Institut Paul Lambin
Deuxième Informatique
Année académique 2016-2017

SYSTÈMES D'EXPLOITATION.

Alain Goffi Ing. Civil Ailg
(Rev.2016) Jean-Paul Colard, Lic.Sc.Math.App., ULB

Livres de référence :

- Computer Architecture, Design and Performance, B. Wilkinson Prentice Hall
- Systèmes d'exploitation, A. Tanenbaum, Interéditions
- Introduction to computer architecture and organization, H. Lorin Wiley
- Principes fondamentaux des systèmes d'exploitation, A.M. Lister Eyrolles
- MVS Principles of operations, IBM
- Journal of Reaserch and development, IBM
- Systems journal, IBM
- The design of the Unix OS. M.J. Bach, Prentice Hall
- Windoms NT Unleashed, SAMS

I. Introduction.

Avant d'avancer dans la matière, il est nécessaire de rappeler quelques grandes lignes du cours de première année.

1. Historique.

La plupart des ordinateurs modernes sont inspirés du schéma de la machine mécanique conçue par Charles Babbage en 1834. Cette machine exécutait étape par étape des instructions en vue de délivrer le résultat d'une opération arithmétique. La liste d'instructions à exécuter et les données étaient lues à partir de cartes perforées, introduites en mémoire (mécanique à l'époque), et d'autres cartes étaient perforées par la machine en vue de produire un résultat.

Tous les composants de base d'un ordinateur moderne étaient rassemblés dans cette machine :

- une mémoire contenant les instructions à exécuter et les données associées à ces instructions;
- une unité de contrôle assurant le transfert des informations entre la mémoire et l'unité centrale;
- une unité centrale capable d'exécuter des instructions;
- des unités d'entrée/sortie permettant l'échange d'informations avec l'extérieur.

L'architecture de cette machine n'a pu être convenablement exploitée que 100 ans plus tard grâce aux possibilités de l'électronique. Les développements les plus remarquables furent effectués par Von Neumann et ses collègues d'où l'appellation décernée aux ordinateurs répondant à la définition précédente.

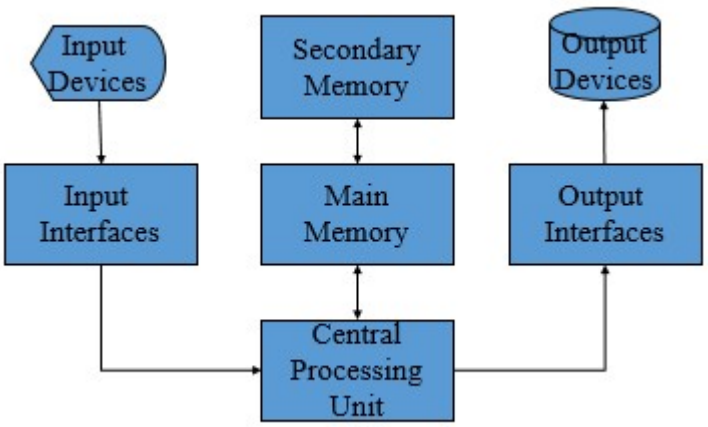


Fig 1. Composants de base.

Il existe d'autres types d'architectures:

- les Dataflow Computer: systèmes pour lesquels les instructions ne sont pas exécutées en séquence;
- les réseaux neuronaux.

2. Machines dites de Von Neumann

A. Mémoires

Les instructions et les données sont stockées en mémoire centrale (Central Storage) appelée également mémoire principale (Main Memory). Ces mémoires sont organisées de telle sorte qu'un mot d'information (plusieurs bits contigus) est localisé à une adresse directement accessible : Random Access Memory.

On trouve dans la majorité des systèmes une mémoire secondaire (Secondary Memory) ou mémoire étendue (Expanded Memory) qui a pour but principal d'étendre de manière économique la capacité des systèmes. Ces mémoires font partie de ce qu'il est convenu d'appeler la hiérarchie des mémoires qui fera l'objet du premier chapitre de ce cours.

Le stockage en mémoire des instructions et des données est une caractéristique typique des machines dites de Von Neumann. Ces informations transitent vers l'unité centrale par un chemin identique. Les architectures dans lesquelles les données et les instructions parviennent à l'unité centrale par des chemins différents répondent à l'appellation architecture Harvard.

B. Registres.

Le processeur dispose d'un certain nombre de registres internes lui permettant de réaliser des opérations spécifiques sur des nombres, des adresses et des informations de contrôle. Le nombre et le type de registres varient en fonction des architectures[1].

Toutes les architectures proposent cependant un registre appelé le compteur ordinal (Program Counter, Instruction Pointer, Program Status Word…), registre interne contenant l'adresse mémoire de la prochaine instruction à exécuter. On trouve parfois un registre de pile (Stack pointer) qui contient l'adresse mémoire du sommet d'une pile (stack). Cette pile reprenant l'ensemble des adresses de retour auxquelles le contrôle doit être rendu en sortie de routines/programmes appelés successivement.

[1] Le mot architecture doit, à partir de cet instant, être compris dans le sens implémentation détaillée par opposition à l'architecture d'ensemble reprenant les composants de base.

Un ensemble de registres généraux (General Purpose Register), de registres d'adresses sont également présents. Ces registres sont plus rapides que la mémoire centrale, sont parfois implicitement employés dans une instruction et permettent donc d'augmenter la vitesse de traitement de l'information.

C. Instruction set.

Les instructions exécutées par l'unité centrale sont appelées instructions machine et font partie de l'instruction Set attaché à l'architecture. Une séquence d'instructions appartenant à l'instruction set est appelée un programme. Chaque instruction permet de préciser l'opération à effectuer mais aussi les opérandes avec lesquels l'opération doit s'exécuter. On précise soit explicitement l'opérande (il s'agit d'instructions de type Immediate en Assembler) soit l'adresse de ces opérandes (Three/two/one address instruction format).

D. Microcode.

L'unité centrale des machines modernes est tellement évoluée qu'il devient impossible d'obtenir un fonctionnement correct par une interconnexion de portes (gates).

Les opérations requises pour exécuter une instruction consistent dans les cas simples (hors pipelining) à lire l'instruction en mémoire, incrémenter le compteur ordinal (Fetch Cycle) et ensuite à exécuter cette instruction c.à.d. lire les données en mémoire et y stocker le résultat (Execute Cycle). Chacune de ces étapes peut faire l'objet d'une micro-instruction. Une séquence de micro-instructions est ainsi établie pour chaque instruction appartenant à l'instruction set. Ces séquences sont appelées microprogrammes ou microcode.

E. Les interruptions.

L'unité centrale répond à des signaux qui viennent de l'intérieur ou de l'extérieur du système par des interruptions. A la réception du signal, le hardware interrompt la séquence d'exécution des instructions afin d'analyser l'événement et de prendre les actions adéquates en réponse à cet événement.

Lors de l'interruption, le contenu du compteur ordinal est sauvegardé à une adresse prédéfinie en mémoire. Son contenu est remplacé par une nouvelle valeur qui est égale à l'adresse en mémoire de la première instruction du programme chargé d'analyser l'événement. Nous avons expliqué en première année les contraintes que devaient respecter ces programmes d'analyse des interruptions et les notions de classe et de priorités des interruptions.

F. Les unités entrées/sorties.

Les unités d'entrées/sortie attachées à un système échangent des informations avec celui -ci soit au travers d'un bus (implémentation microprocesseur) soit au travers d'un canal. Les unités d'entrées/sorties sont pilotées par des contrôleurs : soit par un disk controller que l'on retrouve sur la carte d'un disque dur PC soit par un contrôleur

capable de gérer un ensemble de disques (32 ou 64) un cache et/ou du Non Volatile Storage.

G. Mode Superviseur.

L'objectif d'un système d'exploitation est de gérer au mieux les ressources qui lui sont confiées tout en se montrant le plus économe possible dans la consommation de ces ressources. Le "au mieux" va devoir être adapté en fonction des circonstances c'est à dire du type de sollicitations soumises au système.[2]

Le système d'exploitation ne diffère pas fondamentalement des programmes d'application classique. Il est un programme donc une séquence d'instructions stockées en mémoire, tout au plus s'exécute-t-il dans un mode appelé le mode Superviseur et est-il appelé à répondre à des situations particulières auxquelles les programmes classiques ne sont jamais confrontés. Le mode superviseur est un état particulier de l'unité centrale qui permet l'exécution d'instructions dites privilégiées.

[2] en d'autres mots la charge de travail ou workload.

3. Evolution des systèmes d'exploitation.

Les possibilités des équipements électroniques des premières machines étaient limitées, tant par la taille de la mémoire que par la vitesse de traitement des unités centrales. Les systèmes d'exploitation n'avaient pas vraiment de raison d'exister, leur consommation des maigres ressources disponibles justifiait la nécessité pour un programme de prendre en charge des fonctions communes : ordres de lectures, ordres d'impression, gestion de la mémoire, ...

L'extension des capacités mémoire a permis de rassembler sous le label Système d'exploitation un ensemble de routines assurant ces services généraux. Des opérations de lecture ne sont plus réalisées directement par un programme mais font l'objet d'une demande de service à une routine appartenant au système d'exploitation qui rendra le service demandé. Ces routines du système d'exploitation pourront même rester en mémoire entre les exécutions des différents programmes qui les sollicitent.

Ces routines sont chargées lors de l'opération appelée IPL (et oui!) pour Initial Program Load ou boot pour les accros PC.

Ces premiers systèmes supportaient l'exécution d'un seul programme "utilisateur" censé solliciter le programme "Système d'exploitation" en vue d'obtenir des services. La mémoire adressable correspond exactement à la taille de la mémoire réelle, il n'est pas question de mémoire secondaire ou étendue dans ce genre de systèmes.

A. Le plus simple : une seule tâche active

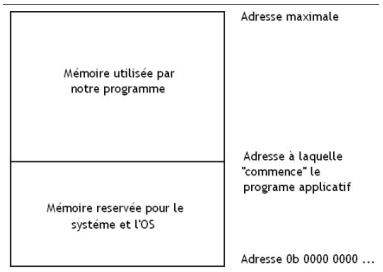


Fig. 2. OS mono-tâche.

L'operating system de base. Un seul programme peut être exécuté. Il partage la mémoire avec l'OS dont le but est de rendre des services communs tels que les Entrées/Sorties, ... La mémoire adressable correspond univoquement à la mémoire réelle. Type d'O/S que l'on trouve dans les années 60 en MF et dans les années 70 en PC.

B. Un peu plus évolué : nombre fixe puis variable de tâches.

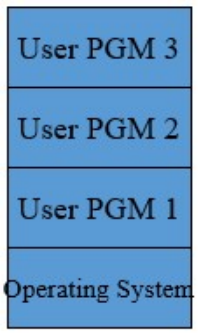


Fig. 3. OS multi-tâches

Les systèmes d'exploitation basés sur les principes précédents conduisaient à un gaspillage de ressources. Le processeur est en effet en attente sur toutes les interruptions. Un nombre de tâches fixe au départ puis variable permet de remédier à cet inconvénient : le processeur active une nouvelle tâche si une interruption survient. Afin de protéger la portion de mémoire allouée à chacune des tâches, le système gère une clé de protection. L'OS utilise la clé 0 lui permettant d'adresser la totalité de la mémoire.

C. Adressage virtuel. L'introduction de la pagination.

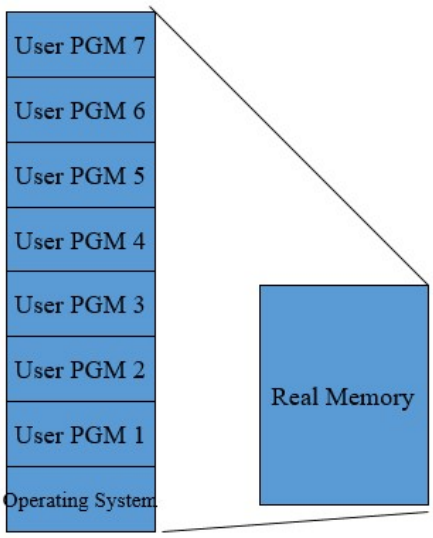
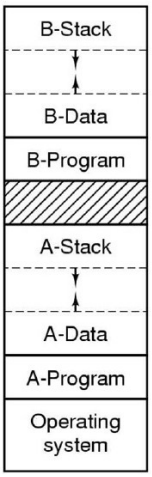


Fig. 4. Mémoire virtuelle.

Charger un nombre variable de tâches en mémoire impose des contraintes sur l'espace adressable de chacune de ces tâches. La mémoire réelle n'est pas extensible et son partage implique une réduction dans la taille des programmes et des données chargées. Des compromis difficiles devaient être trouvés entre le nombre de tâches concurrentes et les besoins en adressage de ces dernières. L'idée de la mémoire virtuelle consiste à présenter aux programmes actifs un espace contigu de mémoire dont le correspondant physique est soit de la mémoire réelle soit un support disque.

La mémoire virtuelle est en fait constituée de mémoire réelle, d'une ou de plusieurs mémoires auxiliaires et d'un composant (ASM) Software les gérant.

Tous les accès de l'unité centrale doivent se faire sur un frame présent en mémoire centrale. Si ce frame est absent, une condition dite Page Fault est rencontrée et l'ASM ramène le frame en mémoire en réponse à l'interruption. Cette gestion de la mémoire est complètement transparente.

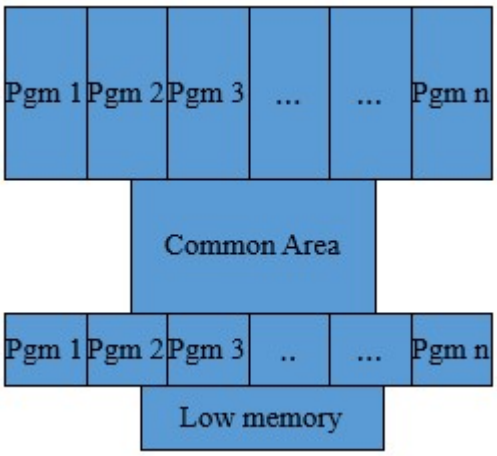


D. Segmentation.

Fig. 5. Segmentation.

La mémoire virtuelle, bien que plus étendue que la mémoire réelle, est un espace linéaire partagé par des tâches concurrentes. L'introduction d'une segmentation de cette mémoire virtuelle permet d'attribuer à chaque tâche un segment de mémoire dont la taille dépend du range d'adresses que le processeur peut générer. Une adresse en 64 bits conduit à une mémoire virtuelle de 2**64 soit 16 Exabytes.

E. Adressage étendu.



L'attribution d'un espace linéaire de mémoire virtuelle (address space) à chaque tâche implique l'existence de zones de mémoire communes. Les mécanismes de protection empêchant les accès concurrents à des éléments de la mémoire virtuelle impliquent des mécanismes de communication entre les tâches. L'utilisation de ressources communes implique l'existence de mécanismes de sérialisation et d'attribution de priorités. Enfin, la mémoire virtuelle doit être gérée, nous verrons les mécanismes de gestion basé sur la pagination et la segmentation.

Fig. 6. Adressage étendu.

La multiplication des tâches dans le système donc des address spaces a introduit des contraintes sur la mémoire réelle. La notion de hiérarchie des mémoires a été introduite afin de réaliser des systèmes rapides, supportant de manière efficace la multi-programmation, la pagination et la segmentation.

F. Composants d'un système d'exploitation.

Un système d'exploitation n'est pas construit de manière monolithique, il est plutôt constitué de composants aux fonctions bien précises, chacun des composants pouvant être sollicité par l'autre.

1. Program Manager : le composant en charge de l'activation des programmes.
2. Interrupt Handler : le composant en charge de la réponse aux interruptions
3. I/O & Files : les composants chargés de gérer les périphériques, réaliser les opérations d'entrée/sortie, gérer les catalogues et directories ainsi que les accès aux fichiers (méthodes d'accès).
4. Resource controller : le composant en charge de la gestion des ressources (sérialisation et synchronisation)
5. Memory Managers : les composants en charge de la gestion des mémoires : Real Storage Manager, Virtual Storage Manager, Auxiliary Storage Manager.
6. Task Manager : le composant en charge de la gestion des tâches dans le système.
7. Recovery Manager : le composant en charge de la gestion des erreurs.