CHAPITRE I : INTRODUCTION

1.1 QU'EST CE QU'UN SYSTEME D'EXPLOITATION ?

Il existe plusieurs définitions possibles des systèmes d'exploitation.

Un système d'exploitation peut être vu comme un programme agissant en tant qu' intermédiaire entre l'utilisateur et le matériel de l'ordinateur. Le but principal d'un système d'exploitation est de fournir un environnement qui permette à un utilisateur d'exécuter des programmes.

Le système d'exploitation (SE) est donc une partie importante de tout système informatique. En effet, on peut diviser grossièrement un système informatique en trois composants : *le matériel*, *le système d'exploitation* et *les programmes d'application* (figure 1.1).

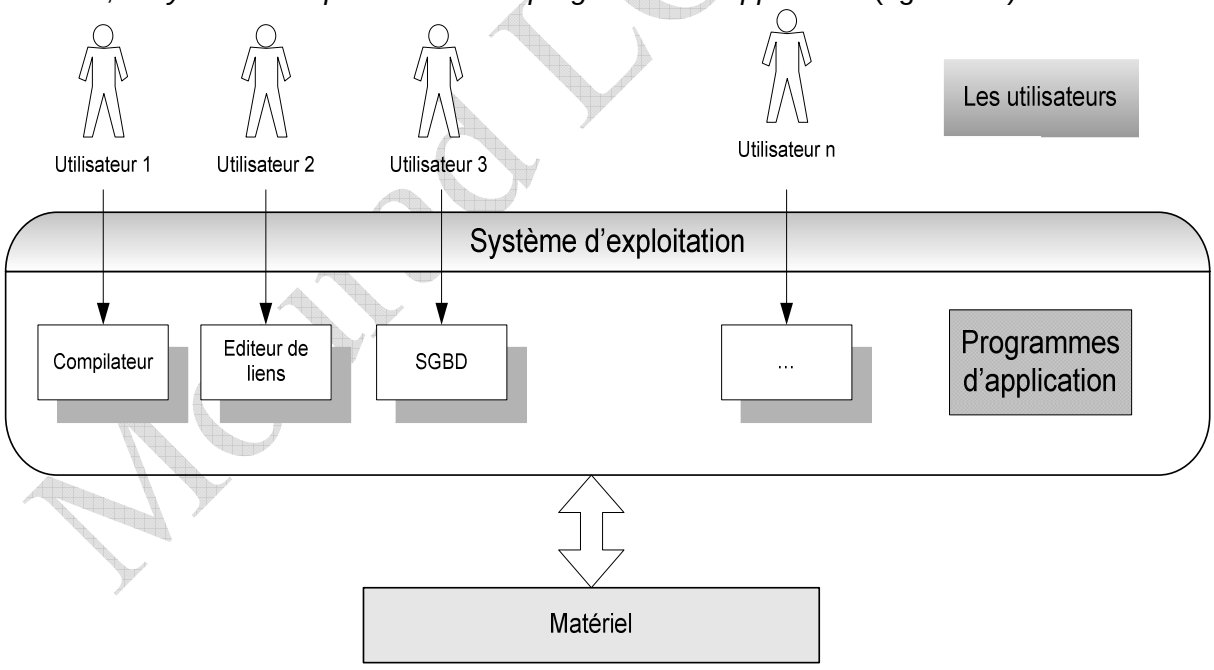

Fig 1.1 Vue abstraite d'un système informatique

Le matériel : fournit les ressources informatiques de base , comme : l'unité centrale, la mémoire, les périphériques d'entrée/sortie, ….

Les programmes d'application : définissent les façons d'utiliser ces ressources pour résoudre les problèmes informatiques des utilisateurs. Exemples : compilateurs, les systèmes de gestion de base de données, les applications de gestion, les logiciels de Bureautique, … etc.

Il peut exister plusieurs utilisateurs (des personnes, des machines, d'autres ordinateurs, …) qui essayent de résoudre des problèmes différents sur le même système informatique. A un instant donné, il peut donc exister plusieurs programmes d'application différents ; le SE doit alors contrôler et coordonner l'utilisation du matériel parmi les divers programmes d'application pour les divers utilisateurs.

D'autre part, un SE peut être considéré comme **un programme d'allocation** de ressources. Un système informatique possède plusieurs ressources (matérielles et logicielles) qui peuvent être requises pour résoudre un problème donné : temps processeur, espace mémoire, espace de stockage des fichiers, périphériques d'entrée/sortie. Le SE agit en gestionnaire de ces ressources et les alloue aux programmes et aux utilisateurs quand cela est nécessaire à leur tâche. Puisqu'il put exister plusieurs requêtes , parfois incompatibles, pour les ressources, le SE doit décider à quelle requête on doit allouer les ressources afin d'exploiter le système informatique de façon efficace et optimale.

Une vue légèrement différente d'un SE se focalise sur le besoin de contrôler les divers périphériques d'Entrée/Sortie et les programmes utilisateurs. Un SE est donc **un programme de contrôle**. Ce dernier vérifie l'exécution des programmes utilisateurs pour éviter des erreurs et une utilisation incorrecte de l'ordinateur.

En conclusion , nous pouvons dire donc que :

Un système d'exploitation = Programme d'allocation + Programme de contrôle.

1.2 HISTORIQUE :

Pour comprendre ce que sont les SE, nous devons tout d'abord comprendre comment ils se sont développés. Cela permettra de voir comment les composants des SE ont évolué comme des solutions naturelles aux problèmes des premiers SE.

1.2.1 Les premiers systèmes :

Les premiers ordinateurs étaient physiquement des machines très volumineuses utilisées à partir d'une console. Le programmeur, qui était aussi l'opérateur du système informatique, écrivait un programme et le faisait ensuite fonctionner directement à partir de la console de l'opérateur. Le programme était d'abord chargé manuellement en mémoire, à partir de bandes perforées, en utilisant des interrupteurs du panneau frontal (une instruction à la fois),. Les touches appropriées étaient pressées pour fixer l'adresse de début et pour commencer l'exécution du programme. Le programmeur/opérateur pouvait surveiller l'exécution du programme par les voyants d'affichage de la console. Si l'on découvrait des erreurs, le programmeur pouvait arrêter le programme, examiner les contenus de la

mémoire et des registres et déboguer le programme directement à partir de la console. Les résultats étaient imprimés ou perforés sur des bandes ou des cartes afin de les imprimer plus tard.

Peu à peu on a développé du matériel et du logiciel supplémentaire : Les lecteurs de cartes, les imprimantes, les bandes magnétiques, … On a conçu des assembleurs, des chargeurs et des éditeurs de liens pour faciliter la tâche de programmation. Des bibliothèques ayant des fonctions communes ont été crées. Les fonctions communes pouvaient donc copiées dans un nouveau programme sans avoir à les écrire à nouveau, ce qui a permis la réutilisabilité des logiciels.

Plus tard sont apparus des compilateurs FORTRAN et COBOL, … etc, ce qui rendait la tâche de la programmation plus facile, mais le fonctionnement de l'ordinateur plus complexe. Par exemple, pour préparer l'exécution d'un programme FORTRAN, le programme devait d'abord charger le compilateur FORTRAN dans l'ordinateur (existant sur une bande). Il y avait donc un temps de préparation considérable pour l'exécution du travail. Chaque travail s'effectuait en étapes séparées : chargement de la bande de compilateur, exécution du compilateur, déchargement de la bande de compilateur, mise en place de la bande de l'assembleur, exécution de l'assembleur, déchargement de la bande de l'assembleur, chargement et exécution du programme projet.

1.2.2 Systèmes simples de traitements par lots :

Le problème des premiers systèmes était le temps de préparation du travail qui était très important. Pendant que les bandes se montaient ou que le programmeur utilisait la console, l'UC restait inactive. Ce qui engendrait des coût s d'indisponibilité très importants.

Pour venir à bout de ce temps inactif, on a développé **l'enchaînement automatique des travaux**. Cette technique a permis de créer les premiers SE rudimentaires. Ce qu'on l'on désirait, c'était une procédure pour transférer automatiquement le contrôle d'un travail à un autre. A cet effet, on crée un programme appelé **moniteur résident**.

Quand l'ordinateur était allumé, le moniteur résident était appelé et il transférait le contrôle à un programme. Quand le programme finissait, il renvoyait le contrôle au moniteur résident , lequel continuait avec le travail suivant. Ainsi, le moniteur résident enchaînait automatiquement d'un programme à un autre.

Mais comment le moniteur résident pouvait-il savoir quel programme exécuter ?. Réponse : Grâce aux cartes de contrôle. En plus du programme et des données nécessaires au travail, le programmeur introduisait des cartes spéciales (les cartes de contrôle) avec des directives pour le moniteur résident qui lui indiquaient les programmes à exécuter.

Par exemple, un programme utilisateur normal pouvait demander d'exécuter l'un des trois programmes : le compilateur FORTRAN (FTN), l'assembleur (ASM) ou le programme de l'utilisateur (RUN). On peut utiliser une carte de contrôle pour chacun d'eux (figure 1.2) .

$FTN : Exécuter le compilateur FORTRAN.
$ASM : Exécuter l'assembleur
$RUN : Exécuter le programme de l'utilisateur.

On peut utiliser deux cartes de contrôle supplémentaires pour définir les limites de chaque travail :

$JOB : Première carte de travail.
$END : Dernière carte de travail.

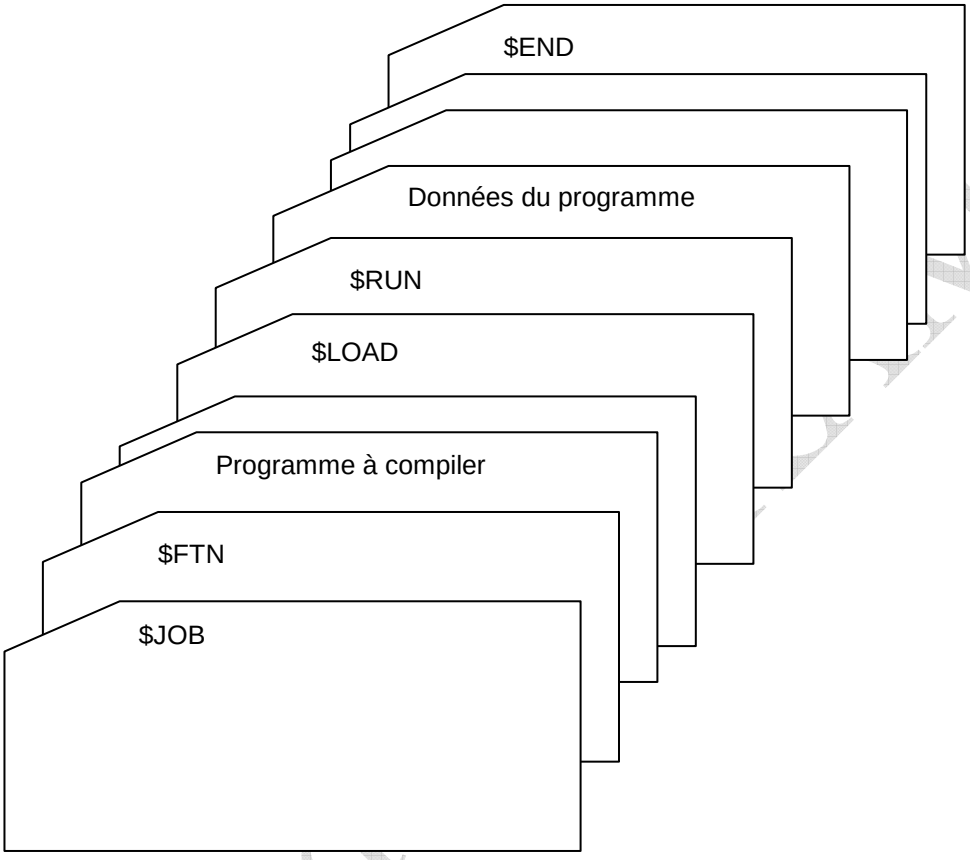

Figure 1.2 Système de traitement par lot

Un moniteur d'enchaînement possède ainsi plusieurs parties identifiables. L'une d'entre elles est **l'interpréteur de cartes de contrôle** qui a la responsabilité de lire et d'exécuter les instructions sur les cartes. Cet interpréteur appelle par intervalle **un chargeur** pour charger les programmes d'application en mémoire. Le chargeur fait appel à son tour à des opérations d'Entrées/Sorties. Le moniteur possède donc un ensemble de drivers pour les périphériques d'entrée/sortie.

Critique des systèmes de traitement par lot :

- Manque d'interaction entre l'utilisateur et le travail pendant l'exécution.
- Lenteur des opérations d'E/S

En effet, même avec l'enchaînement automatique des travaux, l'UC est souvent inactive. Le problème réside dans la vitesse des périphériques d'E/S mécaniques qui sont intrinsèquement plus lents que les dispositifs électroniques.

1.2.3 Systèmes multiprogrammés :

Un système multiprogrammé maintient plusieurs travaux en mémoire à la fois (voir figure).

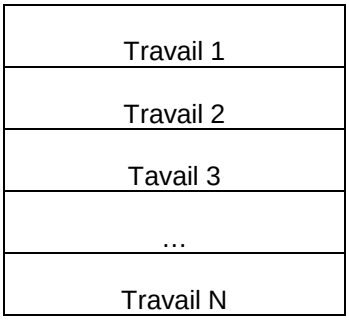

Figure 1.6 Schéma mémoire pour un système de multiprogrammation

Le SE choisit et commence à exécuter un des travaux en mémoire. Par la suite, le travail risque d'attendre une tâche, comme le montage d'une bande, une donnée frappée au clavier ou l'achèvement d'une opération d'E/S. Dans un système n'étant pas multiprogrammé, le SE commute simplement sur un autre travail et l'exécute. Quand ce travail a besoin d'attendre, l'UC est connectée sur un autre travail et ainsi de suite. Ainsi tant qu'il existera un travail à exécuter, l'UC ne sera jamais inactive.

1.2.4 Système en temps partagé :

Le temps partagé est une extension logique de la multiprogrammation. Plusieurs travaux sont exécutés par l'UC qui les commute entre eux mais les commutations sont si fréquentes que les utilisateurs peuvent dialoguer avec chaque programme pendant son exécution.

Un système informatique interactif permet la communication en ligne entre l'utilisateur et le système. L'utilisateur donne des instructions au système d'exploitation ou directement à un programme et reçoit une réponse immédiate.

Les systèmes en temps partagé ont été développés pour fournir une utilisation interactive du système informatique. Un SE en temps partagé utilise le scheduling de l'UC et la multiprogrammation pour fournir à chaque utilisateur une petite portion de temps.

Un système d'exploitation en temps partagé permet aux différents utilisateurs de partager l'ordinateur simultanément. Comme le système commute très vite d'un utilisateur à un autre, chaque utilisateur a l'impression qu'il possède son propre ordinateur, alors qu'en réalité l'ordinateur est partagé entre plusieurs utilisateurs.

Exemple : l'OS 360/IBM.

1.2.5 Les systèmes des ordinateurs personnels :

Comme le coût du matériel a baissé, il est devenu possible d'avoir un système informatique dédié à un seul utilisateur. Ces types de systèmes informatiques sont habituellement appelés ordinateurs personnels ou PC.

Les PC sont apparus dans les années 70. C'étaient des machines bon marché mais dont les SE n'étaient ni multiutilisateurs, ni multitâches.
Exemples de systèmes : MS-DOS de Microsoft, Macintosh d'Apple.

1.2.6 Les systèmes parallèles :

La plupart des systèmes actuels sont des systèmes monoprocesseurs, c'est à dire qu'ils possèdent une seule UC principale. Cependant, ils se dessine une tendance en faveur des systèmes multiprocesseurs. De tels systèmes possèdent plus d'un processeur en étroite communication partageant le bus de l'ordinateur, l'horloge et quelque fois la mémoire et les périphériques.

Il existe plusieurs raisons de créer de tels systèmes :

- Augmenter la capacité de traitement du système.
- Possibilité d'économiser du coût par rapport à plusieurs systèmes mono-processeurs, car les processeurs peuvent partager les périphériques, les boîtiers et les alimentations électriques. Si plusieurs programmes doivent exploiter le même ensemble de données, il est moins cher de stocker ces données sur un seul disque et de les faire partager par tous les processeurs plutôt que d'avoir plusieurs ordinateurs avec des disques locaux et de nombreuses copies de données.
- Les systèmes multiprocesseurs présentent un autre avantage, ils augmentent la ***fiabilité***. En effet une panne d'un processeur n'arrêtera pas le système.

1.2.7 Systèmes répartis :

Une tendance récente dans les systèmes informatiques consiste à répartir le calcul entre plusieurs processeurs. A l'opposé des systèmes centralisés, dans les systèmes répartis on ne partage pas de mémoire ou d'horloge. Au lieu de cela, chaque processeur possède sa propre mémoire locale. Les processeurs communiquent entre eux à travers des lignes de communication, comme des bus rapides ou des lignes téléphoniques. Les systèmes sont habituellement appelés répartis (ou faiblement couplés).

Dans un système réparti, les processeurs peuvent varier en taille et en fonction. Ils peuvent inclure des petits microprocesseurs, des stations de travail, des micro-ordinateurs et des grand systèmes informatiques à usage général. Ces processeurs sont appelés : site, nœud, poste, …selon le contexte.

Il existe de nombreuses raisons de construire des systèmes répartis :

- *Partage de ressources* : Si plusieurs sites différents (avec des possibilités différentes), sont connectés entre eux, un utilisateur dans un site doit être capable d'utiliser les ressources disponibles dans un autre site. Par exemple, un utilisateur dans un site A peut imprimer sur une imprimante à laser disponible sur un site B. De même qu'un utilisateur dans B peut accéder à un fichier résident dans A.
- *Accélération du cal*cul : Si l'on peut subdiviser un traitement particulier en plusieurs sous-traitements pouvant s'exécuter en concurrence, un système réparti doit nous permettre de distribuer le traitement entre les différents sites.

- *Fiabilité* : si un site tombe en panne dans un système réparti, les sites restant peuvent continuer à fonctionner.
- *Communication* : les utilisateurs peuvent déclencher des transferts de fichiers ou communiquer entre eux à travers le courrier électronique, par exemple.

1.2.8 Les systèmes à temps réel :

Les systèmes à temps réel sont une autre forme de SE spécialisés. Un système à temps réel est utilisé quand il existe des exigences impérieuses de temps de réponse pour le fonctionnement d'un processeur ou pour le jeu de données. Il est souvent employé comme dispositif de contrôle dans une application dédiée. Les capteurs amènent des données à l'ordinateur. Celui-ci doit analyser les données et éventuellement régler les contrôle pour modifier les entrées des capteurs.

Exemple de système à temps réel : systèmes contrôlant les expérimentations scientifiques, les systèmes d'imagerie médicale, les systèmes de contrôle industriel.

Un système d'exploitation à temps réel possède des contraintes de temps fixes. Le traitement doit être effectué dans les contraintes définies.

1.3 COMPOSANTS D'UN SYSTEME D'EXPLOITATION :

Pour créer un système aussi grand et complexe qu'un système d'exploitation, il est nécessaire de le découper en pièces plus petites. Chacune d'entre elles devrait être une portion bien délimitée du système, avec des entrées, des sorties et des fonctions soigneusement définies. Bien entendu les systèmes d'exploitation ne possèdent pas tous la même structure. Cependant, plusieurs systèmes modernes possèdent les composants suivants :

- Gestion de processeurs.
- Gestion mémoire principale.
- Gestion mémoire auxiliaire.
- Gestion du système d'entrée/sortie.
- Gestion des fichiers.
- Système de protection.
- Gestion de réseaux.
- Système interpréteur de commande.

1.4 STRUCTURE EN COUCHES D'UN SYSTEME D'EXPLOITATION :

La conception d'un SE est basée sur une structure à couches. Elle consiste à découper le SE en un certain nombre de couches (niveaux), chacune d'entre elles étant construite au dessus des couches inférieures.

La couche inférieure (couche 0) est le matériel ; la couche supérieure (couche N) est l'interface utilisateur.

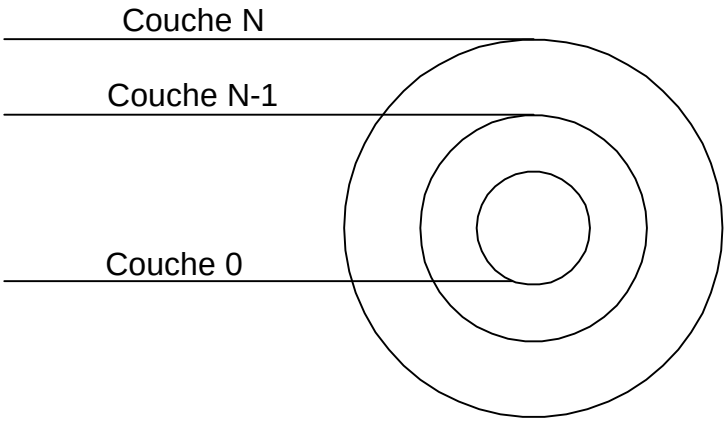

Figure 1.7 Structures en couches d'un système d'exploitation.

Ainsi une couche i peut utiliser tous les objets (variables et routines) des couches inférieure j Le principal avantage de l'approche en couches est la modularité. Les couches sont sélectionnées de telle sorte que chacune utilise seulement les fonctions (opérations) et services des couches de niveau inférieur. Cette approche simplifie le débogage et la vérification du système.

Définition d'un appel système :

Un appel système (ou appel superviseur) ou appel noyau est un appel d'une des fonctions du noyau du système d'exploitation (implémenté au niveau d'une des couches).

Les appels systèmes peuvent être grossièrement classés en cinq catégories principales :

- Contrôle de processus (arrêter, charger, exécuter un processus, …).
- Manipulation de fichiers (ouvrir, fermer, …).
- Manipulation de périphériques (demander, libérer, …).
- Manipulation de l'information (modifier l'heure, …).
- Communications (créer , supprimer des connexions, …)..