Introduction aux systèmes et réseaux informatiques

Sacha Krakowiak
Université Joseph Fourier
Laboratoire Sirac (INPG - INRIA - UJF)

http://sirac.imag.fr/~krakowia

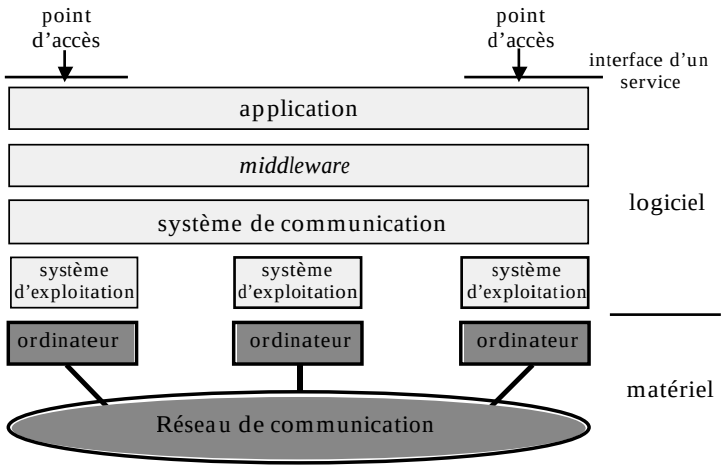
Introduction aux systèmes et réseaux

- **Objectif de ce cours**
  - Présenter les principaux concepts des systèmes d'exploitation, des réseaux et des applications réparties, sans développements techniques détaillés
  - Illustrer concrètement ces concepts au moyen d'exemples pratiques (projet)

- **Plan du cours**
  - Cours 1 à 5 : Introduction aux systèmes d'exploitation
    - Fonctions, organisation
    - Processus, fichiers ; exécution des programmes
  - Cours 6 à 10 : Introduction aux réseaux
    - Protocoles et interfaces, caractéristiques
    - Réseaux locaux
    - Organisation et fonctionnement de l'Internet
  - Cours 11 et 12 : Introduction aux applications réparties
    - Client-serveur, objets répartis

- **TP** (début le 16/1)
  - 4 semaines : Introduction aux systèmes : processus et fichiers en Unix
  - 8 semaines : Introduction aux applications réparties : projet de tableur réparti en Tcl/TK-DP

Les composants d'un système informatique



Interface

- **Un service est caractérisé par son interface**
  - L'interface est l'ensemble des fonctions accessibles aux utilisateurs du service
  - Chaque fonction est définie par
    - son format (la description de son mode d'utilisation) - ou encore sa syntaxe
    - sa spécification (la description de son effet) - ou encore sa sémantique
  - Ces descriptions doivent être
    - précises
    - complètes (y compris les cas d'erreur)
    - non ambiguës
- **Principe de base : séparation entre interface et réalisation**
  - Les descriptions de l'interface d'un service doivent être totalement indépendantes du mode de réalisation du service. Les avantages sont les suivants :
  - Facilite la portabilité
    - transport d'une application utilisant le service sur une autre réalisation du service
    - passage d'un utilisateur sur une autre réalisation du système
  - Permet de remplacer une réalisation du service par une autre, à condition qu'elle réalise la même interface

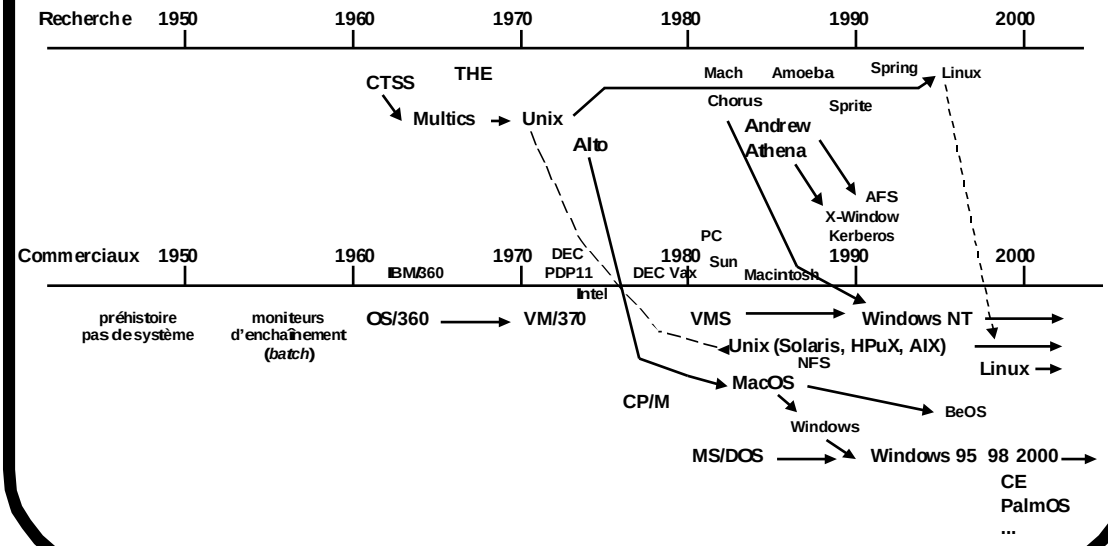
Exemples de services

- **Accès à l'information**
  - Documentation (exemple : bibliothèques virtuelles)
  - Informations (exemple : bulletin météo)
  - Moteur de recherche sur le World Wide Web (Google, AltaVista, Yahoo!, etc.)
- **Communication**
  - Courrier éléctronique (*mail*)
  - Forums de discussion (*news*)
  - Transfert de fichiers (*ftp*)
  - Utilisation d'un ordinateur distant (*telnet*)
- **Commerce électronique**
  - Achat de biens et de services
  - Ventes aux enchères
  - Relations entre entreprises

Rôle d'un système d'exploitation

- **Place**
  - Le système d'exploitation est l'intermédiaire entre un ordinateur (ou en général un appareil muni d'un processeur) et les applications qui utilisent cet ordinateur ou cet appareil. Son rôle peut être vu sous deux aspects complémentaires.
- **Adaptation d'interface**
  - Le système fournit à ses utilisateurs une interface plus commode à utiliser que celle du matériel :
    - il dissimule les détails de mise en œuvre (plus haut niveau d'abstraction)
    - il dissimule les limitations physiques (taille de mémoire, nombre de processeurs) et le partage des ressources entre plusieurs utilisateurs
  - On dit parfois que le système réalise une "machine virtuelle"
- **Gestion de ressources**
  - Le système gère les ressources matérielles et logicielles : mémoire, processeurs, programmes, communications. Cette gestion comprend l'allocation, le partage et la protection.
- **Où trouve-t-on des systèmes d'exploitation ?**
  - sur les ordinateurs : Unix, MacOS, Windows 98, Windows NT, BeOS
  - sur des appareils divers : téléphone portable, assistant personnel, carte à puce.

Historique sommaire des systèmes d'exploitation



Fonctions d'un système d'exploitation

	organe physique	entité virtuelle
■ **Gestion d'activités**		
◆ déroulement de l'exécution	processeur	processus
◆ événements		
■ **Gestion d'information**		
◆ accès dynamique (exécution)	mémoire principale	mémoire virtuelle
◆ conservation permanente de l'information	disque	fichier
◆ partage de l'information		
■ **Gestion des communications**		
◆ interface avec l'utilisateur	écran clavier, souris	fenêtre
◆ impression	imprimante	
◆ réseau	réseau	flot d'entrée-sortie
◆ organes spécialisés	capteurs, scanner, DVD, ...	
■ **Protection**		
◆ de l'information	tous	domaine
◆ des ressources		

Qualités requises d'un système d'exploitation

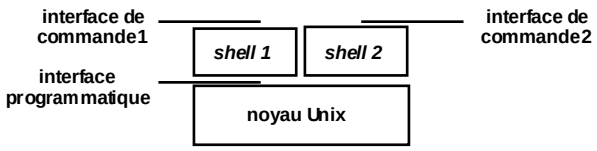
- **Première qualité : "se faire oublier" : la fonction d'un ordinateur est d'excuter les applications, pas le système d'exploitation !**
  - utilisation efficace des ressources
  - fiabilité
  - tolérance aux fautes (du matériel, des utilisateurs, des programmes)
- **Qualité de l'interface (en particulier pour les systèmes interactifs)**
  - convivialité
  - simplicité d'utilisation
  - interface "naturelle" (pas de surprises)
  - documentation en ligne (contextuelle)
- **Bonne intégration au réseau**
- **Sécurité et protection**
- **Répertoire étendu de fonctions**

Interfaces d'un système d'exploitation

Un système d'exploitation présente en général deux interfaces

- **Interface "programmatique", ou API (*Application Programming Interface*)**
  - utilisable à partir des programmes s'exécutant sous le système
  - composée d'un ensemble d'appels systèmes (appels de procédures, avec paramètres)
- **Interface de l'utilisateur, ou interface de commande**
  - utilisable par un usager humain, sous forme textuelle ou graphique
  - composée d'un ensemble de commandes
    - textuelles (exemple en Unix : `rm *.o`)
    - graphiques (exemple : déplacer l'icone d'un fichier vers la corbeille)

Exemple (Unix)



Exemple d'usage des interfaces (Unix)

- **Interface programmatique (en C)**

  le morceau de programme ci-contre utilise les fonctions `read( )` et `write( )` pour recopier un fichier dans un autre

```
...
while (bytesread = read(from_fd, buf, BLKSIZE)) {
    if ((bytesread == -1) && (errno != EINTR))
      break;

    else if (bytesread > 0) {
      bp = buf;
      while(byteswritten = write(to_fd, bp, bytesread)) {
        if ((byteswritten == -1) && (errno != EINTR))
          break;
        else if (byteswritten == bytesread)
          break;
        else if (byteswritten > 0) {
          bp += byteswritten;
          bytesread -= byteswritten;
        }
      }
      if (byteswritten == -1)
        break;
    }
  }
...
```

- **Interface de commande**

  `cp fich1 fich2`

  recopie `fich1` dans `fich2`

- **Documentation**
  - Documentation en ligne par `man`

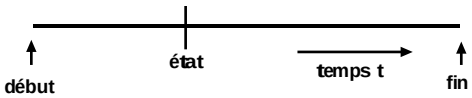
  `man -s 1 <nom de la commande>` : documentation des commandes

  `man -s 2 <nom de la commande>` : documentation des appels système

  par défaut : `man -s 1`

Processus

- **Définition**
  - un processus (séquentiel) est l'entité dynamique représentant l'exécution d'un programme sur un processeur
    - différence entre processus et programme : le programme est une description statique ; le processus est une activité dynamique (il a un début, un déroulement et une fin, il a un état qui évolue au cours du temps)

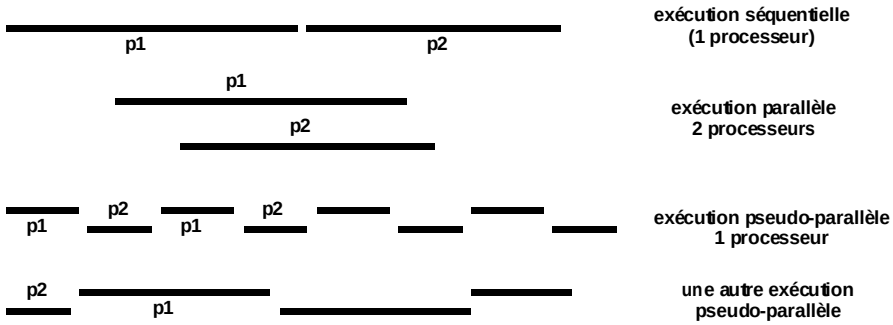


- **Intérêt de la notion de processus**
  - abstraction de la notion d'exécution séquentielle, qui la rend indépendante de la disponibilité effective d'un processeur physique
  - représentation des activités parallèles et de leurs interactions
- **Exemple de processus**
  - l'exécution d'un programme
  - la copie d'un fichier sur disque
  - la transmission d'une séquence de données sur un réseau

Parallélisme et pseudo-parallélisme

- Soit deux processus p1 et p2 (exécution de deux programmes séquentiels P1 et P2

- Mise en œuvre concrète de l'exécution de p1 et p2



Relations entre processus

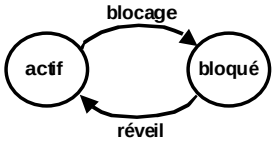
- Compétition
  - Situation dans laquelle plusieurs processus doivent utiliser simultanément une ressource à accès exclusif (ressource ne pouvant être utilisée que par un seul processus à la fois)
  - Exemples
    - processeur (cas du pseudo-parallélisme)
    - imprimante
  - Une solution possible (mais non la seule) : faire attendre les processus demandeurs que l'occupant actuel ait fini (premier arrivé, premier servi)
- Coopération
  - Situation dans laquelle plusieurs processus collaborent à une tâche commune et doivent se synchroniser pour réaliser cette tâche.
  - Exemples
    - p1 produit un fichier, p2 imprime le fichier
    - p1 met à jour un fichier, p2 consulte le fichier
  - La synchronisation se ramène au cas suivant : un processus doit attendre qu'un autre processus ait franchi un certain point de son exécution

Faire attendre un processus

- Dans les deux types de relations (compétition ou coopération), on est conduit à faire attendre un processus. Comment réaliser cette attente ?
- Solution 1 : attente active

p1	p2
while (ressource occupée) { }; ressource occupée = true; ...	ressource occupée = true; utiliser ressource; ressource occupée = false;

  - très peu économique si pseudo-parallélisme
  - difficulté d'une solution correcte (à voir plus tard)
- Solution 2 : blocage du processus
  - On définit un nouvel état pour les processus, l'état bloqué. L'exécution d'un processus bloqué est arrêtée, jusqu'à son réveil explicite par un autre processus



Résumé de la séance 1

- Services et interfaces
- La place et les fonctions du système d'exploitation
  - fournir une interface commode (machine virtuelle)
  - gérer les ressources
    - activités, information, communications; protection
- Les deux interfaces d'un système d'exploitation
  - appels systèmes (pour les programmes
  - commandes (pour les utilisateurs)
- La notion de processus
  - parallélisme et pseudo-parallélisme
  - compétition et coopération
  - faire attendre un processus : blocage

Bibliographie sommaire

A. Tanenbaum, *Systèmes d'exploitation*, Dunod, 1999
J.-M. Rifflet, *La programmation sous Unix*, McGraw-Hill, 3ème édition, 1993