Conservatoire National des Arts et Métiers
FOAD Ile De France

Les Systèmes d'Exploitation

Exercices et Travaux Pratiques
20 Juillet 2020

Sommaire

Le système d'exploitation réalise donc une couche logicielle placée entre la machine matérielle et les applications. Le système d'exploitation peut être découpé en plusieurs grandes fonctionnalités. Dans une première approche, ces fonctionnalités qui seront étudiées plus en détail dans ce cour sont :

- la fonctionnalité de gestion du processeur : le système doit gérer l'allocation du processeur aux différents programmes pouvant s'exécuter. Cette allocation se fait par le biais d'un **algorithme d'ordonnancement** qui planifie l'exécution des programmes. Dans ce cadre, une exécution de programme est appelée **processus** ;
- la fonctionnalité de gestion des objets externes : comme la mémoire centrale est une mémoire volatile, toutes les données devant être conservées au-delà de l'arrêt de la machine doivent être stockées sur une mémoire de masse non volatile (disque dur, Clé USB, CD-Rom...). La gestion de l'allocation des mémoires de masse ainsi que l'accès aux données qui y sont stockées s'appuient sur la notion de **fichiers** et de **système de gestion de fichiers** ;
- la fonctionnalité de gestion des entrées-sorties : le système doit gérer l'accès aux périphériques. Il doit faire la liaison entre les appels de haut niveau des programmes utilisateurs (exemple : fgetc()) et les opérations de bas niveau de l'unité d'échange responsable du périphérique (unité d'échange clavier) : c'est le **pilote d'entrées-sorties** (***driver***) qui assure ce travail ;
- la fonctionnalité de gestion de la mémoire : le système doit gérer l'allocation de la mémoire centrale entre les différents programmes pouvant s'exécuter, c'est-à-dire qu'il doit trouver une place libre suffisante en mémoire centrale pour que le chargeur puisse y placer un programme à exécuter, en s'appuyant sur les mécanismes matériels sous-jacents de **segmentation** et de **pagination**. Comme la mémoire physique est souvent trop petite pour contenir la totalité des programmes, la gestion de la mémoire se fait selon le principe de la mémoire virtuelle : à un instant donné, seules sont chargées en mémoire centrale les parties de code et données utiles à l'exécution, les autres étant temporairement stockées sur disque ;
- la fonctionnalité de gestion de la concurrence : comme plusieurs programmes coexistent en mémoire centrale, ceux-ci peuvent vouloir communiquer pour échanger des données. Par ailleurs, il faut synchroniser l'accès aux données partagées afin de maintenir leur cohérence. Le système d'exploitation offre des outils de communication et de synchronisation entre processus ;
- la fonctionnalité de gestion de la protection : le système doit fournir des mécanismes garantissant que ses propres ressources (processeur, mémoire, fichiers) ne peuvent être utilisées que par les programmes auxquels les droits nécessaires ont été accordés. Ii faut notamment protéger le système et la machine des programmes utilisateurs (**mode « exécution utilisateur »** et **mode « superviseur »**) ;
- la fonctionnalité d'accès au réseau : des exécutions de programmes placées sur des machines physiques distinctes doivent pouvoir échanger des données. Le système d'exploitation fournit des outils permettant aux applications distantes de dialoguer au travers une couche de protocoles réseau telle que TCP/IP.

Les fonctionnalités du système d'exploitation utilisent les mécanismes offerts par le matériel de la machine physique pour réaliser leurs opérations. Notamment, le système d'exploitation s'interface avec la couche matérielle, par le biais du mécanisme des interruptions (Cf.

chapitre précédent), qui lui permet de prendre connaissance des événements survenant au niveau matériel.

Par ailleurs, le système d'exploitation s'interface avec les applications du niveau utilisateur par le biais des fonctions prédéfinies que chacune de ses fonctionnalités offre. Ces fonctions que l'on qualifie de **routines systèmes** constituent les points d'entrées des fonctionnalités du système d'exploitation. Il est possible de les appeler depuis les applications de niveau utilisateur. Ces appels peuvent se faire à deux niveaux :

- dans le code d'un programme utilisateur à l'aide d'un **appel système**, qui n'est autre qu'une forme d'appel de procédure amenant à l'exécution d'une routine système ;
- depuis le « prompt » de l'interpréteur de commandes, à l'aide d'une commande. L'interpréteur de commandes est un outil de niveau utilisateur qui accepte les commandes de l'utilisateur, les analyse et lance l'exécution de la routine système associée ;
- plus récemment, dans les systèmes d'exploitations graphique, en utilisant les mécanismes d'icônes (double clic, menu contextuel, clisser/déposer, etc.).

Complément de cours : Différents types d'OS

Les systèmes d'exploitation « ayant à gérer plusieurs tâches » peuvent être classés selon différents types :

- les systèmes à *traitements par lots* ;
- les systèmes *multi-utilisateurs interactifs* ;
- les systèmes *temps-reels*.

Les systèmes à traitements par lots

Les systèmes à traitement par lots (ou systèmes *batch*) constituent en quelque sorte les ancêtres de tous les systèmes d'exploitation. Ils sont nés de l'introduction sur les toutes premières machines de deux programmes permettant une exploitation plus rapide et plus rentable du processeur, en vue d'automatiser les tâches de préparation des travaux à exécuter. Ces deux programmes sont :

- le chargeur dont le rôle a été initialement de charger automatiquement les programmes dans la mémoire centrale de la machine depuis les cartes perforées ou le dérouleur de bandes ;
- le moniteur d'enchaînement de traitements, dont le rôle a été de permettre l'enchaînement automatique des travaux soumis en lieu et place de l'opérateur de la machine.

Le principe du traitement par lots s'appuie sur la composition de lots de travaux ayant des caractéristiques ou des besoins communs, la formation de ces lots visant à réduire les temps d'attente du processeur en faisant exécuter les uns à la suite des autres ou ensemble, des travaux nécessitant les mêmes ressources.

La caractéristique principale d'un système de traitement par lots est qu'il n'y a pas d'interaction possible entre l'utilisateur et la machine durant l'exécution du programme soumis. Le programme est soumis avec ses données d'entrées et l'utilisateur récupère les résultats de l'exécution ultérieurement, soit dans un fichier, soit sous forme d'une impression. C'est le mode de fonctionnement typique des anciens MainFrame.

Les systèmes interactifs

La particularité d'un système d'exploitation interactif est que l'utilisateur de la machine peut interagir avec l'exécution de son programme. Typiquement, l'utilisateur lance l'exécution de son programme et attend, derrière le clavier et l'écran, le résultat de celle-ci. S'il s'aperçoit que l'exécution n'est pas conforme à son espérance, il peut immédiatement agir pour arrêter celle-ci, et analyser les raisons de l'échec.

Puisque l'utilisateur attend derrière son clavier et son écran et que, par nature, l'utilisateur de la machine est un être impatient, le but principal poursuivi par les systèmes interactifs va être d'offrir pour chaque exécution le plus petit temps de réponse possible. Pour parvenir à ce but, la plupart des systèmes interactifs travaillent en temps partagé (impression de multitâche donné en exécutant rapidement de courts fragments de plusieurs programmes les uns à la suite des autres).

En effet, un système en temps partagé permet aux différents utilisateurs de partager l'ordinateur simultanément, tout en ayant par ailleurs la sensation d'être seul à utiliser la machine. Ce principe repose notamment sur un partage de l'utilisation du processeur par les

différents programmes des différents utilisateurs. Chaque programme occupe à tour de rôle le processeur pour un court laps de temps (le quantum) et les exécutions se succèdent suffisamment rapidement sur le processeur pour que l'utilisateur ait l'impression que son travail dispose seul du processeur.

Les systèmes « temps-réel »

Les systèmes temps réel sont des systèmes liés au contrôle de procédé pour lesquels la caractéristique primordiale est que les exécutions de programmes sont soumises à des contraintes temporelles, c'est-à-dire qu'une exécution de programme est notamment qualifiée par une date butoir de fin d'exécution, appelée « **échéance** », au-delà de laquelle les résultats de l'exécution ne sont plus valides.

Exemple : programme de contrôle d'un automate d'une chaîne de production, pilotage d'un missile, etc.

Pour garantir le respect de limites ou contraintes temporelles, il est nécessaire que :

- les différents services et algorithmes utilisés s'exécutent en temps borné. Un système d'exploitation temps réel doit ainsi être conçu de manière à ce que les services qu'il propose (accès hardware, etc.) répondent en un temps borné ;
- les différents enchaînements possibles des traitements garantissent que chacun de ceux-ci ne dépassent pas leurs limites temporelles. Ceci est vérifié à l'aide d'un test appelé « *test d'acceptabilité* ».

On distingue deux types de systèmes « temps réel » :

- le temps réel *strict* (ou dur, de l'anglais *hard real-time*) : il ne tolère aucun dépassement de ces contraintes, ce qui est souvent le cas lorsque de tels dépassements peuvent conduire à des situations critiques, voire catastrophiques : pilote automatique d'avion, système de surveillance de centrale nucléaire, etc. ;
- le temps réel *souple* (ou mou, de l'anglais *soft real-time*) : à l'inverse, le temps réel souple s'accommode de dépassements des contraintes temporelles dans certaines limites au-delà desquelles le système devient inutilisable : visioconférence, jeux en réseau, etc.

On peut ainsi considérer qu'un système temps réel strict doit respecter des limites temporelles données même dans la pire des situations d'exécution possibles. En revanche un système temps réel souple doit respecter ses limites pour une moyenne de ses exécutions. On tolère un dépassement exceptionnel, qui sera peut-être rattrapé à l'exécution suivante.

TP 1 : Instructions Fondamentales des OS

Les ordinateurs et la programmation ont été inventés par des mathématiciens. La programmation et le pilotage des ordinateurs c'est de la manipulation de variables numériques, c'est du pur calcul.

Manipulation des variables numériques

Pour le calcul mathématique on a besoin :

- Des 4 opérations mathématiques : + - x /
- Du débranchement conditionnel (test)
- Des lectures /écritures de données en mémoire (données de calcul)

Ces fonctionnalités existent dans tous les langages de programmation : Les langages évolués comme le langage C et les langages machine ou assembleurs.

Manipulation des variables non numériques

Pour manipuler des variables non numériques, du texte ou de la mise en page par exemple , on transcode le tout en en numérique (le texte et ses caractéristiques) et on se trouve ramené au cas précédent.

Pilotage d'un périphérique

Pour piloter les périphériques Disque / Ecran /Clavier / Cartes de communication etc. ce que l'on appelle les entrées / sorties, on lit et on écrit des commandes numériques et des données dans les registres de commande, de compte-rendu et de données des périphérique en question.

Pour le pilotage des périphériques on a donc besoin :

- Des lectures /écritures de données dans les registres des périphériques

Ces fonctionnalités qui relèvent des prérogatives des systèmes d'exploitation n'existent pas en langage C mais seulement en assembleur 80x86 en mode maître (noyau).

En langage C on utilise des primitives (API) fournies par le système d'exploitation pour accéder aux entrées / sorties et cela sous le contrôle de ce dernier.

Montée en puissance de la programmation

Aux fonctionnalités de base ci-dessus, il faut en rajouter un certain nombre permettant la montée en puissance de la programmation comme les fonctionnalités ci-dessous qui existent généralement dans les langages de programmation évolués :

- Les boucles itératives

Certains programmes effectuant un grand nombre de fois les mêmes calculs, il est impensable d'avoir à spécifier ces calculs un par un, il faut donc disposer d'un mécanisme permettant de factoriser les calculs analogues.

- Les tableaux

Certains programmes manipulant un grand nombre de données similaires, il est impensable d'avoir à manipuler ces données une par une, il faut donc disposer d'un mécanisme permettant de factoriser la manipulation des données similaires.

- Appel / Retour de sous-programme

Pour des raisons de lisibilité et de maintenance, il est impensable de développer un programme complexe d'un seul tenant, il faut donc disposer d'un mécanisme permettant de le découper en sous-programmes, chaque sous-programmes effectuant un traitement bien identifié.

La Gestion de la Pile

La gestion des sous-programmes implique la disponibilité d'une pile pour :
- sauvegarder les adresses retour
- passer les arguments
- stocker les variables locales

- Empiler / Dépiler

Ces fonctionnalités qui relèvent des prérogatives des systèmes d'exploitation ne sont pas directement accessibles en langage C mais seulement en assembleur 80x86.

En langage C on empile et dépile implicitement lors de la création ou de l'exécution de sous programmes, le compilateur fait appel à des primitives système pour cela.

La Programmation Système

Pour passer à la programmation d'un système d'exploitation multitâche nous devons disposer en plus d'un mécanisme d'interruption, soit les 5 fonctionnalités complémentaires suivantes :

- Appel / Retour de routine d'interruption
- Lancer une interruption, masquer les interruptions, démasquer les interruptions

Ces fonctionnalités qui relèvent des prérogatives des systèmes d'exploitation n'existent pas en langage C mais seulement en assembleur 80x86 en mode maître (noyau).

En langage C on utilise des primitives (API) fournies par le système d'exploitation pour accéder aux signaux Linux mais pas directement au mécanisme d'interruption.

Travail à effectuer

Nous aboutissons ci-dessous à la liste des fonctionnalités de base incontournables de la programmation applicative et système:

- Les 4 opérations : + - x /
- Le débranchement conditionnel
- Les lectures /écritures de données en mémoire
- La boucle itérative
- L'accès aux tables

- - - - - - - - - - -

- Lecture /écriture des registres des périphériques
- Empiler / Dépiler
- Appel / Retour de sous-programme
- Lancer une interruption, masquer les interruptions, démasquer les interruptions
- Appel / Retour de routine d'interruption

1) Vous allez par de courtes séquences d'instructions C, nous expliquer comment mettre en action les 5 premières fonctionnalités.

2) Ensuite plus difficile, par de courtes séquences d'instructions assembleur 80x86 (il va falloir vous y mettre, du moins en théorie) vous allez devoir nous expliquer comment mettre en action les 5 dernières fonctionnalités ci-dessus.