Systèmes et Réseaux (ASR 2) - Notes de cours
Cours 1

Anne Benoit

January 28, 2015

Fonctionnement du cours

Planning du cours: `http://graal.ens-lyon.fr/~abenoit/asr15/`

2h cours / 2h TD ou TP par semaine; cours le mercredi à 10h15 sauf contre-ordre.

Première semaine: pas de TD, et deux cours la deuxième semaine. Cours mercredi 28 janvier à 10h15, mercredi 4 février à 10h15, et jeudi 5 février à 10h15, puis tous les mercredis.

Evaluation

Contrôle continu: 1 partiel et 1 (ou plusieurs) DM, peut être des devoirs sur table "surprise". Examen à la fin du semestre, note finale: (CC+2NE)/3. Questions de cours/TD.

Résumé du cours

Suite du cours ASR1, on se concentre sur le S et le R (architecture traitée en ASR1).

Pré-requis: notions d'architecture, maîtrise du C.

Plan du cours:

Système:

- Introduction sur les composants d'un OS, notion de processus et de threads
- Synchronisation des processus et deadlocks
- Ordonnancement des processus
- Mémoire et mémoire virtuelle

Réseaux:

- Structure d'un réseau, les différentes couches
- TCP/IP, MAC
- Algorithmes de routage
- Contrôle de congestion

Objectifs: connaître les fondamentaux. On ne détaillera pas le fonctionnement de tous les systèmes et de tous les protocoles réseaux. Présentation des idées clés qui sont à la base de tout système et protocole réseau. Avant de regarder comment ça marche, on étudiera "pourquoi" cela fut inventé, quel problème cela résout, et puis cela fait "quoi"?

Bibliographie

Silberschatz, Galvin et Gagne, "Operating System Concepts" pour la partie systèmes. Les curieux peuvent consulter et expérimenter avec SOS, Simple Operating System, `http://sos.enix.org/`.

James Kurose et Keith Ross, "Analyse structurée des réseaux" pour la partie réseaux.

1 Structure des systèmes d'exploitation

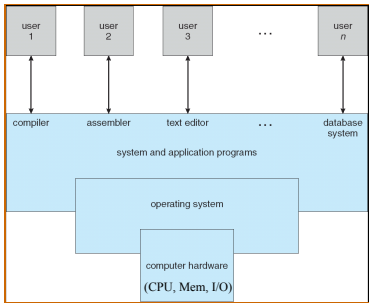
Objectifs: présenter les différents composants d'un OS et les bases de l'organisation d'un système informatique; décrire les services fournis par l'OS aux utilisateurs, processus et autres systèmes; discuter des différentes façons de structurer un OS.
Aperçu:

- OS: intermédiaire entre utilisateur et matériel (hardware).
- But: environnement dans lequel l'utilisateur peut exécuter ses programmes facilement et efficacement.
- OS: logiciel qui gère le matériel, qui doit fournir des mécanismes appropriés. Il cherche à optimiser l'utilisation du matériel.
- Différences entre OS, suivant buts recherchés: facile d'utilisation, performant, combinaison? Buts doivent être identifiés avant de se lancer dans la conception d'un OS.
- L'OS doit être conçu morceau à morceau.

1.1 Organisation et architecture d'un système informatique

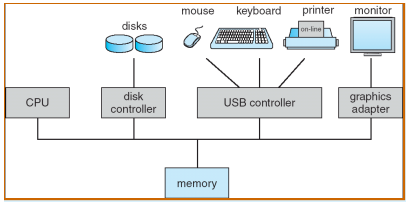
Système informatique: 4 principaux composants:

1. Matériel: CPU, mémoire, périphériques d'entrée-sortie (I/O);
2. OS: contrôle le matériel;
3. Programmes des applications: utilisation des ressources système pour résoudre des problèmes définis par les utilisateurs;
4. Utilisateurs: les personnes.



Organisation.

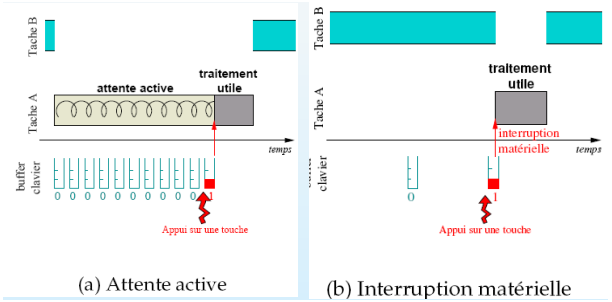
- Un ou plusieurs CPUs, et contrôleurs de périphériques qui sont connectés via un unique bus qui donne accès à une mémoire partagée;
- Execution concurrente, compétition pour cycles mémoires;
- Besoin de synchroniser les accès à la mémoire (contrôleur de mémoire);
- Périphériques I/O et CPU peuvent s'exécuter en parallèle;
- CPU: déplace données de/vers la mémoire vers/de des buffers locaux (registres); opérations load/store;
- I/O entre le périphérique et le contrôleur (ou buffer local); informe le CPU quand il a terminé via une interruption.



Gestion des interruptions.

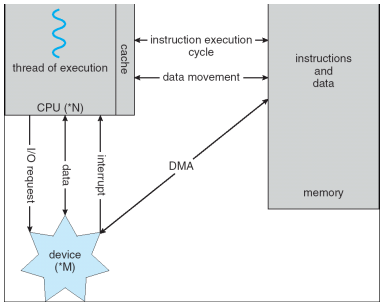
- OS basés sur les interruptions, notamment pour la gestion des périphériques;
- Autre type d'interruption: un *trap* est une interruption générée par un logiciel (erreur ou requête utilisateur);
- Interruption: récupérer l'adresse de la routine d'interruption (vecteur d'interruptions, ou sondage pour trouver le périphérique concerné); sauver les registres et le compteur de programme pour pouvoir reprendre l'exécution après l'interruption;

- Interruptions désactivées lorsqu'on traite une interruption (éviter de perdre une interruption);
- Attente active ou interruption matérielle:



Structure des I/O.

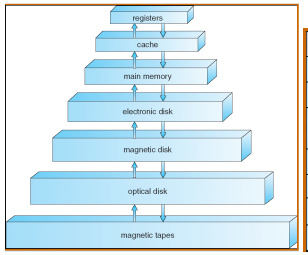
- I/O basées sur les interruptions;
- Un driver de périphérique qui peut parler avec le contrôleur, et qui présente une interface uniforme au reste de l'OS;
- DMA: accès direct mémoire pour les I/O rapides, permet de transmettre les données beaucoup plus rapidement, pas d'intervention du CPU.



Structure du stockage.

- Mémoire: seul média que le CPU peut accéder directement, mais trop petit pour contenir tous les programmes et les données, et volatile;
- Stockage secondaire: extension de la mémoire, non volatile;

- Hiérarchie des périphériques de stockage: plus c'est rapide, plus c'est cher et plus c'est petit!
- L'OS contrôle les mouvements entre mémoire et disque. En dessous, c'est matériel;
- Notion de cache: copier de l'information dans un stockage plus rapide temporairement; politiques de taille de cache et de remplacement des données.

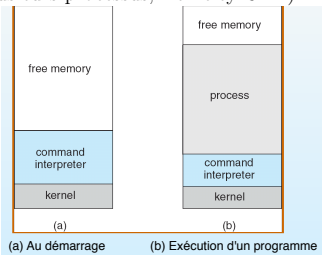


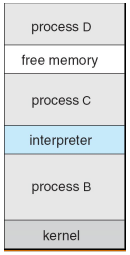
Level	1	2	3	4
Name	registers	cache	main memory	disk storage
Typical size	< 1 KB	> 16 MB	> 16 GB	> 100 GB
Implementation technology	custom memory with multiple ports, CMOS	on-chip or off-chip CMOS SRAM	CMOS DRAM	magnetic disk
Access time (ns)	0.25 – 0.5	0.5 – 25	80 – 250	5,000,000
Bandwidth (MB/sec)	20,000 – 100,000	5000 – 10,000	1000 – 5000	20 – 150
Managed by	compiler	hardware	operating system	operating system
Backed by	cache	main memory	disk	CD or tape

Concepts importants.

- **Multi-programmation:** pour l'efficacité, avoir plusieurs utilisateurs qui se partagent le CPU et les périphériques d'IOs (un utilisateur ne peut pas tout utiliser tout le temps); ordonnancement des tâches: si tâche en cours d'exécution bloquée sur IO, on en choisit une autre.
- **Partage du temps:** extension logique: le CPU passe d'une tâche à l'autre très rapidement, pour avoir une grande interaction; temps de réponse $<$ 1 seconde. Notion de processus, ordonnancement des processus, swapping si tous les processus ne tiennent pas en mémoire, et utilisation de mémoire virtuelle (cf cours mémoire).

Illustration de l'exécution avec MS-DOS (une seule tâche à la fois, le processus écrase une partie du CI, qui sera rechargé sur le disque après exécution du processus), et FreeBSD (plusieurs processus, Berkeley Unix):

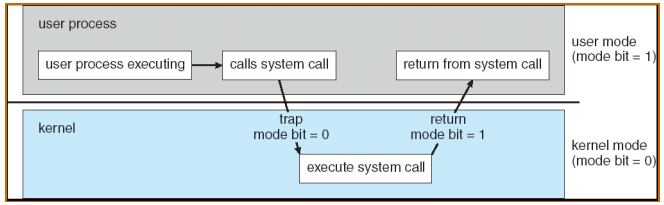




Modes utilisateurs et noyaux.

- Opérations: basées sur les interruptions déjà évoquées;

- Deux modes d'opération pour protéger l'OS: mode utilisateur et mode noyau (kernel);
- Le matériel fournit un bit de mode qui indique si le système tourne en mode U ou K; certaines instructions ne peuvent s'exécuter qu'en mode K (privilégiées);
- Notion d'appels systèmes, qui permet de basculer en mode K;



1.2 Services fournis par l'OS

Gestion des processus. Processus= entité active ($\neq$ programme). Besoin de ressources (CPU, mémoire, IO, fichiers). Un ou plusieurs fils d'exécution (threads), chaque thread a son propre compteur de programme. Multi-programmation: plusieurs processus concurrents.

Responsabilités de l'OS: créer/supprimer processus utilisateur et système; suspendre/reprendre processus; fournir des mécanismes de synchronisation et de communication entre processus, et pour gérer les interblocages.

Cours sur les processus, les threads, la synchronisation, les interblocages, et l'ordonnancement des processus.

Gestion de la mémoire. Il faut que les données/instructions soient en mémoire pour pouvoir exécuter une instruction. Gestion de la mémoire: décider ce qui est en mémoire pour optimiser l'utilisation du CPU et le temps de réponse des utilisateurs.

Responsabilités de l'OS: savoir qui utilise quelles parties de la mémoire; décider quelles données déplacer dans/hors de la mémoire.

Cours mémoire et mémoire virtuelle.

Gestion du stockage et des IOs. Notion de fichier qui donne une vue uniforme et logique de l'information stockée. Fichiers organisés habituellement en répertoires, contrôle d'accès sur les fichiers, et l'OS fournit la possibilité de créer/supprimer les fichiers et répertoires, des primitives pour les manipuler, des techniques de sauvegarde sur stockage stable.

Gestion également des disques secondaires et des opérations associées, tels la gestion des espaces libres, ...

L'OS cherche à cacher les spécificités matérielles de l'utilisateur; sous-système spécifique aux entrées-sorties qui s'occupe de la gestion mémoire des IOs, des interfaces et des drivers.

On n'en discutera pas en détail dans le cours (pas de concepts rigolos).

Services pour l'utilisateur.

- Interface utilisateur → CLI: ligne de commande uniquement (par exemple, rm file.txt), plusieurs "shells"; GUI: interface graphique (icônes qui représentent les fichiers..., utilisation de la souris); Batch: commandes dans un fichier; Systèmes modernes: à la fois CLI et GUI
- Execution de programmes: charger un programme en mémoire, l'exécuter, terminer l'exécution;
- Opérations IO: l'utilisateur ne peut pas directement contrôler un périphérique d'IO, c'est le système qui fournit les IOs;
- Système de fichiers;
- Communications: mémoire partagée ou échange de messages;
- Detection d'erreurs: erreurs matérielles CPU, mémoire, I/O, ou dans le programme utilisateur → s'assurer d'une exécution correcte et consistante; outils de débogage...

Services pour l'efficacité: partage de ressources (plusieurs jobs/utilisateurs).

- Allocation des ressources (cycles CPU, mémoire, fichiers, périphériques IO);
- Comptes (savoir qui utilise quoi en quelle quantité);
- Protection et sécurité (les processus concurrents ne doivent pas interférer entre eux).
  - Protection: mécanisme de contrôle d'accès aux ressources défini par l'OS.
  - Sécurité: permet la défense contre des attaques (internes ou externes, tels des virus, des vols d'identité, ...)
  - Qui peut faire quoi? Notions d'ID d'utilisateur et de groupe associés aux processus et aux fichiers.

1.3 Appels systèmes et programmes systèmes

Interface aux services fournis par l'OS, écrits en langage de haut niveau (C, C++);

API: Application Program Interface. Utilisée par les programmes pour faire des appels systèmes.

Les classiques sont Win32 API (Windows), POSIX API (Unix, Linux, Mac OS X), Java API (Machine virtuelle Java JVM).

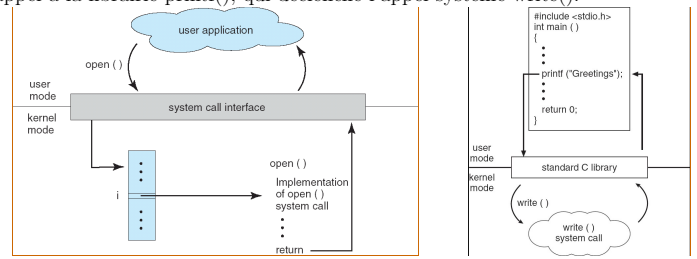
API: portabilité, i.e., un programme qui utilise une API peut tourner sur n'importe quel système qui supporte cette API.

Exemple d'API: lors de la copie d'un fichier, séquence d'appels systèmes dont par exemple l'appel à ReadFile() (Win32 API): description des paramètres à passer, de la valeur de retour.

Implémentation des appels systèmes. Un numéro associé à chaque appel système. Interface d'appels systèmes qui maintient une table indexée par ces numéros.

L'appelant n'a pas besoin de connaître les détails de l'implémentation, mais juste obéir à l'API et comprendre ce que l'OS va faire. TPs: implémentation d'un appel système.

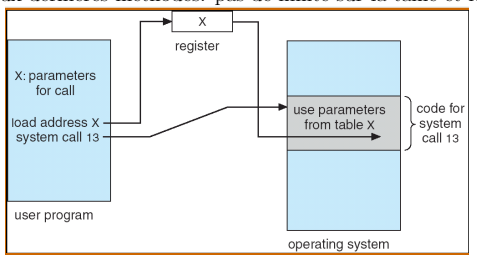
Exemple d'appel système via l'interface ou via une librairie C: le programme C invoque l'appel à la librairie printf(), qui déclenche l'appel système write().



Passage des paramètres: informations autre que le numéro de l'appel système invoqué, comme avec ReadFile. Trois méthodes pour passer les paramètres à l'OS:

1. Les mettre directement dans les registres du CPU;
2. Les stocker dans une table en mémoire, et l'adresse du bloc mémoire est passé en paramètre dans un registre (approche de Linux et Solaris, exemple ci-dessous);
3. Les placer directement sur la pile du programme.

Deux dernières méthodes: pas de limite sur la taille et le nombre des paramètres.



Programmes système. La plupart des utilisateurs voient l'OS comme défini par les programmes systèmes, et non les appels systèmes. Programmes systèmes: environnement commode pour le développement de programmes et leur exécution. Va d'une simple interface d'un appel système à des programmes plus complexes.

Exemple de la gestion de fichiers: programme système pour la copie, qui fait appel à plusieurs appels systèmes (OpenFile, ReadFile, WriteFile, ...), mais aussi information de débogage, éditeurs de texte, commandes pour chercher dans un fichier, compilateurs, assembleurs, charger et exécuter des programmes, communiquer entre processus, envoi de messages, navigateur web, messagerie électronique, ...