Systèmes d'exploitation

(*Operating Systems*)

Introduction

SITE : http://www.sir.blois.univ-tours.fr/˜mirian/

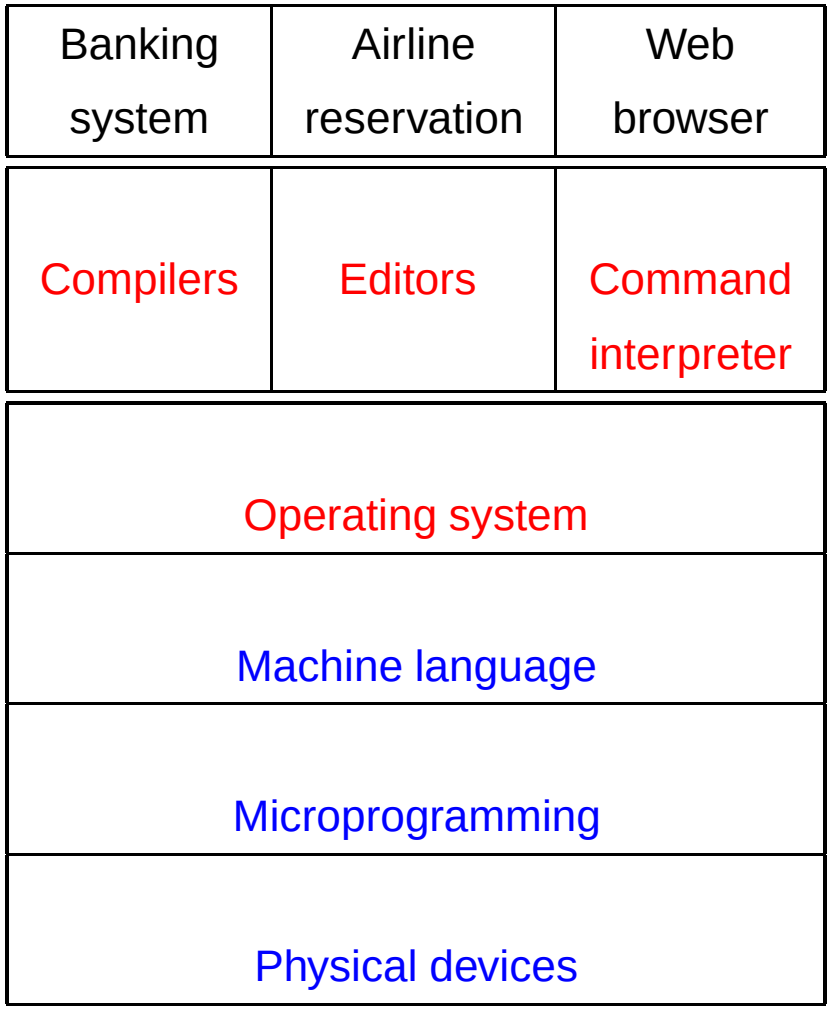
Qu'est-ce qu'un SE ?

- Ensemble de logiciels qui tournent en permanence sur un ordinateur et le contrôlent à partir de son démarrage (*boot*) et tant que celui-ci est allumé.
- Exemples :
  - Unix : Créé en 1969, rapidement multi-utilisateur, écrit en langage C.
  - Linux : Clone gratuit d'UNIX pour les PC, *open source*.
  - Mac OS : Premier à proposer le concept des fenêtres, du glisser-déposer, la corbeille, le plug-and-play; aujourd'hui possède le *noyau* Linux, avec une interface graphique élégante et ergonomique, et optimisation particulière des traitement multimédia.
  - MS-DOS (Microsoft disque operating system) : SE des premiers PC, mono-utilisateur, mono-tâche, interface ligne de commande.
  - MS-Windows : Inspiré par l'interface Macintosh; tout d'abord, une *coquille* graphique pour DOS. Seulement à partir de Windows 95 nous commençons à assister à un transfert de nombreuses fonctionnalités de DOS vers Windows.
  - Windows NT : Système d'exploitation indépendent de DOS. Techniquement nettement supérieur à Windows.

Systèmes d'exploitation

- Ordinateur: Sans le logiciel (*software*), machine sans utilité
- Deux types de logiciels:
  1. **Programmes systèmes (*system programs*):** gère le fonctionnement de l'ordinateur
  2. **Programmes d'application:** exécute le travail demandé par les utilisateurs
- **Systèmes d'exploitation:** programme système **fondamental**
  - Contrôle toutes les ressources de l'ordinateur
  - Base sur laquelle les programmes d'application sont écrits

Système informatique (1)



Hardware, Programmes système et Programmes d'application

Système informatique (2)

1. Matériel (*hardware*) : Ressources informatique de base: CPU (UC: unité centrale), mémoire, dispositifs E/S
2. Système d'exploitation: Programme intermédiaire entre l'utilisateur et le matériel
3. Programmes d'application: Comment utiliser le matériel pour résoudre les problèmes informatiques des utilisateurs
4. Utilisateurs: Personnes, machines, autres ordinateurs

Rôles du SE

- Le système d'exploitation joue deux rôles :

1. d'une **machine virtuelle** (abstraite)
   Le SE présente au programmeur une **interface d'accès aux ressources** de l'ordinateur (sous forme d'appels système). Ainsi le programmeur peut faire **abstraction** des détails de fonctionnement des ressources.

   Cette interface est fondée sur des **objets abstraits** dont les plus importants sont les **fichiers** et les **processus**. Par exemple, le programmeur voit un disque comme une collection de fichiers qui peuvent être lus, écrits et fermés

2. d'un **administrateur de ressources**
   Le SE gère l'utilisation des ressources par différents utilisateurs et les éventuels conflits.

Buts d'un SE

1. Fournir un environnement où l'utilisateur puisse exécuter des programmes
2. Rendre le système informatique pratique pour l'utilisateur
3. Utiliser le matériel de façon efficace

Définitions des systèmes d'exploitation

- Programme d'allocation des ressources: gérer et allouer des ressources
- Programme de contrôle: contrôler l'exécution des programmes des utilisateurs et l'opération des dispositifs d'entrée/sortie
- Noyau *(Kernel)*: le programme qui est exécuté tout le temps (tout les autres sont des programmes d'application)

Évolution des SE

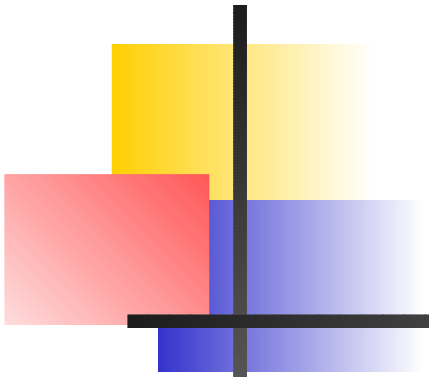
- SE: développé pour faciliter l'utilisation du matériel
- Projet et utilisation des SE ont engendré des modifications du matériel
- Vision historique des SE: les problèmes des SE ont engendré des innovations du matériel
- Les SE ont une histoire ancienne qui commence au moment où ils ont commencé à remplacer les opérateurs des ordinateurs jusqu'à aujourd'hui avec des systèmes multiprogrammation.

Tâches d'un système d'exploitation

- Gestion de processus
- Gestion de la mémoire
- Gestion des fichiers
- Gestion des E/S

Les programmes utilisateurs peuvent accéder à ces différentes fonctionnalités à l'aide des **appels système**.

Pour créer un système aussi grand et complexe qu'un SE, il est nécessaire de le découper en pièces plus petites



Pour partager les ressources, comment les SE peuvent assurer qu'un programme qui fonctionne mal ne va pas causer des problèmes à l'exécution d'autres programmes?

Avec une protection matérielle : le mode double.

- Le matériel permet **2 modes de fonctionnement**:
  - **Mode moniteur ou superviseur ou système ou privilégié** (*Monitor mode, supervisor mode, system mode*): exécution de la part du SE
    Instructions privilégiées: instructions machine risquant de nuire.
    Les instructions privilégiées sont exécutées seulement en mode superviseur.
  - **Mode utilisateur** (*User Mode*): exécution de la part de l'utilisateur.
    S'il se produit une tentative d'exécuter une instruction privilégiée, le matériel ne la réalise pas mais traite l'instruction comme illégale et bloque le SE.

Fonctionnement en mode double

- Au moment d'initialiser le système, le matériel démarre en mode superviseur
- Ensuite le SE est chargé et démarre les processus utilisateurs en mode utilisateur
- Chaque fois qu'un déroutement (*trap*) ou une interruption se produit, le matériel commute du mode utilisateur au mode superviseur
  $\Rightarrow$ Chaque fois que le SE prend le contrôle de l'ordinateur, il est en mode superviseur
  $\Rightarrow$ Le système commute toujours au mode utilisateur avant de donner la main à un programme utilisateur

Certaines instructions sont privilégiées, comme les instructions E/S.

Comment un programme utilisateur peut donc exécuter des E/S?

En utilisant les appels système

Appels système (1)

- Un appel système est une **fonction fournie par le noyau** (*kernel*) d'un SE et **utilisée par les programmes s'exécutant dans l'espace utilisateur** (en d'autres termes, tous les programmes distincts du noyau).
- Le rôle du noyau est de gérer les ressources matérielles et de fournir aux programmes une interface uniforme pour l'accès à ces ressources.
- Quelques appels systèmes classiques :
  - `open, read, write` et `close` qui permettent les manipulations sur les systèmes de fichiers,
  - `alloc, free` pour allouer et désallouer de la mémoire.
- Sur la majorité des systèmes d'exploitations, les appels système peuvent être utilisés comme de simples fonctions écrites en C.
- Sur la plupart des noyaux (notamment les noyaux monolithiques comme le Noyau Linux) les appels systèmes sont implémentés par une instruction machine (**interupt, supervisor call, ...**) qui fait basculer le processeur dans le noyau en **mode superviseur** (en ayant convenablement passé les paramètres de l'appel système, par exemple dans les registres).

Appels système (2)

Vu du programme applicatif, un appel système est atomique (il s'est exécuté -éventuellement en erreur- ou pas).

Catégories

1- Contrôle de processus:

charger, exécuter, créer, terminer des processus, obtenir, signaler des événements, libérer de la mémoire, etc

2- Manipulation de fichiers:

créer, supprimer, ouvrir, fermer, lire, écrire, repositionner,etc

3- Gestion de périphériques:

demander, libérer, obtenir, attacher, etc

4- Entretient d'information:

obtenir, définir l'heure ou la date, définir les données du systèmes

5- Communications

créer, supprimer des connexions de communication, envoyer, recevoir de messages, transférer des informations sur les états, etc

Concepts de base (1)

- **Processus** : Un processus est un programme en exécution
  Programme: entité passive
  Processus: entité active (compteur d'instructions)
- **Fichier** : une unité de stockage logique, c'est-à-dire, un ensemble d'informations en relation entre elles, qui est enregistré sur la mémoire auxiliaire (disque).
  - Les processus utilisent la **mémoire vive/cache/registres** pour sauvegarder leurs codes et leurs données; mais ces types de mémoire sont **volatiles**.
  - Le stockage dans un fichier permet la préservation d'une grande quantité d'information de façon **non volatile** (résiste à la fin d'un processus) et rend cette information disponible à plusieurs processus.
  - **Système de fichiers:** Partie du SE responsable de la gestion de fichiers.
  - Le SE établit une correspondance entre les fichiers et les dispositifs physiques (non volatile).
  - Le système de fichiers se présente généralement comme une structure arborescente de répertoires (ou dossiers) dont l'origine est appelée racine.

Concepts de base (2)

- **Shell** : Principale interface entre un utilisateur placé devant son terminal et le SE, sauf si l'utilisateur a recours à une interface graphique.
  - Exemples de Shell : `sh, csh, ksh, bash`
  - Quand un utilisateur se connecte un shell est lancé. Ce shell a pour entrée standard le clavier du terminal et pour sortie standard son écran. Il commence par afficher un **prompt** qui indique à l'utilisateur que le shell est prêt à recevoir une commande.
- Exemple :

```
$ date
Mon Aug 20 17:25:58 CEST 2007
$
```

Le prompt est `$`. L'utilisateur tape `date`. Le shell crée un processus (enfant) qui exécute le programme `date`. Pendant que le fils s'exécute, le shell attend sa terminaison. Quand elle a lieu, le prompt réapparaît et le shell attend de lire la commande suivante.

L'initialisation d'un ordinateur (1)

Comment un ordinateur commence à fonctionner?

Quand l'ordinateur est allumé, le code trouvé dans le premier secteur du disque de *boot* est lu dans la mémoire et exécuté.

- Dans une **disquette**, ce secteur contient le programme d'amorçage (*bootstrap program*).
- Dans un **disque dur**, ce secteur contient un petit programme et une table de partitions (un disque dur est divisé en partitions).
  - Le programme est exécuté pour lire la table de partitions et sélectionner la partition active.
  - La partition active possède un programme d'amorçage dans sont premier secteur qui est alors chargé et exécute (comme dans le cas de la disquette).

L'initialisation d'un ordinateur (2)

- Programme d'amorçage (*bootstrap program*):
  - Initialise tous les aspects du système (CPU, contrôleurs de périphériques, mémoire)
  - Sait comment charger/démarrer le système d'exploitation
    1. Trouver et charger en mémoire le noyau (*kernel*) du SE
    2. SE exécute le premier processus (*init*) et attend un événement (interruption)
  - **Interruption** : événement qui modifie le flux de commande d'un programme
    1. Interruptions matérielles : Permettent la prise en compte d'une requête de service système (mémoire, contrôleur de périphérique, clavier, lecteur, ...). À tout moment le matériel peut activer une interruption.
    2. Interruptions logicielle : Activée par l'exécution d'un appel système (**system call** ou **monitor call**)

Traitement d'interruptions

- Le programme en cours est arrêté.
- Le système d'exploitation préserve l'état de la CPU (sauvegarde des registres et du compteur ordinal).
- SE détermine le type d'interruption.
- Pour chaque type d'interruption une partie de code du SE détermine l'action qui doit être prise.
- Dès que cette procédure est terminée, le programme interrompu reprend son exécution.
- Lors de la reprise, la machine doit se trouver exactement dans l'état où elle était au moment de la prise en compte de l'interruption.
- SE modernes: orienté interruptions (*interrupt driven*)