Chapitre 1 GENERALITES

Les logiciels peuvent être classés en deux catégories :

- les programmes d'application des utilisateurs
- les programmes système qui permettent le fonctionnement de l'ordinateur. Parmi ceux-ci, le système d'exploitation (SE dans la suite).

Le SE soustrait le matériel au regard du programmeur et offre une présentation agréable des fichiers. Un SE a ainsi deux objectifs principaux :

- **présentation** : Il propose à l'utilisateur une abstraction plus simple et plus agréable que le matériel : une **machine virtuelle**

- **gestion** : il ordonne et contrôle l'allocation des processeurs, des mémoires, des icônes et fenêtres, des périphériques, des réseaux entre les programmes qui les utilisent. Il assiste les programmes utilisateurs. Il protège les utilisateurs dans le cas d'usage partagé.

1. HISTORIQUE

Les premiers ordinateurs étaient mis à la disposition d'un programmeur selon un calendrier de réservation : un usager avec un travail unique utilisait seul la machine à un moment donné. Puis vint l'époque du **traitement par lots** (batch) : enchaînement, sous le contrôle d'un moniteur, d'une suite de travaux avec leurs données, confiés à l'équipe d'exploitation de la machine (inconvénient : temps d'attente des résultats pour chaque utilisateur).

Cette pratique a nécessité trois innovations :

- le contrôle des E/S et leur protection pour éviter le blocage d'un lot
- un mécanisme de comptage de temps et de déroutement autoritaire des programmes pour éviter le blocage d'un lot à cause d'une séquence trop longue. Ce furent les première interruptions
- les premiers langages de commande (JCL) sous forme de cartes à contenu particulier introduites dans le paquet ($JOB, $LOAD, $RUN, etc...)

Historiquement, on peut dire que les SE sont vraiment nés avec les ordinateurs de la 3ème génération (ordinateurs à circuits intégrés apparus après 1965). Le premier SE digne de ce nom est l'OS/360, celui des IBM 360, famille unique de machines compatibles entre elles, de puissances et de configurations différentes. Bien que son extrême complexité (due à l'erreur de couvrir toute la gamme 360) n'ait jamais permis d'en réduire le nombre de bogues, il apportait deux concepts nouveaux :

- **la multiprogrammation** : partitionnement de la mémoire permettant au processeur d'accueillir une tâche dans chaque partie et donc d'être utilisé plus efficacement par rapport aux temps d'attente introduits par les périphériques (le processeur est ré-alloué)

- **les E/S tamponnées** : adjonction à l'UC d'un processeur autonome capable de gérer en parallèle les E/S ou canal ou unité d'échange. Cela nécessite une politique de partage du bus ou d'autres mécanismes (vol de cycle, DMA).

Au MIT, F.J. CORBATO et son équipe ont réalisé dès 1962, sur un IBM 7094 modifié, le premier SE expérimental à **temps partagé** (mode interactif entre plusieurs utilisateurs

simultanés), baptisé CTSS. Une version commerciale, nommée MULTICS (MULTiplexed Information and Computing Service), a été ensuite étudiée par cette équipe, les Bell Laboratories et General Electric. Les difficultés ont fait que MULTICS n'a jamais dépassé le stade expérimental sur une douzaine de sites entre 1965 et 1974. Mais il a permis de définir des concepts théoriques importants pour la suite.

La technologie à base de circuits intégrés de la 3ème génération d'ordinateurs a permis l'apparition des mini-ordinateurs et leur diffusion massive (précédant celle des micro-ordinateurs). En 1968, l'un des auteurs de MULTICS, Ken Thompson a effectué une adaptation de MULTICS mono-utilisateur sur un mini-ordinateur PDP-11 de DEC inutilisé dans son Laboratoire des Bell Laboratories. Son collègue Brian Kernighan la nomma UNICS (Uniplexed - *à l'opposé de Multiplexed* - Information and Computer Service), qui devint ensuite UNIX . En 1972, son collègue Dennis Ritchie traduisit UNIX en C, langage qu'il venait de mettre au point avec Kernighan..... L'ère des grands SE avait commencé.

Avec la grande diffusion des micro-ordinateurs, l'évolution des performances des réseaux de télécommunications, deux nouvelles catégories de SE sont apparus :

- **les SE en réseaux** : ils permettent à partir d'une machine de se connecter sur une machine distante, de transférer des données. Mais chaque machine dispose de son propre SE

- **les SE distribués ou répartis** : l'utilisateur ne sait pas où sont physiquement ses données, ni où s'exécute son programme. Le SE gère l'ensemble des machines connectées. Le système informatique apparaît comme un mono-processeur.

2. ELEMENTS DE BASE D'UN SYSTEME D'EXPLOITATION

Les principales fonctions assurées par un SE sont les suivantes :

- gestion de la mémoire principale et des mémoires secondaires,
- exécution des E/S à faible débit (terminaux, imprimantes) ou haut débit (disques, bandes),
- multiprogrammation, temps partagé, parallélisme : interruption, ordonnancement, répartition en mémoire, partage des données
- lancement des outils du système (compilateurs, environnement utilisateur,...) et des outils pour l'administrateur du système (création de points d'entrée, modification de privilèges,...),
- lancement des travaux,
- protection, sécurité ; facturation des services,
- réseaux

L'interface entre un SE et les programmes utilisateurs est constituée d'un ensemble d'instructions étendues, spécifiques d'un SE, ou appels système. Généralement, les appels système concernent soit les processus, soit le système de gestion de fichiers (SGF).

2.1 Les processus

Un processus est un programme qui s'exécute, ainsi que ses données, sa pile, son compteur ordinal, son pointeur de pile et les autres contenus de registres nécessaires à son exécution.

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

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

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

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

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

A chaque instant, un processus est caractérisé par son **état courant** : c'est l'ensemble des informations nécessaires à la poursuite de son exécution (valeur du compteur ordinal, contenu des différents registres, informations sur l'utilisation des ressources). A cet effet, à tout processus, on associe un **bloc de contrôle de processus** (BCP). Il comprend généralement :

- une copie du PSW au moment de la dernière interruption du processus
- l'état du processus : prêt à être exécuté, en attente, suspendu, ...
- des informations sur les ressources utilisées
    - mémoire principale
    - temps d'exécution
    - périphériques d'E/S en attente
    - files d'attente dans lesquelles le processus est inclus, etc...
- et toutes les informations nécessaires pour assurer la reprise du processus en cas d'interruption

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

2.2 Les interruptions

Une interruption est une commutation du mot d'état provoquée par un signal généré par le matériel. Ce signal est la conséquence d'un événement interne au processus, résultant de son exécution, ou bien extérieur et indépendant de son exécution. Le signal va modifier la valeur d'un indicateur qui est consulté par le SE. Celui-ci est ainsi informé de l'arrivée de l'interruption et de son origine. A chaque cause d'interruption est associé un **niveau** d'interruption. On distingue au moins 3 niveaux d'interruption :

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

UNIX admet 6 niveaux d'interruption : interruption horloge, interruption disque, interruption console, interruption d'un autre périphérique, appel système, autre interruption.

Le chargement d'un nouveau mot d'état provoque l'exécution d'un autre processus, appelé le **traitant** de l'interruption. Le traitant réalise la sauvegarde du contexte du processus interrompu (compteur ordinal, registres, indicateurs,....). Puis le traitant accomplit les opérations liées à l'interruption concernée et restaure le contexte et donne un nouveau contenu au mot d'état : c'est l'**acquittement** de l'interruption.

Généralement un numéro de priorité est affecté à un niveau d'interruption pour déterminer l'ordre de traitement lorsque plusieurs interruptions sont positionnées. Il est important de pouvoir retarder, voire annuler la prise en compte d'un signal d'interruption. Les techniques que l'on utilise sont le **masquage** et le **désarmement** des niveaux d'interruption :

- le masquage d'un niveau retarde la prise en compte des interruptions de ce niveau. Pour cela, on positionne un indicateur spécifique dans le mot d'état du processeur. Puisqu'une interruption modifie le mot d'état, on peut masquer les interruptions d'autres niveaux pendant l'exécution du traitant d'un niveau. Lorsque le traitant se termine par un acquittement, on peut alors démasquer des niveaux qui avaient été précédemment masqués. Les interruptions intervenues pendant l'exécution du traitant peuvent alors être prises en compte

- le désarmement d'un niveau permet de supprimer la prise en compte de ce niveau par action sur le mot d'état. Pour réactiver la prise en compte, on réarme le niveau. Il est évident qu'un déroutement ne peut être masqué; il peut toutefois être désarmé.

2.3 Les ressources

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

Un processus demande au SE l'accès à une ressource. Certaines demandes sont implicites ou permanentes (la ressource processeur). Le SE **alloue** une ressource à un processus. Une fois une ressource allouée, le processus a le droit de l'utiliser jusqu'à ce qu'il **libère** la ressource ou jusqu'à ce que le SE reprenne la ressource (on parle en ce cas de **ressource préemptible**, de préemption).

On dit qu'une ressource est en mode d'**accès exclusif** si elle ne peut être allouée à plus d'un processus à la fois. Sinon, on parle de mode d'**accès partagé**. Un processus possédant une ressource peut dans certains cas en modifier le mode d'accès.

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

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

2.4 L'ordonnancement

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

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

2.5 Le système de gestion de fichiers

Une des fonctions d'un SE est de masquer les spécificités des disques et des autres périphériques d'E/S et d'offrir au programmeur un modèle de manipulation des fichiers agréable et indépendant du matériel utilisé.

Les appels système permettent de créer des fichiers, de les supprimer, de lire et d'écrire dans un fichier. Il faut également ouvrir un fichier avant de l'utiliser, le fermer ultérieurement. Les fichiers sont regroupés en répertoires arborescents; ils sont accessibles en énonçant leur chemin d'accès (chemin d'accès absolu à partir de la racine ou bien chemin d'accès relatif dans le cadre du répertoire de travail courant).

Le SE gère également la protection des fichiers.

3. STRUCTURE D'UN SYSTEME D'EXPLOITATION

On peut distinguer quatre grandes catégories de SE.

3.1 Les systèmes monolithiques

Le SE est un ensemble de procédures, chacune pouvant appeler toute autre à tout instant. Pour effectuer un appel système, on dépose dans un registre les paramètres de l'appel et on exécute une instruction spéciale appelée **appel superviseur ou appel noyau**. Son exécution commute la machine du mode utilisateur au **mode superviseur ou noyau** et transfère le contrôle au SE. Le SE analyse les paramètres déposés dans le registre mentionné plus haut et en déduit la procédure à activer pour satisfaire la requête. A la fin de l'exécution de la procédure système, le SE rend le contrôle au programme appelant.

Généralement, un tel SE est organisé en 3 couches :

- une procédure principale dans la couche supérieure, qui identifie la procédure de service requise

- des procédures de service dans la couche inférieure à la précédente qui exécutent les appels système

- des procédures utilitaires dans la couche basse qui assistent les procédures système. Une procédure utilitaire peut être appelée par plusieurs procédures systèmes.

3.2 Les systèmes en couches

On peut généraliser la conception précédente et concevoir un SE composé de plusieurs couches spécialisées, chaque couche ne pouvant être appelée que par des procédures qui lui sont immédiatement inférieures. Citons par exemple le premier SE de cette nature proposé par Dijkstra en 1968 :

- couche 0 : allocation du processeur par commutation de temps entre les processus, soit à la suite d'expiration de délais, soit à la suite d'interruption (multiprogrammation de base du processeur)

- couche 1 : gestion de la mémoire, allocation d'espace mémoire pour les processus (pagination)

- couche 2 : communication entre les processus et les terminaux

- couche 3 : gestion des E/S (échanges d'information avec des mémoires tampons, c'est à dire avec des périphériques abstraits, dégagés des spécificités matérielles)

- couche 4 : programmes utilisateurs

3.3 Les machines virtuelles

Une des premiers SE à gérer le concept de machine virtuelle a été l'adaptation temps partagé de l'OS/360 d'IBM, proposé vers 1968 sous le nom de CP/CMS, puis sous le nom de VM/370 en 1979.

Le cœur du SE, appelé moniteur de machine virtuelle ou VM/370, s'exécute à même le matériel et fournit à la couche supérieure plusieurs machines virtuelles. Ces machines virtuelles sont des copies conformes de la machine réelle avec ses interruptions, ses modes noyau/utilisateur, etc...

Chaque machine virtuelle peut exécuter son propre SE. Lorsqu'une machine virtuelle exécute en mode interactif un appel système, l'appel est analysé par le moniteur temps partagé de cette machine, CMS. Toute instruction d'E/S, toute instruction d'accès mémoire est convertie par VM/370 qui les exécute dans sa simulation du matériel. La séparation complète de la multiprogrammation et de la machine étendue rend les éléments du SE plus simples et plus souples. VM/370 a gagné en simplicité en déplaçant une grande partie du code d'un SE dans le moniteur CMS.

3.4 L'architecture client/serveur

Cette tendance s'est accentuée dans les SE contemporains en tentant de réduire le SE à un noyau minimal. Une des formes les plus accentuées de cette évolution est l'architecture client/serveur.

La plupart des fonctionnalités d'un SE sont reportées dans des processus utilisateurs. Pour demander un service comme la lecture d'un bloc de fichier, le processus utilisateur ou processus client envoie une requête à un processus serveur qui effectue le travail et envoie une réponse. Le noyau ne gère que la communication entre les clients et les serveurs. Cependant, le noyau est souvent obligé de gérer certains processus serveurs critiques comme les pilotes de périphériques qui adressent directement le matériel.

La décomposition du SE en modules très spécialisés le rend facile à modifier. Les serveurs s'exécutent comme des processus en mode utilisateur et non pas en mode noyau. Comme ils n'accèdent donc pas directement au matériel, une erreur n'affecte que le serveur et pas l'ensemble de la machine.

En outre, ce modèle est bien adapté aux systèmes distribués. Un client n'a pas besoin de savoir si le SE fait exécuter sa requête par un serveur de sa propre machine ou celui d'une machine distante.