Systèmes d'exploitation

Soraya Zertal

Laboratoire Li-PaRAD, Université de Versailles

E-mail: soraya.zertal@uvsq.fr

Plan :

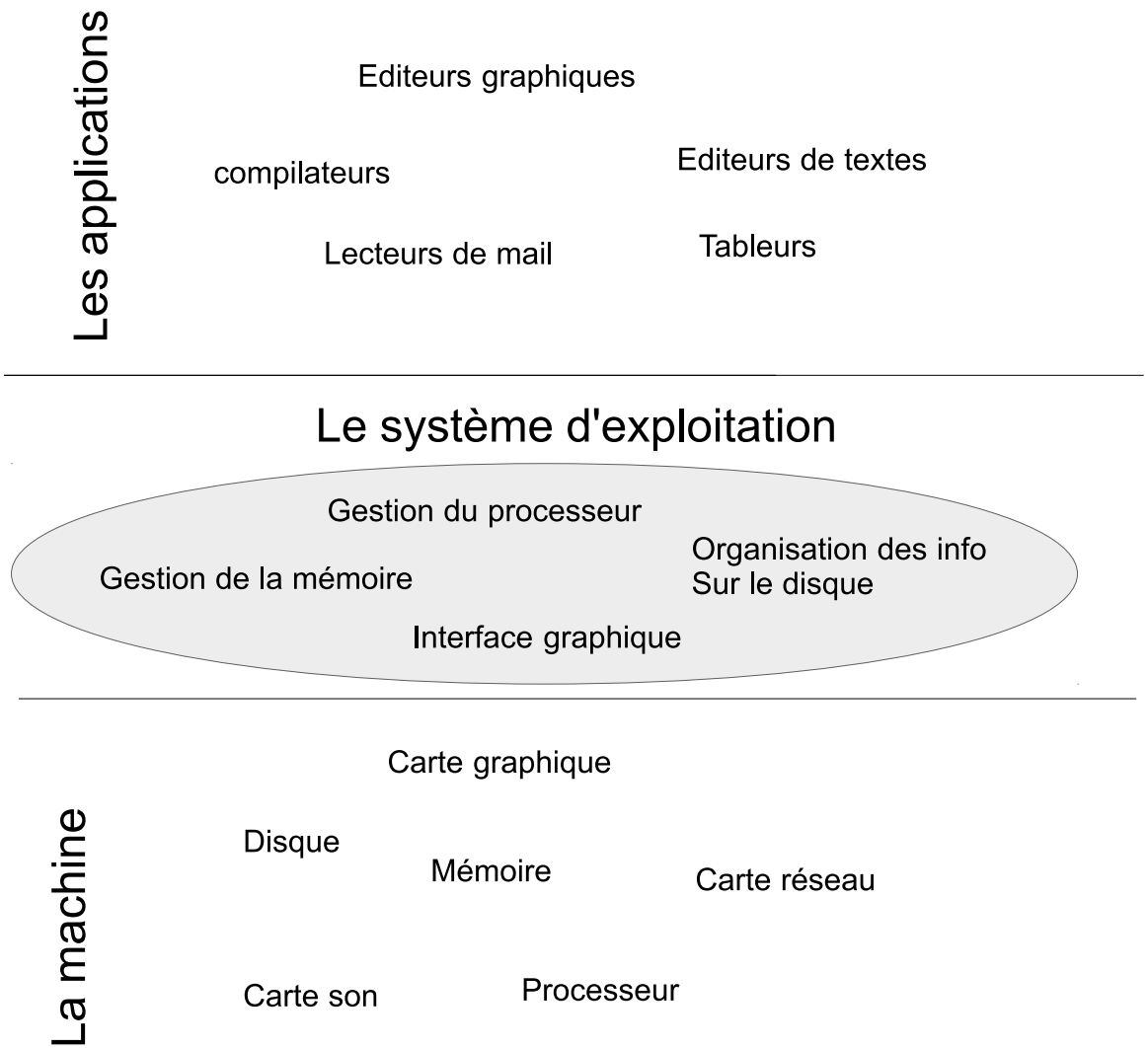
- Introduction aux systèmes d'exploitation
- Système de fichiers
- Processus et Threads
- Gestion de la mémoire
- Entrées/Sorties

Bibliographie

- Andrew Tanenbaum, *Systèmes d'exploitation*, Pearson Education France, 2003
- Jean-Marie Rifflet, *Unix, programmation et communication*, Dunod, 2003
- A. Silberschatz, P. B. Galvin and G. Gagne, *Principes des systèmes d'exploitation*, Vuibert, 2008

Cours 1: Introduction aux systèmes d'exploitation

Qu'est-ce qu'un système d'exploitation ?



Rôle du système d'exploitation

1. Masquer la complexité du matériel qu'il gère.
   Exemple : Pour lire sur un disque, on ne veut pas avoir à gérer le déplacement des têtes de lecture.

2. Gérer les ressources et les faire fonctionner ensemble de manière sûre et équitable (en temps et en espace).

Le Processeur (CPU)

Le CPU extrait des instructions de la mémoire et les exècute.

- Deux modes de fonctionnement :
  - Le mode utilisateur restreint l'accès aux ressources de l'ordinateur (mémoire, périphériques).
  - Le mode noyau a accès à toutes les ressources, sans contrainte.
- Un programme fonctionnant en mode utilisateur dispose d'une zone de mémoire qui lui attribuée, et n'a accès aux ressources de l'ordinateur qu'à travers des appels système, qui eux s'exécutent en mode noyau.
- Cela permet de limiter l'accès à certaines données sensibles ou privées, ce qui permet une sécurisation de la mémoire et des périphériques.

Registres CPU

- registres de données contenant les variables importantes et des résultats temporaires ;
- le compteur ordinal qui contient l'adresse de la prochaine instruction à extraire de la mémoire ;
- le pointeur de pile qui désigne l'adresse du sommet de la pile ;
- le registre d'état qui stocke l'état actuel du processeur : mode utilisateur/noyau, bits de comparaison sur le dernier calcul,...;
- ...

Appels système

Un appel Système est une fonction qui permet d'effectuer une tâche au niveau système pour le compte d'un processus utilisateur.

- Les processeurs ont généralement une instruction dédiée pour les appels systèmes.
- Le matériel peut déclencher des appels système par exemple pour un périphérique branché et/ou débranché, ...

La mémoire

La mémoire principale (RAM) permet de stocker les programmes en cours d'exécution et leurs données temporaires.
Plusieurs niveaux de mémoire avec des capacités et des temps d'accès differents : Registres, Cache, Mémoire principale (RAM), Disque magnétique

Chaque niveau s'éloigne du processeur $\Rightarrow$ temps d'accès plus longs, prix moindre.

La plupart des ordinateurs dispose d'une ROM (Read Only Memory) : mémoire à contenu fixe qui contient entre autres le programme de démarrage

La mémoire : espace physique

- Rappel : La mémoire est un ensemble d'octets numérotés par des adresses.
- Généralement les octets de la RAM se situent entre les adresses 0 et *taille de la RAM*-1.
  Exemple : 2 GB de RAM $= 2 \times 2^{30}$ octets
- Des adresses en dehors de la RAM peuvent servir à:
  - l'accès aux différents périphériques
  - l'accès à la ROM
- L'espace d'adressage physique d'un ordinateur est l'ensemble de toutes les adresses possibles.
  Exemple : sur un processeur 32 bits, l'espace d'adressage physique est situé entre les adresses 0 et $2^{32} - 1$

La gestion de la mémoire

Faire de la mémoire un seul espace commun pour :

- Les codes et les données des programmes
- Les données du système

Problèmes :

- Les programmes peuvent lire et modifier les données du système et des autres programmes, sans aucun contrôle.
- Un programme erronné peut faire arrêter tout le système.
- Un programme peut arrêter le système volontairement.

La MMU

- La mémoire principale peut contenir plusieurs programmes à la fois.
  Une unité de gestion de mémoire (MMU) permet de :
  1. Protéger les programmes les uns des autres, et le noyau de tous les programmes (sécurité du système)
  2. Gérer la réallocation
- La MMU permet le *mapping* d'un ensemble d'adresses physiques vers un espace d'adressage virtuel (en mode utilisateur uniquement) en utilisant une paire de registres base/limite.

La mémoire cache

- Dissimule la relative lenteur de la RAM, en stockant les données les plus récemment utilisées (statistiquement, susceptibles d'être réutilisées à nouveau).
- Lors du passage à l'exécution d'un autre programme en mémoire, il y a un changement de contexte.
  Le contenu des registres est stocké dans une zone mémoire dédiée du programme en cours. Le nouveau programme et ses registres sont chargés.
  La mémoire cache contient alors les données de l'ancien programme, et les premiers accès mémoire du nouveau seront coûteux en temps (défauts de cache)

Les entrées/sorties

- Les périphériques, plus ou moins complexes, sont reliés à la machine par le biais de contrôleurs d'entrée-sortie, qui assurent le contrôle effectif de chaque périphérique.
- Un logiciel pilote est nécessaire pour homogénéiser le système avec chaque type de contrôleur : disque, carte graphique, imprimante ...

Les entrées/sorties : suite

Recyclage des concepts :
Exemple : sous Unix, toute ressource du système est accessible à travers un fichier (fichiers réguliers, entrées/sorties standard, ...)

⇒ Un nombre très restreint d'appels système est suffisant pour accèder à l'ensemble des ressources du système.

Les processus

Définition :

- Un processus est un programme en cours d'exécution.
- Il dispose de son espace d'adressage propre, allant de l'adresse 0 à une adresse limite déterminée.
- L'espace d'adressage contient :
  - le code binaire (les instructions) du programme
  - les données du programme : variables et paramètres statiques
  - la pile d'exécution

Les processus : pile d'exécution

La pile est une zone de mémoire qui contient, entre autres :

- les paramètres des fonctions dont l'exécution du programme est entrée mais n'est pas encore sortie ;
- l'adresse de retour d'appels de fonctions.

Exécution en temps partagé
Le système attribue des parts de temps CPU à chacun des processus. Quand le processus en cours atteint la limite du temps attribué, le noyau attribue CPU à un autre processus.
Le noyau stocke une table des processus en mémoire avec les informations spécifiques à chaque processus (registres CPU, registres MMU, etc.).

Création / suppression de processus Chaque processus peut lui-même créer des processus, qui héritent de ses caractéristiques (UID, GID, droits d'accés). Il en résulte une **hiérarchie des processus**, ayant une structure arborescente.

Environnement d'un processus sous Unix

- Tout processus dispose d'un environnement : variables, fichiers ouverts (entrée et sortie standard incluses).
- Un processus nouvellement créé hérite de l'environnement de son processus parent.

Le shell

- C'est l'interface utilisateur de base. Il exécute des commandes données au clavier par l'utilisateur, éventuellement complétées d'arguments.
- Le recyclage des concepts offre la possibilité à l'utilisateur :
  - de rediriger l'entrée ou la sortie standard par défaut vers un fichier, une imprimante ...
  - d'utiliser la sortie standard d'un processus comme entrée standard d'un autre processus : chaînage de commandes à l'aide de tubes, emploi de commandes filtres.
  - en combinant ces possibilités de redirection, d'effectuer des opérations complexes à l'aide de commandes simples.
- Possibilité d'effectuer des séquences complexes de commandes stockées dans un fichier (script shell)

Exemples de commandes shell

- `date`
  Le shell crée un processus fils qui exécute le programme `date`. Le shell attend la fin de l'exècution du programme, puis rend la main à l'utilisateur.
- `date >fichier`
  Redirection de la sortie standard. Au lieu d'afficher le résultat de la commande `date` sur le terminal, ce résultat est stocké dans le fichier `fichier`.
- `sort <fichier1 >fichier2`
  Le programme `sort` est exécuté, son entrée standard étant lue dans le fichier `fichier1` et sa sortie écrite dans `fichier2`.

Exemples de commandes shell

- `cat fichier1 fichier2 | sort >/dev/lp`
  Le programme `cat` est exécuté avec les arguments `fichier1` et `fichier2`. La sortie standard du processus est utilisée comme entrée standard d'un autre processus qui exécute le programme `sort`, dont la sortie standard est redirigée vers le fichier périphérique `/dev/lp` (typiquement, l'imprimante).