méthodes de gestion de la mémoire, comme le système Buddy, les tables de bits, et les listes linéaires chaînées, offrent des approches complémentaires pour résoudre les problèmes de fragmentation et d'allocation dynamique.

Le Buddy System est une méthode qui divise la mémoire en blocs de tailles puissances de deux, permettant une gestion efficace et rapide des allocations et libérations de mémoire. Lorsqu'une partition est demandée, le système trouve le bloc le plus proche en taille et le divise en deux si nécessaire, créant ainsi des "paires de copains" (buddies) qui facilitent la fusion des blocs libérés. Cette approche réduit la fragmentation externe en regroupant les blocs adjacents lorsqu'ils deviennent disponibles.

La gestion de l'espace par table de bits utilise un tableau de bits (bitmap) où chaque bit représente un bloc de mémoire. Un bit à 0 indique un bloc libre, tandis qu'un bit à 1 indique un bloc occupé. Cette méthode permet une gestion rapide et simple de l'espace mémoire, mais peut être moins efficace pour les systèmes avec de très grandes tailles de mémoire, où le tableau de bits pourrait devenir volumineux et consommer beaucoup d'espace.

La gestion de l'espace par liste linéaire chaînée implique l'utilisation d'une liste de blocs de mémoire chaînés, où chaque bloc contient des informations sur la taille et l'état de la partition, ainsi que des pointeurs vers les blocs adjacents. Cette méthode facilite l'insertion et la suppression de blocs libres, mais peut entraîner une recherche plus lente et une surcharge supplémentaire due à la gestion des pointeurs. Les listes linéaires chaînées permettent une gestion flexible de la mémoire, bien qu'elles puissent introduire une complexité supplémentaire dans la gestion des allocations et des libérations.

IV.2.2.3 Compactage

Les partitions multiples dans la gestion de la mémoire peuvent conduire à une fragmentation, un phénomène où la mémoire disponible est insuffisamment organisée pour répondre efficacement aux besoins des processus. Cette fragmentation se divise en deux types principaux :

- ✓ Fragmentation interne : Ce type de fragmentation survient lorsque la mémoire allouée à un processus est légèrement supérieure à celle réellement nécessaire. La différence entre la mémoire allouée et celle utilisée constitue la fragmentation interne. Elle représente une portion de mémoire à l'intérieur d'une partition qui reste inutilisée, bien qu'elle soit réservée. Cette inefficacité résulte du fait que la mémoire est divisée en partitions de taille fixe, ce qui ne correspond pas toujours exactement aux besoins des processus.

- ✓ Fragmentation externe : La fragmentation externe apparaît lorsque l'espace total de mémoire disponible est suffisant pour satisfaire une requête, mais cet espace est fragmenté en petits blocs non contigus. Bien que la somme de ces blocs libres soit suffisante pour charger un processus, leur dispersion empêche l'utilisation effective de la mémoire, car le processus ne peut être chargé dans un seul bloc continu. Cela

réduit l'efficacité de la gestion de la mémoire et peut mener à une situation où, malgré une quantité suffisante de mémoire libre, aucun processus ne peut être chargé.

Pour remédier à la fragmentation externe, une technique appelée compactage est utilisée. Le compactage consiste à déplacer les processus en mémoire de manière à regrouper tous les blocs libres en une seule grande zone contiguë, souvent vers le début ou le milieu de la mémoire. Cette opération, bien qu'efficace pour éliminer la fragmentation externe et améliorer les performances en permettant l'allocation de mémoire contiguë pour de nouveaux processus, est coûteuse en termes de temps de calcul du CPU. Le compactage est généralement déclenché lorsqu'un processus demande une partition suffisamment grande, mais que l'espace mémoire disponible est fragmenté, empêchant ainsi son allocation. Cette technique, parfois appelée "ramasse-miettes" ou "garbage collection", est essentielle pour maintenir une utilisation efficace de la mémoire centrale dans les systèmes multiprogrammés.

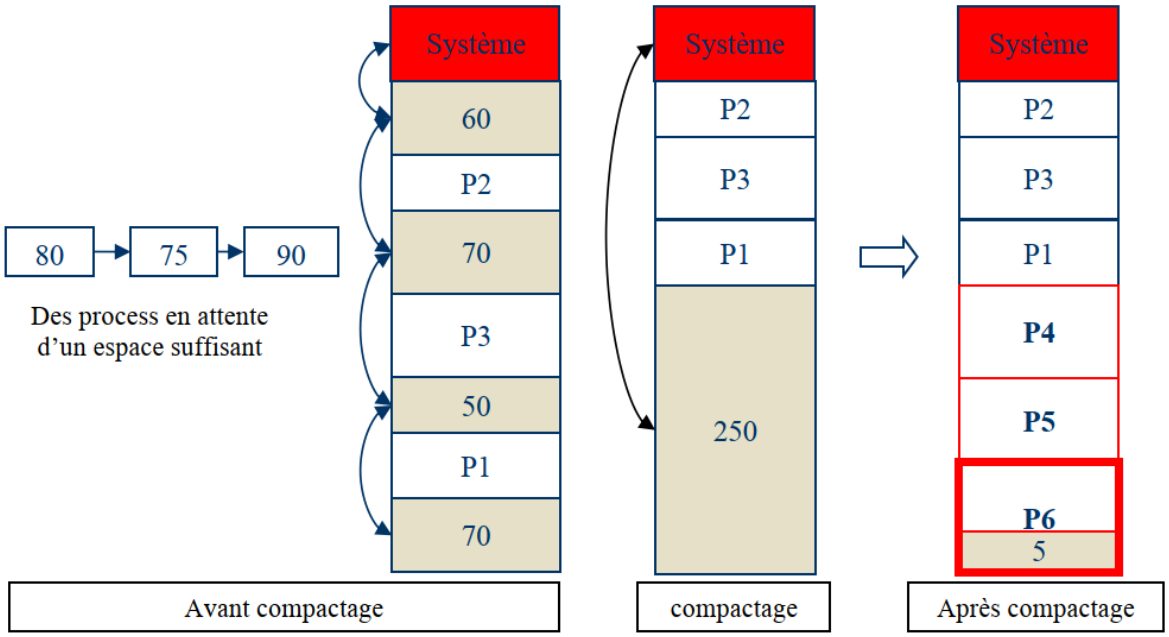


Figure 29 Compactage

IV.2.3 Critiques

Les techniques d'allocation contiguë présentent plusieurs limitations notables :

- ✓ Lors de l'allocation, le gestionnaire de mémoire doit trouver un espace contigu suffisamment grand pour héberger un programme. Même si l'ensemble des fragments de mémoire libre suffirait en théorie à loger ce programme, l'allocation contiguë échoue si ces fragments ne sont pas adjacents. À mesure que la taille d'un

programme augmente, les chances de trouver un espace contigu adéquat diminuent, ce qui limite l'efficacité de l'allocation.

- ✓ La fragmentation externe est un problème courant dans les systèmes utilisant l'allocation contiguë, survenant fréquemment à cause des multiples allocations et désallocations de mémoire. Bien que le compactage soit une solution traditionnelle pour regrouper les fragments de mémoire libre en un seul bloc contigu, cette méthode devient de plus en plus inefficace et coûteuse en termes de temps de traitement.

- ✓ Les programmes dont la taille dépasse celle de la mémoire centrale ne peuvent tout simplement pas être exécutés, car l'allocation contiguë exige que l'intégralité du programme soit chargée en mémoire d'un seul tenant.

Pour surmonter ces limitations, le gestionnaire de mémoire doit revoir sa manière de traiter les programmes. Au lieu de considérer un programme comme une entité insécable, il devient plus efficace de le diviser en plusieurs segments ou blocs insécables, chacun pouvant être chargé dans une zone de mémoire libre, même si ces zones sont dispersées. Cette approche permet d'exploiter les fragments de mémoire disponibles et d'optimiser l'utilisation de la mémoire centrale, en rendant possible l'exécution de programmes plus volumineux sans nécessiter un espace contigu unique.

IV.3 Allocation non contiguë

IV.3.1 Introduction

Dans la gestion de la mémoire d'un système informatique, l'allocation de l'espace mémoire est une tâche cruciale qui influence directement la performance des programmes et l'efficacité du système. L'une des premières méthodes développées pour cette tâche est l'allocation contiguë, où chaque processus est chargé en mémoire comme une entité unique, occupant une zone de mémoire continue. Bien que simple et facile à implémenter, cette approche présente des défis considérables, notamment en ce qui concerne la fragmentation de la mémoire et l'incapacité à maximiser l'utilisation des espaces libres.

L'allocation contiguë fonctionne efficacement dans des environnements où les processus sont de petite taille et où la demande de mémoire reste modérée. Cependant, à mesure que les programmes deviennent plus volumineux et que la complexité des systèmes augmente, les limitations de cette méthode deviennent plus apparentes. La fragmentation externe, où la mémoire libre est dispersée en petits fragments non contigus, est l'un des problèmes majeurs, rendant difficile l'allocation efficace des ressources pour de grands programmes. De plus, le besoin de disposer d'un espace mémoire contigu pour chaque processus limite la flexibilité et la capacité d'exécution des systèmes plus modernes.

Pour remédier à ces défis, des techniques d'allocation non contiguë ont été introduites. Ces méthodes permettent de diviser les programmes en morceaux plus petits qui peuvent être alloués séparément dans différents trous de mémoire, améliorant ainsi l'utilisation des espaces libres. Parmi ces techniques, la pagination et la segmentation se distinguent comme des solutions efficaces pour surmonter les limitations de l'allocation contiguë, offrant des approches plus flexibles pour la gestion de la mémoire tout en minimisant les effets de la fragmentation. Ces stratégies permettent non seulement de mieux exploiter la mémoire disponible, mais aussi de faciliter la gestion des processus dans des systèmes complexes et dynamiques.

IV.3.2 Pagination de la Mémoire Centrale

IV.3.2.1 Principe de la pagination

La pagination est une technique essentielle de gestion de la mémoire, où l'espace d'adressage logique d'un programme est divisé en morceaux de taille fixe appelés pages (c'eat à dire le programme est devisé en pages). Cet espace logique correspond à l'ensemble des adresses que le programme peut utiliser. De manière parallèle, la mémoire physique, ou mémoire centrale (MC), est également découpée en unités de taille fixe appelées cadres de page (ou frames). La taille d'une page est identique à celle d'un cadre de page, et cette taille, définie par le matériel et le système d'exploitation, est généralement une puissance de 2, variant typiquement entre 512 octets et 8192 octets. Ce choix facilite grandement la conversion d'une adresse logique en un numéro de page et un décalage à l'intérieur de cette page.

Lorsque le système charge un programme en mémoire, celui-ci est divisé en un certain nombre de pages. Si L est la taille du programme et taille_page la taille d'une page, alors le nombre de pages p nécessaires est donné par la formule suivante : $p = L/taille\_page$. Autrement dit, p est le plus petit entier supérieur ou égal à L/taille_page. Chaque page du programme est ensuite placée dans un cadre de page libre en mémoire centrale, et ces cadres n'ont pas besoin d'être contigus, ce qui permet une utilisation plus efficace des espaces mémoire disponibles.

Cependant, dans un système de pagination pure, un programme ne peut être chargé que s'il y a suffisamment de cadres de page libres pour accueillir toutes ses pages. Si la taille L du programme n'est pas un multiple exact de la taille_page, la dernière page peut ne pas être entièrement remplie, laissant un espace vide inutilisable, appelé fragmentation interne. Cette fragmentation interne est une conséquence directe de la rigidité du découpage imposé par la pagination.

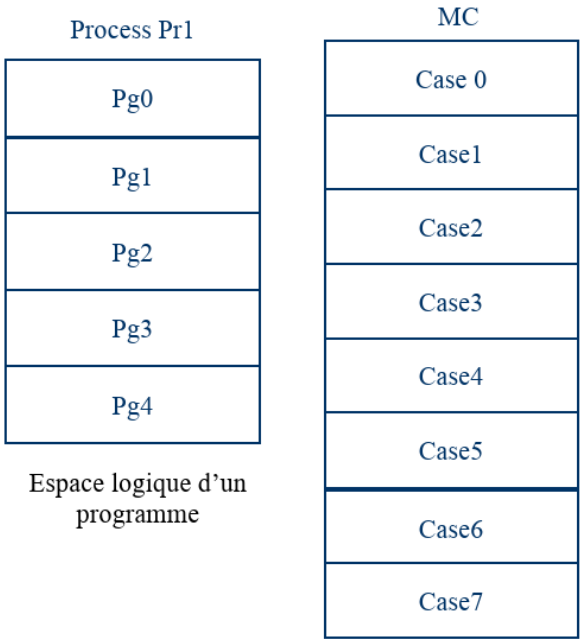


Figure 30 Présentation des pages de programme et les cadre de mémoire centrale

Exemple : Dans un système de pagination, deux processus P1 et P2 peuvent être chargés en mémoire centrale (MC) de manière non contiguë. Supposons que le processus P1 occupe 4 pages logiques et le processus P2 en occupe 2. Ces pages logiques sont mappées à des cadres de pages physiques dans la mémoire centrale. Par exemple, les pages 1 à 4 de P1 peuvent être mappées respectivement aux cadres de pages 5, 0, 2 et 3, tandis que les pages 1 et 2 de P2 peuvent être mappées aux cadres 1 et 6. Grâce à la table des pages, chaque processus peut accéder à ses pages logiques qui sont dispersées dans la mémoire physique, sans se soucier de la contiguïté.

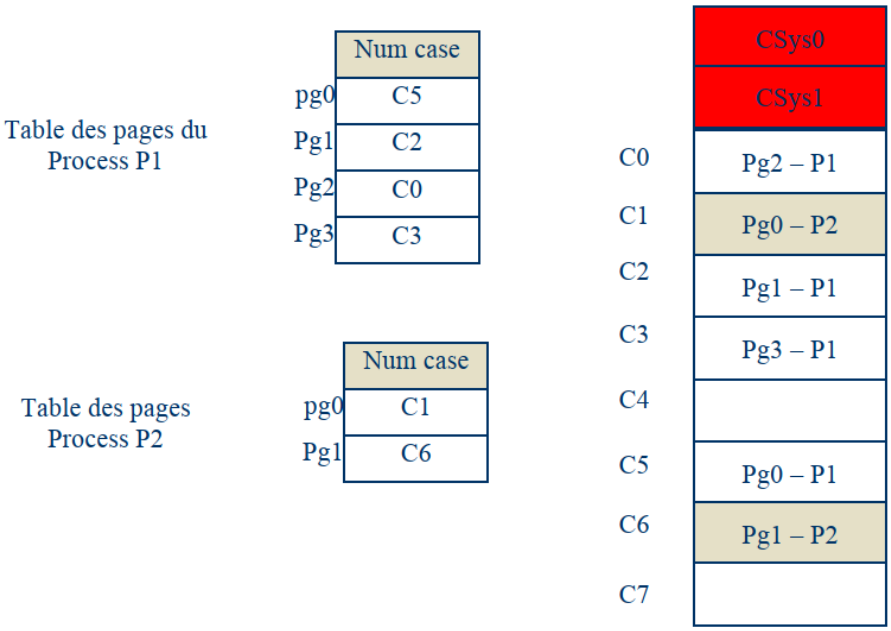


Figure 31 E xemple de pagination

IV.3.2.2 Translation d'adresses

L'association d'une page logique (dans le programme) à une page physique (dans la mémoire) est gérée par une structure de données appelée table des pages (Page Table). Cette table stocke les correspondances entre les numéros de pages logiques et les cadres de pages physiques où les pages sont réellement stockées en mémoire centrale. Le processus de traduction des adresses logiques en adresses physiques est effectué par l'unité de gestion de la mémoire, connue sous le nom de Memory Management Unit (MMU).

Le support matériel pour la pagination, tel que décrit, se base sur le découpage de chaque adresse logique, générée par l'unité centrale (UC), en deux parties distinctes :

- ✓ Le numéro de page (P, Page Number) : Cette partie de l'adresse sert d'index dans la table des pages. La table contient, pour chaque page logique, l'adresse de base correspondante dans la mémoire physique.

- ✓ Le déplacement dans la page (D, Page Offset) : Une fois le numéro de page résolu en une adresse de base physique à partir de la table des pages, le déplacement (offset) est ajouté à cette adresse de base pour obtenir l'adresse physique finale. Cette adresse physique est ensuite utilisée pour accéder à la donnée ou à l'instruction en mémoire centrale.

En combinant le numéro de page et le déplacement, la MMU peut traduire efficacement une adresse logique en une adresse physique, permettant ainsi l'accès à la mémoire centrale sans que les pages logiques aient besoin d'être contiguës dans la mémoire physique.

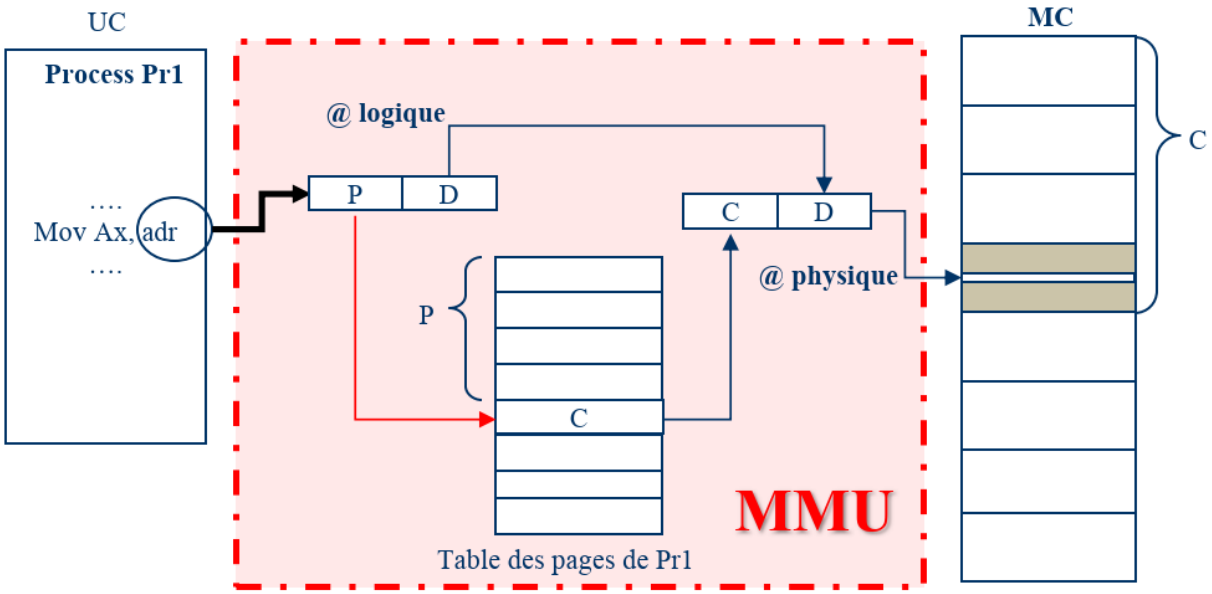
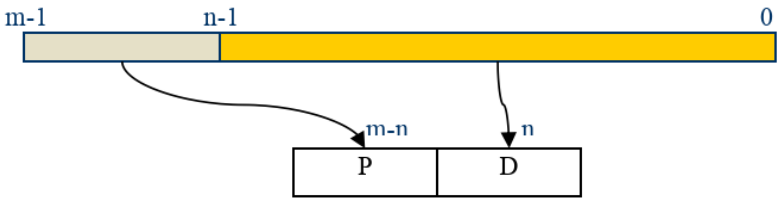


Figure 32 présentation de la translation des adresse logiques en physiques

Pour un espace d'adressage logique de 2^m octets, en considérant des pages de 2^n octets, les m - n premiers bits d'une adresse logique correspondent au numéro de page P et les n bits restants au déplacement D dans la page.



Donc, si T est la taille d'une page et U une adresse logique, alors l'adresse paginée (P, D) est déduite à partir des formules suivantes :

P = U Div T (où, Div est la division entière)

D = U Mod T (où, Mod est le reste de la division)

Exemple : Soit une adresse demandée par le programme AB = 1010 1011 sur 8 bits, m=8 bits. Nous avons aussi des pages de taille 2^6 mots. Donc le nombre de pages est de $2^{m-n} = 2^{8-6} = 4$.

Alors l'adresse logique devienne adresse paginée <P, D> comme suit :

➢ P= 171 Div 64 = 2, P=2. ➔ Le numéro de la page est 2

➢ D =171 Mod 64 = 43, ➔ D=43 est le déplacement (offset) dans la page numéro 2.

1010 1011 → <2 , 43>

IV.3.3 Segmentation de la Mémoire Centrale

IV.3.3.1 Principe de la segmentation

Le principe de la segmentation diffère de celui de la pagination en ce qu'il correspond mieux à la manière dont l'utilisateur conçoit un programme. Dans la segmentation, un programme est vu comme un ensemble d'unités logiques distinctes telles que le code, les données, la pile, le programme principal, et les procédures, souvent appelées segments. Contrairement à la pagination, où la mémoire centrale (MC) est découpée en blocs de taille fixe, la segmentation divise l'espace d'adressage d'un programme en segments de tailles variables, chacun correspondant à une structure logique spécifique du programme selon la perspective de l'utilisateur. Chaque segment est stocké dans un espace mémoire qui lui est propre, et les adresses à l'intérieur d'un segment sont relatives à son début.

La segmentation offre plusieurs avantages, notamment en facilitant l'édition de liens et en permettant le partage de segments de données ou de code entre plusieurs processus. Un autre avantage est l'élimination de la fragmentation interne, car l'espace mémoire alloué à chaque segment correspond exactement à sa taille. Cependant, la segmentation peut entraîner une fragmentation externe en raison de l'allocation dynamique de l'espace mémoire, où des espaces libres non contigus peuvent apparaître dans la mémoire centrale.

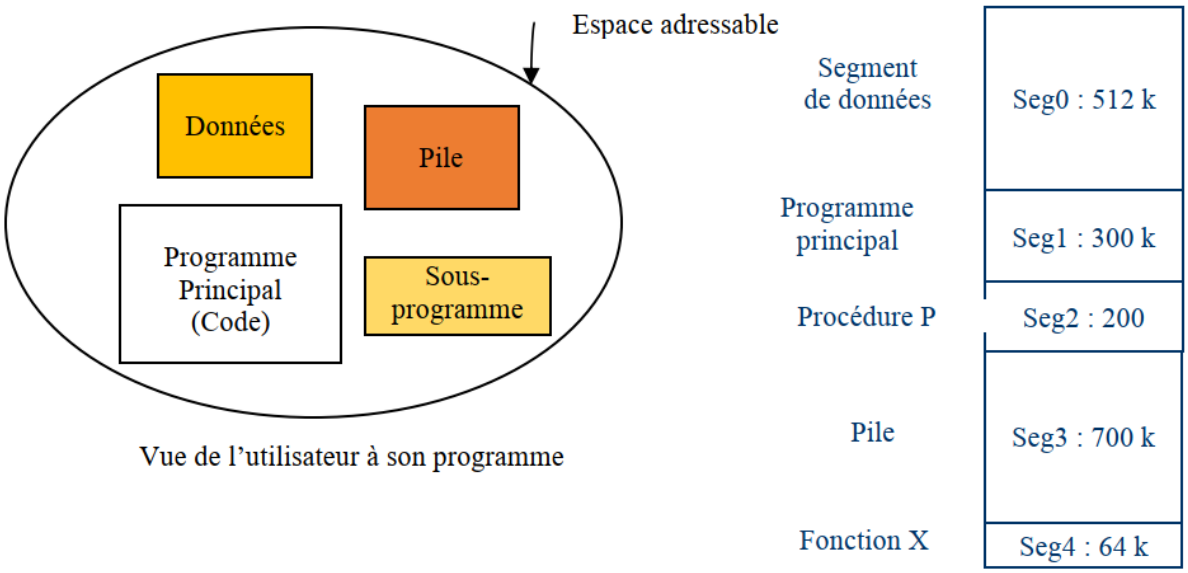


Figure 33 Présentation d'un exemple de segmentation

IV.3.3.2 Adressage en segmentation

Chaque segment dans un système segmenté est identifié par un numéro S et possède une longueur variable L . Un segment est constitué d'un ensemble d'adresses logiques contiguës. Une adresse logique est représentée par un couple (S, D) , où S désigne le numéro du segment et D représente le déplacement à l'intérieur du segment.

Pour associer une adresse logique à une adresse physique, une table appelée "table de segments" est utilisée. Chaque entrée dans cette table contient deux informations essentielles : le segment de base et le segment limite. Le segment de base indique l'adresse physique où commence le segment en mémoire, tandis que le segment limite spécifie la longueur maximale du segment. La conversion des adresses logiques en adresses physiques est gérée par l'unité de gestion mémoire (MMU). Pour qu'une adresse logique soit valide, le déplacement D doit être compris entre 0 et la limite L du segment. Si cette condition est remplie, l'adresse physique est calculée en ajoutant D à la base du segment. Dans le cas contraire, une erreur d'adressage est déclenchée.

Ainsi, dans un système segmenté, chaque instruction référencée par une adresse logique est de la forme <S, D> . Le gestionnaire de mémoire doit suivre la position de chaque segment S en mémoire centrale (MC). Pour cela, une structure logicielle, appelée table des segments, est créée pour chaque processus. Cette table fait correspondre chaque segment chargé en mémoire avec son emplacement physique ainsi que sa taille.

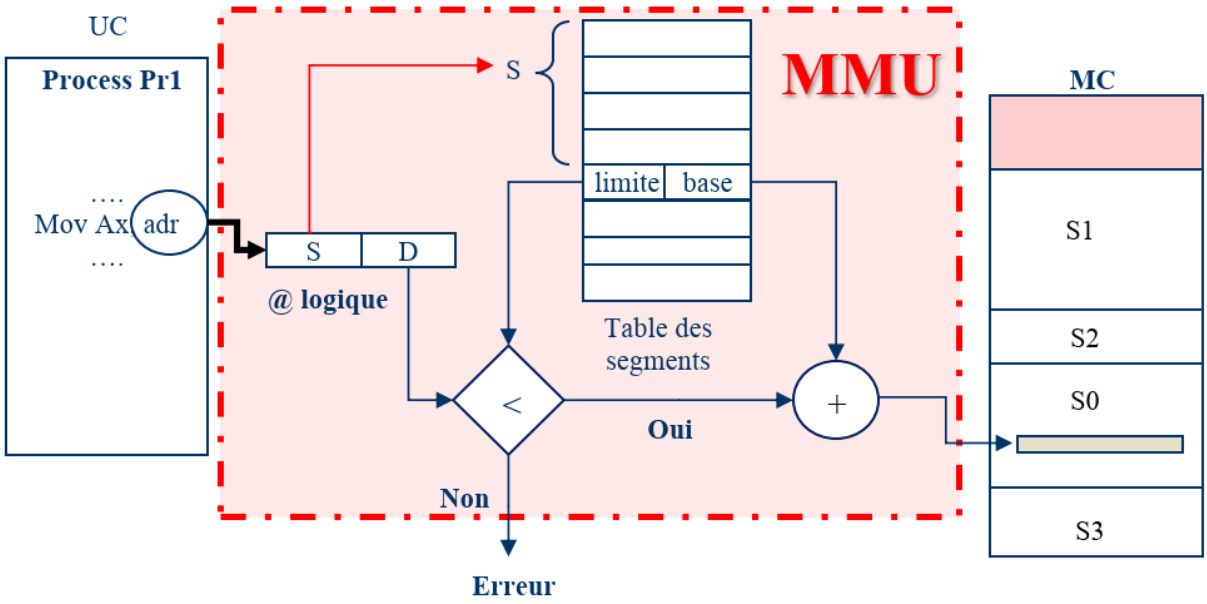


Figure 34 Translation d'adresse en segmentation

Exemple : Sur un système utilisant la segmentation simple, on suppose que les adresses soient décimales au lieu de binaires, calculez l'adresse physique de chacune des adresses logiques, à partir de la table des segments ci-après :

Segment	Base	Limite
0	1100	500
1	2500	1000
2	200	600
3	4000	1200

La correspondance des adresses Logique-physique de chacune des adresses logiques est représentée ci-après :

Adresse logique	Adresse physique
<0, 300>	300<500, donc @=1100+300=1400
<2, 800>	800>600, donc Erreur d'adressage
<1, 600>	600<1000, donc @=2500+600=3100
<3, 1100>	1100<1200, donc @=4000+1100=5100
<1, 1111>	1111>1000, donc Erreur d'adressage

IV.3.3.3 Critiques

La segmentation, bien qu'elle offre des avantages en termes de gestion de la mémoire, présente plusieurs critiques et limitations :

- ✓ Fragmentation Externe :
    - Problème : La segmentation, en permettant à chaque segment d'avoir une taille variable et de se placer de manière non contiguë en mémoire, peut entraîner une fragmentation externe. Ce type de fragmentation se manifeste par la création de nombreux petits espaces libres non contigus qui ne peuvent pas être utilisés efficacement pour les nouveaux segments. Même si la somme de ces espaces est suffisante pour un segment, leur répartition non contiguë les rend inutilisables.
    - Conséquence : Cette fragmentation externe peut conduire à une sous-utilisation de la mémoire et à une diminution des performances, car il devient difficile de trouver des espaces suffisamment grands et contigus pour de nouveaux segments.
- ✓ Complexité de Gestion :
    - Problème : La gestion des segments est plus complexe que celle des pages. Chaque segment peut avoir une taille différente et doit être géré individuellement, ce qui nécessite un suivi précis de l'emplacement et de la taille de chaque segment en mémoire.
    - Conséquence : La table des segments, qui associe les adresses logiques aux adresses physiques, peut devenir très grande et complexe, particulièrement dans les systèmes avec de nombreux processus et segments. Cette complexité peut entraîner des coûts supplémentaires en termes de gestion de la mémoire et de performance du système.
- ✓ Difficulté de Partage et d'Isolation :
    - Problème : Bien que la segmentation facilite le partage de segments entre différents processus, elle peut également compliquer l'isolation entre processus. Les segments partagés doivent être correctement protégés pour éviter que les processus n'interfèrent les uns avec les autres, ce qui peut poser des défis en matière de sécurité et de gestion de la mémoire.
    - Conséquence : Une mauvaise gestion du partage et de la protection des segments peut entraîner des conflits entre processus, des violations de sécurité et des erreurs de mémoire, ce qui affecte la stabilité et la sécurité du système.
- ✓ Chargement Dynamique et Déplacement :
    - Problème : Les segments doivent être chargés et déplacés dynamiquement dans la mémoire, ce qui peut entraîner des coûts en termes de temps et de performance. La nécessité de gérer l'emplacement physique des segments à tout moment peut ralentir les opérations de chargement et de déplacement.
    - Conséquence : Ce besoin constant d'ajustement et de suivi des segments en mémoire peut entraîner une surcharge pour le gestionnaire de mémoire, réduisant ainsi l'efficacité globale du système.

En somme, bien que la segmentation offre une vue plus flexible et logique de la mémoire en alignant la gestion de la mémoire avec la structure des programmes, ses problèmes de fragmentation externe, sa complexité de gestion, ses défis en termes de partage et d'isolation, ainsi que les coûts associés au chargement et au déplacement des segments peuvent limiter son efficacité et sa praticité dans certains contextes.

IV.3.4 Segmentation paginée

La segmentation paginée est une technique de gestion de la mémoire qui combine les principes de la segmentation et de la pagination pour surmonter certaines limitations associées à chacune de ces méthodes. Cette approche est particulièrement utile dans les systèmes où les segments de programme peuvent être très volumineux, entraînant des temps de chargement significatifs si la segmentation seule est utilisée.

Dans le système de segmentation paginée , un programme est d'abord divisé en segments, chaque segment étant ensuite découpé en pages. Cette technique permet de bénéficier des avantages de la segmentation (comme la gestion logique des différentes parties d'un programme) tout en utilisant la pagination pour gérer les segments eux-mêmes. Cette méthode a été développée pour améliorer l'efficacité des systèmes, comme le système Multicast.

Pour chaque processus, on maintient à la fois une table de segments et une table de pages . Contrairement à une adresse de mémoire directe, une adresse de segment n'est pas une adresse physique mais plutôt une référence dans la table des pages du segment correspondant. Lorsqu'une adresse logique est spécifiée sous la forme d'un couple (S, D), où S représente le numéro du segment et D le déplacement au sein du segment, elle est convertie en un triplet (S, P, d). Dans ce triplet, P est le numéro de page au sein du segment S, et d est le déplacement à l'intérieur de la page P. Ce processus de traduction permet d'effectuer la conversion d'une adresse logique en une adresse physique de manière plus efficace et organisée. La traduction d’une adresse logique en adresse physique se fait selon le schéma de la figure 16.

La segmentation paginée , bien que combinant les avantages de la segmentation et de la pagination, présente plusieurs inconvénients. L'une des principales critiques concerne sa complexité accrue, résultant de la gestion simultanée de deux niveaux de tables de traduction : la table des segments et la table des pages. Cette complexité peut entraîner une surcharge significative en termes de gestion et de mémoire pour maintenir ces tables. De plus, le découpage d'un segment en pages peut mener à une fragmentation interne au niveau des pages, où l'espace non utilisé dans la dernière page d'un segment reste inexploitée, malgré la flexibilité apportée par la segmentation. Enfin, le processus de traduction d'adresses, bien que plus flexible, peut introduire des délais supplémentaires et augmenter le

temps nécessaire pour résoudre les adresses logiques en adresses physiques, impactant ainsi les performances globales du système

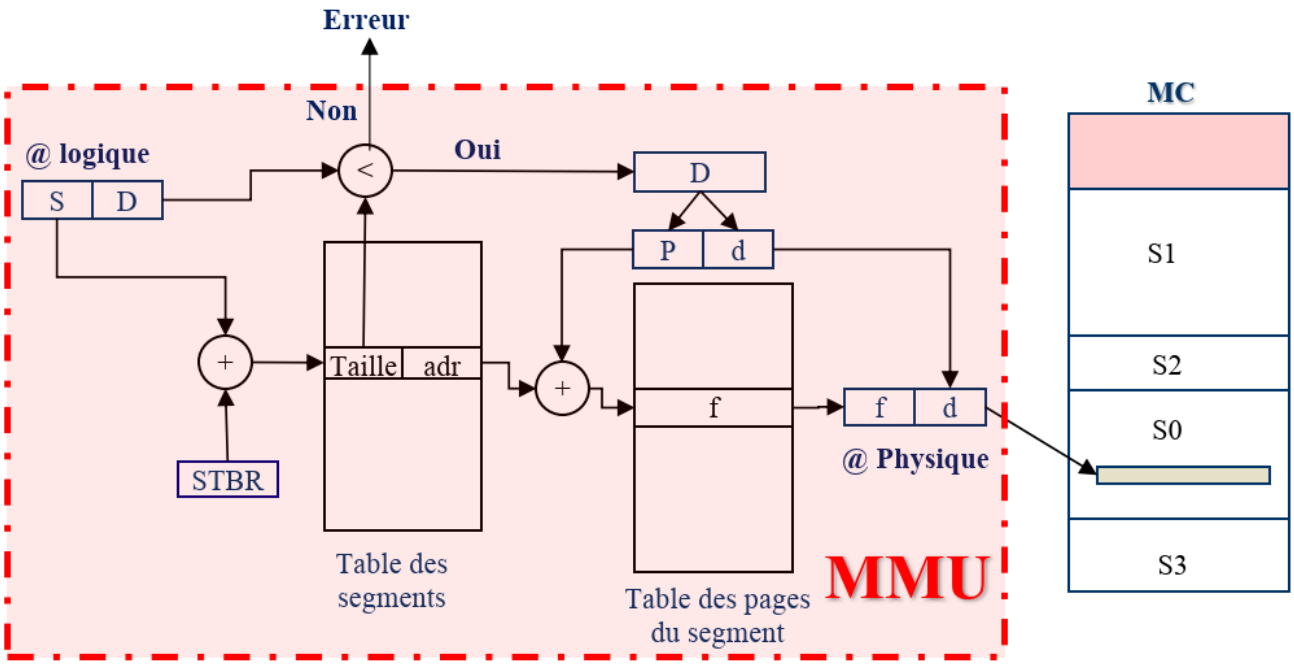


Figure 35 Traduction d'adresse en segmentation paginée

IV.3.5 Mémoire virtuelle

IV.3.5.1 Principe de Mémoire virtuelle

Les systèmes de pagination et de segmentation, bien qu'efficaces, présentent plusieurs limitations importantes. Tout d'abord, pour qu'un programme puisse être chargé en mémoire, il doit y avoir un nombre de cases libres égal au nombre de pages du programme. Cela signifie qu'un programme dont la taille dépasse la capacité totale de la mémoire centrale (MC) ne pourra pas être exécuté, car il ne pourra pas être entièrement chargé. De plus, même lorsqu'un processus est actif, il est possible que seules certaines pages ou segments soient réellement utilisés à un moment donné. Cependant, toutes les pages ou segments nécessaires doivent être chargés en mémoire, ce qui peut entraîner une sous-utilisation de la mémoire pour les processus en attente.

Pour pallier ces limitations, la mémoire virtuelle a été développée. Cette technique permet à des processus plus grands que la capacité physique de la MC d'être exécutés en simulant une mémoire plus grande que la mémoire physique réelle. La mémoire virtuelle fonctionne en utilisant un espace de stockage secondaire, comme un disque dur, pour stocker des parties du programme qui ne tiennent pas dans la MC. Pendant l'exécution, seuls certains morceaux du programme sont chargés dans la MC, tandis que d'autres restent dans la mémoire secondaire. Lorsque le programme a besoin de ces parties, elles sont chargées

dans la MC, éventuellement en échangeant des morceaux déjà présents dans la MC si nécessaire.

Ainsi, la mémoire virtuelle permet au système d'exploiter la mémoire physique et la mémoire secondaire comme une seule entité, offrant une illusion de mémoire infinie au processus utilisateur. Les échanges entre la MC et la mémoire secondaire se font de manière transparente pour l'utilisateur, permettant l'exécution efficace de programmes qui dépassent la taille de la MC réelle.

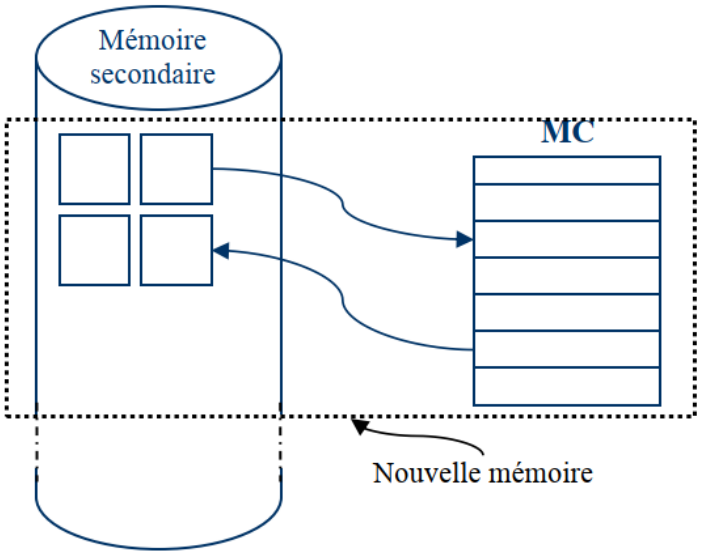


Figure 36 Mémoire virtuelle

IV.3.5.2 Technique de pagination à la demande (On-Demand Paging)

Le principe de la mémoire virtuelle est souvent mis en œuvre à l'aide de la technique de pagination à la demande (On-Demand Paging). Dans ce système, les pages d'un programme ne sont chargées en mémoire centrale (MC) que lorsque le processeur tente d'y accéder, plutôt que d'être toutes chargées au démarrage du programme. Ce mécanisme est comparable à un système de pagination avec va-et-vient, ou swapping, où les pages sont transférées entre la mémoire centrale et le disque selon les besoins.

Pour gérer cette technique, le système d'exploitation utilise une table des pages contenant un bit supplémentaire appelé bit de validation (V). Ce bit indique l'état de chaque page par rapport à la mémoire centrale. Si le bit de validation est réglé sur 1 (V=1), cela signifie que la page est actuellement chargée dans la mémoire centrale et est donc accessible au processeur. En revanche, si le bit est réglé sur 0 (V = 0), cela indique que la page n'est pas en mémoire centrale et que toute tentative d'accès à cette page entraînera une erreur, nécessitant un chargement de la page depuis le disque avant de poursuivre l'exécution.

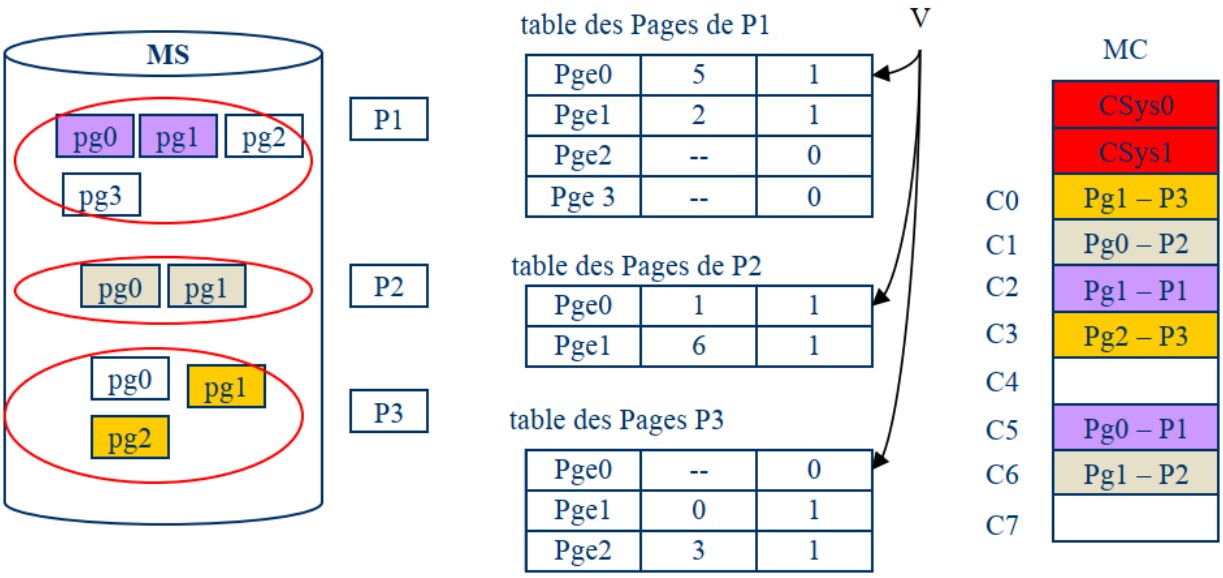


Figure 37 Présentation de mémoire virtuelle avec pagination à la demande

Lorsque le processeur tente d'accéder à une adresse mémoire spécifique, la MMU convertit cette adresse en un couple <P, D>, où P représente le numéro de page et D le déplacement au sein de la page. La MMU utilise le numéro de page (P) pour indexer la table des pages et vérifier l'état de la page à l'aide du bit de validation (V). Si le bit V est à 1, cela indique que la page est chargée en mémoire centrale (MC) et l'accès peut continuer normalement. En revanche, si le bit V est à 0, cela signifie que la page n'est pas en mémoire, ce qui entraîne un défaut de page.

Lorsqu'un processus tente d'accéder à une page qui n'est pas en mémoire, le système déclenche un défaut de page (Page Fault). Ce défaut signale au système d'exploitation qu'une page nécessaire n'est pas chargée en mémoire. La gestion de ce défaut suit plusieurs étapes cruciales :

- ✓ Vérification de la Référence : Le système d'exploitation vérifie d'abord que l'accès à la page est valide et que la référence est correcte.
- ✓ Vérification en Mémoire Auxiliaire : Il s'assure que la page demandée est bien stockée dans la mémoire auxiliaire (souvent un disque dur).
- ✓ Chargement de la Page : Le système trouve un cadre de page libre dans la mémoire centrale, charge la page depuis la mémoire auxiliaire dans ce cadre, et met à jour la table des pages pour refléter la nouvelle présence de la page en mémoire.

Ainsi, la procédure permet de gérer efficacement l'accès aux pages et de maintenir l'intégrité du système de gestion de la mémoire.

IV.3.5.3 Stratégies de remplacement de pages

Lorsque le SE se rend compte au moment de charger une page qu'il n'existe aucun cadre de page disponible, il peut faire recours à un remplacement de page.

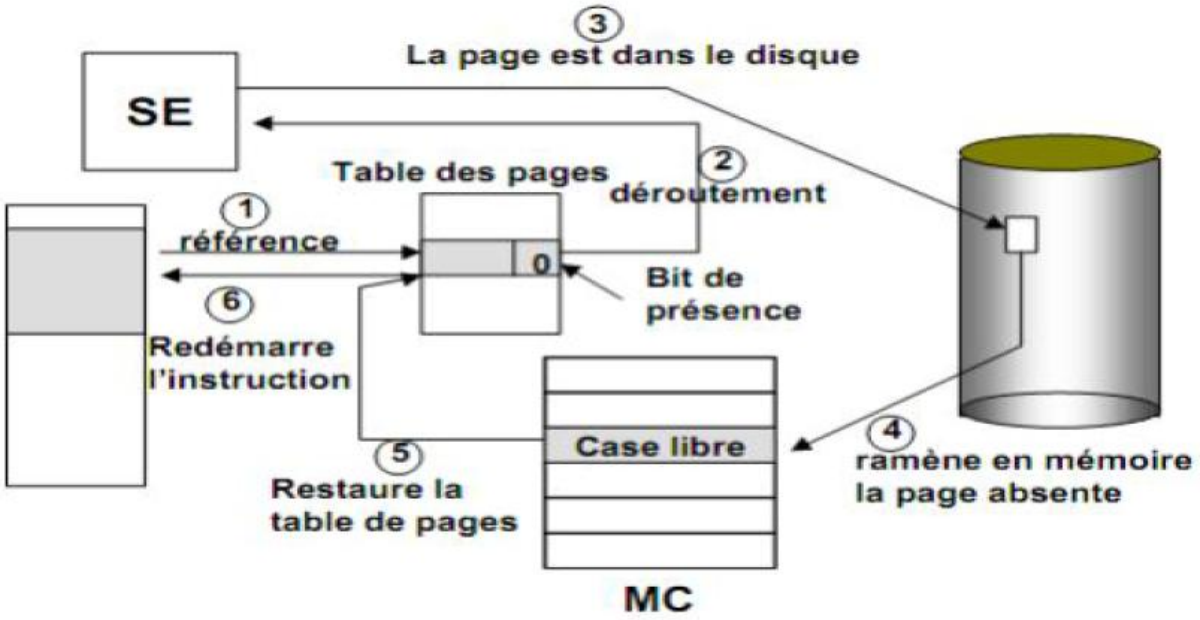


Figure 38 Procédure de traitement d'un défaut de pages

L'objectif principal des algorithmes de remplacement de pages est de minimiser le taux de défauts de page, c'est-à-dire le nombre d'occurrences où un programme tente d'accéder à une page qui n'est pas actuellement en mémoire. Un défaut de page survient lorsque la page demandée est absente de la mémoire centrale et doit être chargée depuis un espace de stockage secondaire, comme le disque. Pour atteindre cet objectif, il est essentiel de choisir un algorithme de remplacement de pages qui gère efficacement les situations où la mémoire est pleine et qu'une nouvelle page doit remplacer une page déjà présente.

Pour évaluer la performance d'un algorithme de remplacement de pages, on utilise une séquence particulière de références mémorielles, appelée chaîne de références , qui représente l'ordre dans lequel un processus accède à ses pages. En exécutant l'algorithme sur cette chaîne de références, on peut déterminer combien de fois le système doit remplacer une page et, par conséquent, le nombre de défauts de page générés. Un algorithme efficace doit réduire ce nombre autant que possible, en optimisant la gestion des pages en mémoire.

En outre, le nombre de cadres de pages disponibles dans la mémoire centrale influence le taux de défauts de page. En général, en augmentant le nombre de cadres de pages disponibles, on réduit le nombre de défauts de page, car plus de pages peuvent être maintenues en mémoire simultanément, diminuant ainsi la nécessité de remplacer des pages et les coûts associés aux accès au stockage secondaire.

IV.3.5.4 Algorithmes de remplacement de pages

Il existe divers algorithmes de remplacement de pages, chacun ayant pour objectif de minimiser le taux de défauts de pages. Pour évaluer l'efficacité d'un algorithme, on le teste sur une séquence spécifique de références mémoire et on calcule le nombre de défauts de pages qu'il génère.

À un moment donné, la séquence des numéros de pages référencées par un processus depuis le début de son exécution est appelée la chaîne de références du processus. En connaissant cette chaîne de références, on peut anticiper le comportement futur du programme.

- ✓ Algorithme FIFO (First-In-First-Out) : Remplace la page qui est entrée la première en mémoire lorsqu'une nouvelle page doit être chargée. Cet algorithme est simple mais peut ne pas être optimal, car la page la plus ancienne n'est pas nécessairement la moins utilisée.

- ✓ Algorithme LRU (Least Recently Used) : Remplace la page qui n'a pas été utilisée depuis le plus longtemps. L'idée est que les pages récemment utilisées ont plus de chances d'être utilisées à nouveau, tandis que celles qui ne l'ont pas été sont moins susceptibles de l'être.

- ✓ Algorithme Optimal (OPT) : Remplace la page qui ne sera pas utilisée pendant la période la plus longue dans le futur. Cet algorithme est théoriquement optimal, mais il nécessite une connaissance préalable de la séquence de références futures, ce qui le rend difficile à mettre en œuvre en pratique.

- ✓ Algorithme de la seconde chance (Second-Chance Algorithm) : Variante de l'algorithme FIFO, il accorde une "seconde chance" aux pages en vérifiant un bit d'utilisation. Si une page a été utilisée récemment, elle n'est pas remplacée et est placée à la fin de la file d'attente.

- ✓ Algorithme NRU (Not Recently Used) : Classe les pages en fonction de leur utilisation récente et de leur statut de modification, puis remplace une page en fonction de ces critères, privilégiant celles qui n'ont pas été utilisées ou modifiées récemment.

Chaque algorithme a ses avantages et inconvénients et peut être plus ou moins efficace selon la nature du programme et la charge de travail.

IV.3.5.5 Exemple

Un processus est divisé en 5 pages, chaque page ayant une taille de 100 unités. Le programme P accède successivement aux adresses mémoire suivantes : 100, 210, 355, 120, 420, 110, 200, 550, 139, 201, 395, 404, 505.

La chaîne de références correspondante des pages est : 1, 2, 3, 1, 4, 1, 2, 5, 1, 2, 3, 4, 5. En supposant une mémoire composée de 4 cadres de pages, Déterminez le nombre de défauts de pages pour chaque algorithme de remplacement de pages ci-dessous :

- ✓ FIFO (First-In, First-Out) : Remplacement de la page la plus ancienne.

Demandes	1	2	3	1	4	1	2	5	1	2	3	4	5
Case 1	1	1	1	1	1	1	1	5	5	5	5	4	4
Case 2		2	2	2	2	2	2	2	1	1	1	1	5
Case 3			3	3	3	3	3	3	3	2	2	2	2
Case 4					4	4	4	4	4	4	3	3	3
Défaut de page	D	D	D		D			D	D	D	D	D	D

9 défauts de pages

- ✓ Optimal : Remplacement de la page qui ne sera pas utilisée pendant la période la plus longue à venir.

Demandes	1	2	3	1	4	1	2	5	1	2	3	4	5
Case 1	1	1	1	1	1	1	1	1	1	1	1	4	4
Case 2		2	2	2	2	2	2	2	2	2	2	2	2
Case 3			3	3	3	3	3	3	3	3	3	3	3
Case 4					4	4	4	5	5	5	5	5	5
Défaut de page	D	D	D		D			D				D	

6 défauts de pages

- ✓ LRU (Least Recently Used) : Remplacement de la page qui n'a pas été utilisée depuis le plus longtemps.

Demandes	1	2	3	1	4	1	2	5	1	2	3	4	5
Case 1	1	1	1	1	1	1	1	1	1	1	1	1	5
Case 2		2	2	2	2	2	2	2	2	2	2	2	2
Case 3			3	3	3	3	3	5	5	5	5	4	4
Case 4					4	4	4	4	4	4	3	3	3
Défaut de page	D	D	D		D			D			D	D	D

8 défauts de pages

- ✓ NFU (Not Frequently Used) : Remplacement de la page la moins souvent utilisée.

Demandes	1	2	3	1	4	1	2	5	1	2	3	4	5
Case 1	1	1	1	1	1	1	1	1	1	1	1	1	1
Case 2		2	2	2	2	2	2	2	2	2	2	2	2
Case 3			3	3	3	3	3	5	5	5	5	4	4
Case 4					4	4	4	4	4	4	3	3	5
Défaut de page	D	D	D		D			D			D	D	D

8 défauts de pages

- ✓ MFU (Most Frequently Used) : Remplacement de la page la plus fréquemment utilisée.

Demandes	1	2	3	1	4	1	2	5	1	2	3	4	5
Case 1	1	1	1	1	1	1	1	5	5	5	5	4	4
Case 2		2	2	2	2	2	2	2	1	1	1	1	5
Case 3			3	3	3	3	3	3	3	2	2	2	2
Case 4					4	4	4	4	4	4	3	3	3
Défaut de page	D	D	D		D			D	D	D	D	D	D

10 défauts de pages

- ✓ Algorithme de la seconde chance : Variante de FIFO, donnant une "seconde chance" aux pages récemment utilisées.

Demandes	1	2	3	1	4	1	2	5	1	2	3	4	5
Case 1	1	1	1	1	1	1	1	1	1	1	1	4	4
Case 2		2	2	2	2	2	2	2	2	2	2	2	2
Case 3			3	3	3	3	3	5	5	5	5	5	5
Case 4					4	4	4	4	4	4	3	3	3
Défaut de page	D	D	D		D			D			D	D	

7 défauts de pages

IV.3.5.6 L'anomalie de Belady

L'anomalie de Belady est un phénomène paradoxal observé dans les systèmes de gestion de mémoire paginée, où l'augmentation du nombre de cadres de pages en mémoire peut entraîner une augmentation du nombre de défauts de pages, au lieu de le réduire comme on pourrait s'y attendre. Ce comportement inattendu se manifeste principalement avec certains algorithmes de remplacement de pages, notamment l'algorithme FIFO (First-In, First-Out).

En théorie, augmenter le nombre de cadres de pages disponibles pour un processus devrait permettre de conserver davantage de pages en mémoire, ce qui réduirait normalement le nombre de défauts de pages. Cependant, avec l'anomalie de Belady, il arrive que l'augmentation du nombre de cadres de pages entraîne une augmentation du nombre total de défauts de pages. Par exemple, dans une séquence donnée de références de pages, le nombre de défauts peut paradoxalement augmenter lorsqu'on passe d'une mémoire avec moins de cadres à une mémoire avec plus de cadres, surtout avec l'algorithme FIFO.

Pour illustrer ce phénomène, considérons une séquence de références de pages et comparons le nombre de défauts de pages pour différentes tailles de mémoire en utilisant l'algorithme FIFO. Avec une mémoire contenant 3 cadres, il est possible d'observer un nombre donné de défauts de pages. Lorsque la taille de la mémoire est augmentée à 4

cadres, le nombre de défauts de pages peut en fait augmenter au lieu de diminuer, ce qui illustre l'anomalie de Belady. Cela se produit parce que l'algorithme FIFO remplace simplement la page la plus ancienne, sans tenir compte de la fréquence ou de l'ordre futur des références des pages. En conséquence, certaines pages cruciales peuvent être évincées prématurément, ce qui augmente les défauts de pages.

Ce phénomène souligne que les algorithmes de remplacement de pages ne sont pas tous équivalents et que certains, comme LRU (Least Recently Used) ou l'algorithme Optimal, sont conçus pour éviter de tels comportements paradoxaux. Ces algorithmes tiennent compte de l'utilisation récente ou future des pages, réduisant ainsi la probabilité de défauts de pages supplémentaires lors de l'augmentation des cadres de pages.

V. Conclusion

En conclusion, ce chapitre a offert une vue d'ensemble approfondie de la gestion de la mémoire centrale dans les systèmes informatiques, en abordant tant les environnements monoprogrammés que multiprogrammés. Nous avons commencé par une introduction aux concepts de base, avant de plonger dans les objectifs du gestionnaire de mémoire et la définition de l'espace d'adressage.

Dans le contexte des systèmes monoprogrammés, nous avons exploré les méthodes de gestion telles que le swapping et la technique de recouvrement, qui permettent de gérer efficacement la mémoire lorsque plusieurs processus ne sont pas simultanément en cours d'exécution. En passant aux systèmes multiprogrammés, nous avons examiné les différentes stratégies d'allocation de la mémoire centrale, en distinguant l'allocation contiguë par partitions fixes et variables, ainsi que les critiques associées à ces méthodes.

Nous avons ensuite abordé l'allocation non contiguë, avec une attention particulière à la pagination, à la segmentation et à la segmentation paginée. Ces techniques permettent une gestion plus flexible de la mémoire et facilitent la mise en œuvre de la mémoire virtuelle, un concept essentiel dans les systèmes modernes pour maximiser l'utilisation des ressources et améliorer les performances.

L'analyse des différentes techniques de gestion de mémoire révèle que chaque approche a ses avantages et inconvénients, et le choix de la méthode appropriée dépend du contexte d'application et des exigences spécifiques du système. La compréhension des mécanismes sous-jacents, tels que les algorithmes de remplacement de pages et les anomalies comme celle de Belady, est cruciale pour optimiser la gestion de la mémoire et améliorer l'efficacité des systèmes informatiques.

Ainsi, ce chapitre souligne l'importance de la gestion de la mémoire dans le bon fonctionnement des systèmes informatiques et prépare le terrain pour une exploration plus approfondie des techniques avancées et des défis futurs dans ce domaine essentiel.

VI. Exercices corrigés

Questions de cours

1. Définissez la mémoire centrale. Pourquoi est-elle essentielle dans un système d'exploitation ?
2. Quels sont les objectifs principaux d'un gestionnaire de mémoire ?
3. Expliquez ce qu'est l'espace d'adressage.
4. Qu'est-ce que le swapping et dans quels cas est-il utilisé ?
5. Comparez la gestion de la mémoire dans les systèmes monoprogrammés et multiprogrammés.
6. Quels sont les avantages et les inconvénients de l'allocation contiguë par partitions fixes et variables ?
7. Décrivez la pagination et son rôle dans la gestion de la mémoire.
8. Quelle est la différence entre la segmentation et la pagination ?
9. Expliquez le concept de mémoire virtuelle et son importance.
10. Comparez les avantages et inconvénients de l'allocation contiguë par partitions fixes et partitions variables. Discutez de l'impact de chaque méthode sur la fragmentation de la mémoire.
11. Écrivez un algorithme simple en pseudo-code pour la gestion de la pagination dans un système de gestion de mémoire. Décrivez comment l'algorithme traite les demandes de pages et la gestion des pages en mémoire.
12. Un système monoprogrammé utilise la méthode de swapping pour gérer les processus en mémoire. Décrivez un scénario où le swapping est utilisé, et discutez des effets sur la performance du système. Quels seraient les avantages et inconvénients dans ce cas précis ?
13. Décrivez comment la mémoire virtuelle fonctionne et comment elle permet à un système d'exploitation de gérer des applications plus grandes que la mémoire physique disponible. Incluez une discussion sur les tables de pages et les mécanismes de translation d'adresses.

QCM

1. Quel est le rôle principal du gestionnaire de mémoire ?
   - a) Augmenter la vitesse du processeur
   - b) Gérer l'allocation et la libération de la mémoire
   - c) Contrôler l'accès aux fichiers
   - d) Gérer les périphériques externes

2. Dans quel type de gestion de mémoire utilise-t-on des partitions de taille fixe?
   - a) Allocation non contiguë
   - b) Allocation contiguë par partitions variables
   - c) Allocation contiguë par partitions fixes
   - d) Pagination

3. Le swapping est principalement utilisé pour :
   - a) Libérer de l'espace sur le disque dur
   - b) Améliorer la sécurité des processus
   - c) Transférer des processus entre la mémoire centrale et le disque dur
   - d) Gérer l'allocation de mémoire virtuelle

4. La segmentation de la mémoire centrale permet de :
   - a) Allouer la mémoire en blocs de taille fixe
   - b) Séparer les données et le code dans des segments distincts
   - c) Allouer la mémoire sans fragmentation externe
   - d) Utiliser des partitions dynamiques

5. La mémoire virtuelle permet à un système d'exploitation de :
   - a) Accroître la capacité mémoire au-delà de la mémoire physique
   - b) Supprimer les erreurs de segmentation
   - c) Optimiser les processus d'entrées/sorties
   - d) Gérer les accès concurrents aux fichiers

Exercices

Exercice 1 :

Soit un système ayant 4 cases mémoire, la taille d'une page = 100. - Un Programme P fait successivement référence aux adresses suivantes : 100, 210, 355, 120,420, 110, 200, 550, 139, 201, 395, 404, 505.

1) Donner la chaîne de références aux pages qui correspondent aux adresses.

2) Calculer le nombre de défauts de pages en appliquant la stratégie MFU, FIFO, Optimal.

Exercice 2 :

On dispose d'un système doté d'une pagination à la demande, suivant deux algorithmes A1 et A2.
Au cours de son exécution, un programme accède successivement aux pages 1, 5, 2, 5, 1, 4, 1, 5, 3.
Le système alloue à ce programme un espace de trois pages.
Avec l'algorithme A1, on constate que l'on a successivement en mémoire les pages suivantes:

```
1 1 1 1 1 2 1 1 1
  5 2 2 2 4 2 4 3
    5 5 5 5 4 5 5
```

Avec l'algorithme A2, on constate que l'on a successivement en mémoire les pages suivantes:

```
1 1 1 1 1 1 1 1 1
  5 2 2 2 4 4 4 3
    5 5 5 5 5 5 5
```

A- A votre avis, lequel des deux algorithmes correspondrait à l'algorithme FIFO, et lequel correspondrait à LRU?
B- Déterminer dans chacun des cas le nombre de défauts de pages.

Exercice 3 :

Considérez une mémoire virtuelle ayant une taille de mémoire physique (principale) de 32 Méga-Octets et supportant une taille de blocs de 512 octets. Aussi, supposez un processus occupant un espace d'adressage logique de 856 Kilo-octets.
1) Calculez le nombre de pages dans l'espace d'adressage logique, et le nombre de cases de l'espace d'adressage physique.
2) Montrez les formats des adresses logiques et physiques. Spécifiez le nombre de bits pour les déplacements, les pages, et les cases.
3) Pour l'adresse logique (virtuelle) 11301, spécifiez son emplacement dans la mémoire physique. Supposez que la page contenant l'adresse 11301 se trouve dans la case 15240.
Considérez la séquence de référence de page 0,1,2,1,2,1,2,1,2,3,4,5, 6, 5, 6, 7 pour un processus.
1) En utilisant le modèle FIFO, représentez l'allocation en mémoire physique des pages pour une taille de 3 cases (ou cadres).et estimez le taux de fautes de pages.

Exercice 4 :

On s'intéresse à la pagination à la demande. Le système dispose de 4 cases qui sont toutes occupées, le tableau donnant ci-dessous, pour chacune d'elles, la date en microsecondes du chargement de la page qu'elle contient, la date en microsecondes du dernier accès à cette page.
Donner quelle sera la page remplacée pour chacun des 4 algorithmes de remplacement suivants : LRU, FIFO, seconde chance

Case	*chargement*	*Accès*	*Accédée*	*modifiée*	*présence*
0	*126*	*279*	*0*	*0*	*1*
1	*230*	*260*	*1*	*0*	*1*
2	*120*	*272*	*1*	*1*	*1*
3	*160*	*280*	*1*	*1*	*1*

Exercice 5 :

On dispose d'un espace adressable virtuel 4 Go (adressable sur 32 bits), et d'un espace physique 32 Mo (adressable sur 25 bits). Une page occupe 1 Ko (déplacement sur 10 bits). Quelle est la taille de la table des pages (en octet), sachant qu'une case de la table de page comporte 1 bit de présence et un n° physique de la page ?

Exercice 6 :

Soit la table de pages suivante , et en Sachant que les pages virtuelles et physiques font 1K octets, quelle est l'adresse mémoire correspondant à chacune des adresses virtuelles suivantes codées en hexadécimal : 142A et 0AF1.

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

Exercice 7 :

On s'intéresse aux systèmes utilisant la pagination

A- Détailler les algorithmes FIFO et LRU.

B- Au cours de son exécution, un programme accède successivement aux pages : 0, 1, 4, 2, 0, 1, 3, 0, 1, 4, 2, 3.

Donner la suite des pages présentes en mémoire ainsi que le nombre de défauts de pages pour chacun des cas suivant :

B.1- Si on utilise l'algorithme FIFO et sachant que le système alloue à ce programme un espace de 3 pages.

B.2- Si on utilise l'algorithme LRU et sachant que le système alloue à ce programme un espace de 3 pages.

B.3- Si on utilise l'algorithme Optimal et sachant que le système alloue à ce programme un espace de 3 pages.

B.4- Si on utilise l'algorithme FIFO et sachant que le système alloue à ce programme un espace de 4 pages.

B.5- Si on utilise l'algorithme LRU et sachant que le système alloue à ce programme un espace de 4 pages.

Exercice 8 :

On dispose d'un système doté d'une pagination à la demande, suivant deux algorithmes A1 et A2, Au cours de son exécution, un programme accède successivement aux pages 1, 5, 2, 5, 1, 4, 1, 5, 3. Le système alloue à ce programme un espace de trois pages.

- Avec l'algorithme A1, on constate que l'on a successivement en mémoire les pages suivantes :

1	1	1	1	1	4	4	4	3
	5	5	5	5	5	1	1	1
		2	2	2	2	2	5	5

- Avec l'algorithme A2, on constate que l'on a successivement en mémoire les pages suivantes :

1	1	1	1	1	1	1	1	1
	5	5	5	5	5	5	5	5
		2	2	2	4	4	4	3

A.- A votre avis, lequel des deux algorithmes correspondrait à l'algorithme FIFO, et lequel correspondrait à LRU ? Justifiez votre raisonnement.

B. – Déterminer dans chacun des cas le nombre de défauts de pages.

Exercice 9 :

Le système dispose de 4 cases qui sont toutes occupées, le tableau ci-dessous donne, pour chacune d'elles, la date en microsecondes du chargement de la page qu'elle contient, la date en microsecondes du dernier accès à cette page et l'état des indicateurs de la case (Read et Modified).

Case	*Chargement*	*Accès*	*R*	*M*
0	126	279	0	1
1	230	260	1	1
2	120	272	1	1
3	160	280	1	1

En justifiant votre réponse, donner quelle sera la page remplacée, pour chacun des 4 algorithmes de remplacement suivants FIFO, LRU, NRU (seconde chance).

VII. Correction des exercices

Questions de cours

1. Définissez la mémoire centrale. Pourquoi est-elle essentielle dans un système d'exploitation ?

La mémoire centrale, aussi appelée mémoire vive ou RAM (Random Access Memory), est l'espace de stockage temporaire utilisé par le système d'exploitation et les applications en cours d'exécution pour stocker des données et des instructions nécessaires au fonctionnement des programmes. Elle est essentielle car elle permet un accès rapide aux données, accélérant ainsi les opérations de traitement.

2. Quels sont les objectifs principaux d'un gestionnaire de mémoire ?

- Les objectifs du gestionnaire de mémoire sont :
  - Allouer la mémoire aux processus de manière efficace.
  - Assurer la protection de la mémoire allouée contre les accès non autorisés.
  - Optimiser l'utilisation de la mémoire pour minimiser la fragmentation et maximiser la performance.
  - Gérer le swapping et la mémoire virtuelle pour permettre l'exécution de processus nécessitant plus de mémoire que la mémoire physique disponible.

3. Expliquez ce qu'est l'espace d'adressage.

L'espace d'adressage est l'ensemble des adresses de mémoire qu'un processus peut utiliser. Dans les systèmes modernes, il inclut à la fois la mémoire physique et virtuelle, permettant au système de mapper les adresses logiques utilisées par les programmes en adresses physiques réelles.

4. Qu'est-ce que le swapping et dans quels cas est-il utilisé ?

Le swapping est une technique où des processus inactifs sont temporairement déplacés de la mémoire centrale vers le disque dur pour libérer de l'espace pour d'autres processus. Il est principalement utilisé lorsque la mémoire centrale est insuffisante pour gérer tous les processus actifs simultanément.

5. Comparez la gestion de la mémoire dans les systèmes monoprogrammés et multiprogrammés.

Dans les systèmes monoprogrammés, un seul processus est chargé en mémoire à la fois, ce qui simplifie la gestion mais limite l'efficacité. Dans les systèmes multiprogrammés, plusieurs processus partagent la mémoire, nécessitant des techniques comme le swapping, la pagination ou la segmentation pour optimiser l'utilisation de la mémoire tout en assurant l'isolation et la protection des processus.

6. Quels sont les avantages et les inconvénients de l'allocation contiguë par partitions fixes et variables ?

- Partitions fixes : Avantages : Simple à implémenter, faible surcharge de gestion. Inconvénients : Fragmentation interne, gaspillage de mémoire si les processus ne remplissent pas totalement les partitions.
- Partitions variables : Avantages : Utilisation plus flexible et efficace de la mémoire. Inconvénients : Fragmentation externe, gestion plus complexe.

7. Décrivez la pagination et son rôle dans la gestion de la mémoire.

La pagination divise la mémoire en blocs de taille fixe appelés pages et la mémoire physique en blocs correspondants appelés cadres. Les pages d'un processus peuvent être chargées dans n'importe quel cadre disponible, permettant une allocation non contiguë, minimisant ainsi la fragmentation externe et rendant la gestion de la mémoire plus flexible.

8. Quelle est la différence entre la segmentation et la pagination ?

La segmentation divise la mémoire en segments logiques basés sur la structure du programme (code, données, pile), tandis que la pagination divise la mémoire en blocs de taille fixe sans se soucier de la structure logique. La segmentation offre une gestion plus naturelle pour les programmes, mais est sujette à la fragmentation externe, contrairement à la pagination.

9. Expliquez le concept de mémoire virtuelle et son importance.

- La mémoire virtuelle permet aux processus de croire qu'ils disposent de plus de mémoire qu'il n'y en a physiquement. Elle utilise le disque dur pour simuler de la mémoire supplémentaire, permettant ainsi l'exécution de programmes nécessitant plus de mémoire que la mémoire vive disponible. Cette technique améliore la multiprogrammation et la gestion des ressources.

10. Correction :

Les partitions fixes entraînent une fragmentation interne et limitent la flexibilité, tandis que les partitions variables réduisent la fragmentation interne mais peuvent entraîner une fragmentation externe. La gestion des partitions variables est plus complexe, mais elles utilisent la mémoire de manière plus efficace en s'adaptant à la taille des processus.

11. Correction :

Un algorithme de pagination typique gère les demandes de pages en vérifiant si la page demandée est en mémoire ; si ce n’est pas le cas, il charge la page depuis le disque et met à jour les tables de pages. La gestion inclut des stratégies comme le FIFO, LRU (Least Recently Used) pour décider quelle page remplacer en cas de manque de cadres disponibles.

12. Correction :

Le swapping libère de la mémoire centrale en déplaçant des processus inactifs sur le disque. Bien que cela augmente la capacité apparente de la mémoire, cela introduit une latence supplémentaire due aux opérations d'E/S, ce qui peut ralentir le système lorsque les processus doivent être réintégrés en mémoire.

13. Correction :

La mémoire virtuelle utilise des tables de pages pour mapper les adresses virtuelles en adresses physiques. Elle permet à chaque processus d'avoir son propre espace d'adressage virtuel, isolant ainsi les processus et améliorant la sécurité. La mémoire virtuelle améliore la gestion des grands programmes et augmente l'efficacité en chargeant en mémoire uniquement les pages nécessaires à l'exécution.

QCM

1. Quel est le rôle principal du gestionnaire de mémoire ?
   - Réponse : b) Gérer l'allocation et la libération de la mémoire.
   - Le gestionnaire de mémoire est responsable de la gestion efficace de l'allocation et de la libération de la mémoire aux processus.

2. Dans quel type de gestion de mémoire utilise-t-on des partitions de taille fixe?
   - Réponse : c) Allocation contiguë par partitions fixes.
   - Ce type de gestion divise la mémoire en partitions de taille fixe, ce qui peut entraîner une fragmentation interne.

3. Le swapping est principalement utilisé pour :
   - Réponse : c) Transférer des processus entre la mémoire centrale et le disque dur.
   - Le swapping déplace temporairement les processus hors de la mémoire centrale pour libérer de l'espace.

4. La segmentation de la mémoire centrale permet de :
   - Réponse : b) Séparer les données et le code dans des segments distincts.
   - La segmentation divise la mémoire en segments logiques distincts correspondant aux différentes parties d'un programme.

5. La mémoire virtuelle permet à un système d'exploitation de :
   - Réponse : a) Accroître la capacité mémoire au-delà de la mémoire physique.
   - La mémoire virtuelle utilise le stockage sur disque pour étendre la capacité apparente de la RAM.

Exercices

Exercice 1 :

1) la chaîne de Références associée est : 1,2,3,1,4,1,2,5,1,2,3,4,5.

2) Politique MFU

Demandes	1	2	3	1	4	1	2	5	1	2	3	4	5
Case 1	**1**	1	1	1	1	1	1	**5**	5	5	5	5	5
Case 2		**2**	2	2	2	2	2	2	**1**	**2**	2	2	2
Case 3			**3**	3	3	3	3	3	3	3	3	3	3
Case 4					**4**	4	4	4	4	4	4	4	4
	D	D	D		D			D	D	D			

7 défauts de pages

3) La méthode qui donne le nombre minimum de défaut de pages est une méthode optimale, elle consiste à choisir une page victime qui ne sera pas référencée dans le futur immédiat.

Demandes	1	2	3	1	4	1	2	5	1	2	3	4	5
Case 1	**1**	1	1	1	1	1	1	1	1	1	1	**4**	4
Case 2		**2**	2	2	2	2	2	2	2	2	2	2	2
Case 3			**3**	3	3	3	3	3	3	3	3	3	3
Case 4					**4**	4	4	**5**	5	5	5	5	5
	D	D	D		D			D				D	

Pour la méthode optimale 6 défauts de pages.

Exercice 02

1

- L'algorithme A1 peut donc être FIFO.
- L'algorithme A2 peut donc être LRU.

2

Avec l'algorithme A_1, les changements sont les suivants, indiqués par *:

*	*	*			*	*	*	*
1	1	1	1	1	2	1	1	1
	5	2	2	2	4	2	4	3
		5	5	5	5	4	5	5

Il y a donc 7 défauts de pages.

Avec l'algorithme A_2, les changements sont les suivants, indiqués par *:

*	*	*			*			*
1	1	1	1	1	1	1	1	1
	5	2	2	2	4	4	4	3
		5	5	5	5	5	5	5

Il y a donc 5 défauts de pages.

Exercice 03

1) Nombre de pages 856 Ko/512 = 1712 pages
Nombre de cases 32 Mo/512 = 65536 cases.

2) Adresse logique : 11 bits pour les numéros de pages et 9 bits pour le déplacement.
Adresse physique 16 bits pour les numéros de cases et 9 bits pour le déplacement.

3) - Adresse de base de la case 15240 alors 15240 * 512 = 7802880
- Déplacement à partir de l'adresse 11301 alors 11301 Mod 512 = 37.
- Adresse physique de l'adresse logique 11301 est : 7802880 + 37 = 7802917

B)

Demandes	0	1	2	1	2	1	2	1	2	3	4	5	6	5	6	7
Case 1	**0**	0	0	0	0	0	0	0	0	**3**	3	3	**6**	6	6	6
Case 2		**1**	1	1	1	1	1	1	1	1	**4**	4	4	4	4	**7**
Case 3			**2**	2	2	2	2	2	2	2	2	**5**	5	5	5	5
	D	D	D							D	D	D	D			D

Le nombre de défauts de page est de 8. le taux par rapport au 16 pages demandées est 8/16 = 50%.

Exercice 04

Dans l'algorithme LRU, on prend la page la moins récemment référencée. Il s'agit donc de prendre la dernière selon le critère de la colonne *Accès*, c'est-à-dire, celle de la case 1 qui a été accédée en 260.

Dans l'algorithme FIFO, on prend la page qui est en mémoire depuis le plus longtemps. Il s'agit donc de suivre le critère de la colonne *Chargement*, c'est-à-dire, celle de la case 2 qui est en mémoire depuis le temps 120.

Dans l'algorithme de la seconde chance, on prend la page qui est en mémoire depuis le plus longtemps, donc selon le critère de la colonne *Chargement*, sauf si son bit accédée est à 1, auquel cas on le remet à 0 et on poursuit le recherche dans l'ordre. Dans l'exemple, la case 2 est la plus ancienne, mais son bit accédée est à 1. La suivante dans l'ordre est la case 0 dont le bit accédée est à 0. C'est donc celle qui est choisie.

Exercice 05 :

Taille Table de Pages = taille d'une entrée de la page * nombre de pages

- Taille d'une entrée : 1(bit de présence) + nbre de bits nécessaires pour adresser les cases
  - Nombre de cases : 32 Mo / 1 Ko = $2^{25} / 2^{10} = 2^{15}$
  - Nbre de bits nécessaire pour adresser les cases : 15 bits
  - ➔ **taille d'une entrée = 16 bits**
- Nombre de pages = 4 Go / 1 Ko = $2^{32}/2^{10} = \mathbf{2^{22}}$
- ➔ **taille de la table de pages = $2^{22} * 16 = 2^{22} * 2^4 = 2^{26} = \sout{64}$ Mo**

Exercice 06 :

1 page = 1Ko = 2^{10} ==➔ offset sur 10 bits

142A

0001	0100	0010	1010
1	4	2	A

N° page = 5 offset = 2A

@ physique : n° cadre = 1 offset = 2A

0000	0100	0010	1010
0	4	2	A

0AF1

0000	1010	1111	0001
0	A	F	1

N° page = 2 offset = 2F1

@ physique : n° cadre = 8 offset = 2F1

0010	0010	1111	0001
2	2	F	1

Exercice 07 :

A- 1. FIFO (First In, First Out) :

- Principe : L'algorithme FIFO remplace la page qui est en mémoire depuis le plus longtemps. Chaque page a une place dans une file d'attente, et lorsque de l'espace est nécessaire, la page à l'avant de la file (celle arrivée en premier) est remplacée.
- Avantage : Simple à implémenter.
- Inconvénient : Ne prend pas en compte la fréquence ou la récence d'utilisation des pages, ce qui peut conduire à remplacer des pages fréquemment utilisées.

2. LRU (Least Recently Used) :

- Principe : L'algorithme LRU remplace la page qui n'a pas été utilisée depuis le plus longtemps. Il utilise des compteurs ou des piles pour suivre l'ordre d'utilisation des pages, favorisant le maintien en mémoire des pages récemment utilisées.
- Avantage : Plus efficace que FIFO pour les systèmes où certaines pages sont utilisées fréquemment.
- Inconvénient : Plus complexe à implémenter que FIFO et nécessite plus de ressources pour suivre l'ordre d'utilisation des pages.

B-

B.1- Si on utilise l'algorithme **FIFO** et sachant que le système alloue à ce programme un espace de 3 pages.

0	0	0	2	2	2	3	3	3	3	3	3
	1	1	1	0	0	0	0	0	4	4	4
		4	4	4	1	1	1	1	1	2	2

➔ défauts de page = 9

B.2- Si on utilise l'algorithme **LRU** et sachant que le système alloue à ce programme un espace de 3 pages.

0	0	0	2	2	2	3	3	3	4	4	4
	1	1	1	0	0	0	0	0	0	2	2
		4	4	4	1	1	1	1	1	1	3

➔ défauts de page = 9

B.3- Si on utilise l'algorithme **Optimal** et sachant que le système alloue à ce programme un espace de 3 pages.

0	0	0	0	0	0	0	0	0	4	4	4
	1	1	1	1	1	1	1	1	1	2	2
		4	2	2	2	3	3	3	3	3	3

➔ défauts de page = 7

B.4- Si on utilise l'algorithme **FIFO** et sachant que le système alloue à ce programme un espace de 4 pages.

0	0	0	0	0	0	3	3	3	3	2	2
	1	1	1	1	1	1	0	0	0	0	3
		4	4	4	4	4	4	1	1	1	1
			2	2	2	2	2	2	4	4	4

➔ défauts de page = 10

B.5- Si on utilise l'algorithme **LRU** et sachant que le système alloue à ce programme un espace de 4 pages.

0	0	0	0	0	0	0	0	0	0	0	3
	1	1	1	1	1	1	1	1	1	1	1
		4	4	4	4	3	3	3	3	2	2
			2	2	2	2	2	2	4	4	4

➔ défauts de page = 8

Exercice 08 :

A.- 1er : FIFO 2ème : LRU

B. – Déterminer dans chacun des cas le nombre de défauts de pages.

1er : NDF = 7 2ème : NDF = 5

Exercice 09 :

FIFO → remplacer case 2 car sa date de chargement est inférieures à celles des autres

LRU → remplacer case1 car sa date d'accès est inférieure à celles des autres

NRU → remplacer case 0 car bit R = 0

Conclusion Générale

Ce polycopié, "Systèmes d'Exploitation 1, Partie 1 (Cours et exercices corrigés)" , représente la première partie d'une série dédiée à l'étude des systèmes d'exploitation. Son objectif est d'accompagner les étudiants dans la découverte des fondements de ce domaine crucial en informatique, à travers une approche théorique appuyée par des exercices pratiques. En parcourant les concepts essentiels, ce document fournit une base solide pour comprendre le rôle des systèmes d'exploitation dans la gestion des ressources matérielles et logicielles d'un ordinateur.

Dans cette première partie, nous avons exploré les concepts fondamentaux qui sous-tendent le fonctionnement des systèmes d'exploitation. Le premier chapitre a posé les bases en présentant l'architecture générale des systèmes informatiques, les modes de fonctionnement, et le rôle des systèmes d'exploitation en tant que gestionnaires de ressources et interfaces entre l'utilisateur et le matériel. Cette introduction a été essentielle pour contextualiser le reste du contenu.

Le deuxième chapitre s'est concentré sur la gestion des processus et l'ordonnancement (scheduling). En étudiant les divers algorithmes d'ordonnancement, tels que FCFS, SJF, Round Robin, et ceux basés sur la priorité, les étudiants ont appris comment les systèmes d'exploitation organisent et optimisent l'exécution des processus pour maximiser l'efficacité et garantir une répartition équitable des ressources.

Le troisième chapitre a abordé la gestion de la mémoire centrale, une composante critique pour le bon fonctionnement des systèmes d'exploitation. En examinant des techniques telles que le swapping, la pagination, la segmentation, et l'utilisation de la mémoire virtuelle, ce chapitre a permis aux étudiants de comprendre comment les systèmes gèrent la mémoire pour assurer des performances optimales et une protection adéquate des processus.

J'espère que ce document a permis d'atteindre plusieurs objectifs pédagogiques clés, notamment :

- Comprendre les rôles et fonctions des systèmes d'exploitation en tant que gestionnaires de ressources et interfaces utilisateur.
- Maîtriser les mécanismes de gestion des processus et des algorithmes d'ordonnancement, qui influencent directement la performance du système.

- Appréhender les techniques de gestion de la mémoire centrale, cruciales pour l'optimisation des ressources matérielles et la stabilité du système.

Ce polycopié marque la première étape d'un parcours plus large qui se poursuit avec une deuxième partie dédiée à l'étude de concepts avancés des systèmes d'exploitation. Dans cette suite, les étudiants aborderont des sujets tels que la gestion de la concurrence avec les sémaphores , les moniteurs , et autres mécanismes de synchronisation essentiels pour assurer la coopération et l'exclusion mutuelle entre les processus. La deuxième partie traitera également des systèmes de fichiers , de la gestion des entrées/sorties , et des concepts de sécurité et de protection , complétant ainsi le panorama des fonctionnalités d'un système d'exploitation moderne.

En guise de conclusion, ce document vise à offrir une compréhension claire et pratique des bases des systèmes d'exploitation. Nous espérons qu'il constituera un support pédagogique précieux pour les étudiants et enseignants, facilitant l'apprentissage et la maîtrise de concepts complexes. La structure en deux parties permet de progresser de manière graduelle, du fondement vers des sujets plus complexes, préparant ainsi les étudiants à des défis techniques avancés. Nous encourageons les enseignants à intégrer ce polycopié dans leurs cours et à en faire un outil d'apprentissage dynamique et interactif. Merci à tous pour votre intérêt et votre engagement, et nous souhaitons que ce travail contribue significativement à votre réussite dans le domaine de l'informatique.

Bibliographie

1. Géronimi, C. (2012). Les systèmes d'exploitation (2e édition). Dunod.
2. Piquet, G. (2015). Systèmes d'exploitation (3e édition). Eyrolles.
3. Bénaben, F., & Pasquali, C. (2011). Systèmes d'exploitation : Concepts et pratiques (2e édition). Hermes Science Publications.
4. Dutot, R. (2013). Introduction aux systèmes d'exploitation : Concepts, principes et outils (2e édition). Pearson.
5. Sansonetti, P., & Rolland, P. (2008). Architecture et fonctionnement des systèmes d'exploitation (3e édition). Lavoisier.
6. Dufour, J., & Gosselin, B. (2016). Systèmes d'exploitation : Théorie et applications (1re édition). Presses Universitaires de France (PUF).
7. Viala, M., & Debaene, J. (2017). Les systèmes d'exploitation en pratique (2e édition). Dunod.
8. Bach, M. J. (2005). La conception des systèmes UNIX (Traduction française). Presses Universitaires de France (PUF).
9. Tamarin, D., & Simon, F. (2014). Les systèmes d'exploitation : Une approche moderne (1re édition). Dunod.
10. Herscovici, M. (2009). Le système d'exploitation : Concepts et pratiques (1re édition). Hermes Science Publications.
11. Bessouf, H. (2023). Systèmes d'exploitation 1 . Université de Mila. Consulté le 01/09/2024. https://elearning.centre-univ-mila.dz/a2024/course/view.php?id=312#section-11
12. Daba, A. (2023). Systèmes d'exploitation 1 . Université de Msila. Consulté le 01/09/2024. https://elearning.univ-msila.dz/moodle/course/view.php?id=9809&lang=fr
13. Saadi, L. (2023). Operating Systems Introduction . Université de Batna. Consulté le 01/09/2024. https://staff.univ-batna2.dz/saadi_leila/classes/operating-systems-introduction-first-year-engineer-20232024
14. https://www.exoco-lmd.com/systemes-dexploitation-i/
15. https://computer-science-dz.wixsite.com/bbami/sd
16. https://www.physiquechimiemathbiologie.com/search/label/Syst%C3%A8mes%20d%27exploitation%201%20L2?m=1
17. https://developpement-informatique.com/article/550/concepts-de-base-de-lordonnancement-du-cpu
18. http://elearning.univ-djelfa.dz/course/view.php?id=933