

# ASR5 - Systèmes d'Exploitation

Introduction

Fabien Rico

Séance 1

Jacques BONNEVILLE	jacques.bonneville@univ-lyon1.fr	TP
Adil KHALFA		TD + TP
Léo LE TARO	leo.le-taro@inria.fr	TD + TP
Fabien RICO	fabien.rico@univ-lyon1.fr	CM+ TD + TP

## Table des matières

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Interface avec le matériel</b>	<b>1</b>
<b>3</b>	<b>Organisation</b>	<b>3</b>
<b>4</b>	<b>Sécurité</b>	<b>4</b>
<b>5</b>	<b>Utilisateur</b>	<b>4</b>

## 1 Introduction

### Introduction : Système d'Exploitation

- Qu'est ce que c'est ?
- À quoi ça sert ?
- Comment ça marche ?
- Comment on l'utilise ?

### Qu'est ce que c'est ?

Tout à part :

- les fenêtres, les icônes,
- les applications (*e.g.*, traitement de texte, navigateur internet, le mail)

### Système d'exploitation

Littéralement : « *ce qui permet d'utiliser la machine* » On peut lui donner quatre grands rôles (qui se recourent plus ou moins) :

- Interface entre applications et matériel (*e.g.*, gestion des périphériques)
- Organisation (*e.g.*, des disques, de la mémoire, et des processus)
- Sécurité (*e.g.*, des données, du matériel)
- Interaction avec le ou les utilisateurs (*e.g.*, comptes, droits, installation)

## 2 Interface avec le matériel

### Exemple

Que se passe-t-il lorsque l'on branche une clé USB ?

Le noyau perçoit le nouveau matériel de type usb de stockage et émule un disque scsi





- Vue unifiée du matériel (mémoire, disque, carte réseau, ...)
- Des objets abstraits (fichiers, répertoires, processus, threads, ...)
- Au matériel
  - Gestion des ressources (conflit d'accès, ordonnancement)
  - Protection contre la mauvaise utilisation
  - Une gestion des évènements (interruptions)

Cela impose des vérifications, des files d'attentes et un accès indirect au matériel, donc :

- Au moins 2 niveaux de fonctionnement
  - Utilisateur (exécution par défaut, sans accès)
  - Noyau (exécution protégée)
- Un moyen de passer de l'un à l'autre les *appels systèmes*
- Un mécanisme de déroutement (interruption) et de mise en attente (file de priorité) des évènements asynchrones

### Appel système

#### Un appel système est

- Une fonction fournie par le système
- Que tout programme peut utiliser
- Qui est exécutée en mode noyau

Ce sont des ponts entre le mode utilisateur et le mode noyau.

Par exemple, pour lire, la fonction `scanf` utilise l'appel système `read`.

Les appels systèmes :

- font des vérifications
- sont toujours susceptibles de générer une erreur
- prennent du temps

## 3 Organisation

### Les ressources proposées par la machine

- Un ou plusieurs processeurs capables d'exécuter (ou non) plusieurs choses à la fois
- De la mémoire vive rapide dont les données disparaissent
- De la mémoire de masse plus lente, plus grande et sur laquelle les données sont conservées (les disques et assimilés)
- Des périphériques d'entrées/sorties Clavier, écran, souris, carte réseaux ...

### Les besoins des utilisateurs ou des programmes

- Accès aux ressources
  - (par exemple arbitrage entre les différents programmes qui veulent écrire en même temps sur le disque)
- Organisation des données (comment écrire les données de manière à les retrouver efficacement ?)
- Gestion des évènements (être prévenue d'un évènement par un périphérique, interruption)

### Retour sur l'arbitrage

#### Gérer les ressources demandées par les programmes

- Que se passe-t-il lorsque deux programmes demandent la même chose en même temps ?
- Il faut arbitrer, et se rappeler quel programme / *processus* a obtenu quoi, maintenir une liste de demandes.

### Ressources les plus importantes : le(s) processeur(s) et la mémoire

Le noyau doit décider :

- Quelle tâche devient active. C'est l'*ordonnancement*
- Quelle tâche a accès à la mémoire (ou est stockée sur disque). C'est le *va-et-vient* ou *swap*

**Quand ?**

Pour gérer l'ordonnancement, le noyau doit reprendre la main. Cela a généralement lieu à chaque passage en mode noyau :

- Lors de la gestion des exceptions :
  - division par zéro
  - accès mémoire non autorisé (« Erreur de segmentation »)
  - instruction interdite (« Ce programme va être arrêté car il a effectué une opération non conforme »)
- Lorsqu'une interruption matérielle se produit (IRQ) :
  - en provenance d'un périphérique
  - du timer (quantum de temps)
- Lors des interruptions logicielles via les appels système

⇒ Quand vous écrivez un texte (*printf*), le système en profite pour faire son travail.

## 4 Sécurité

**Sécurité**

Que se passe-t-il si :

1. On fait tellement de calculs que le processeur dépasse 100 degrés ?
2. On interrompt une écriture de disque brutalement ?
3. Le code d'un programmeur maladroit se met à écrire dans les données du LHC (Large Hadron Collider) ?
4. Vous essayez de lire le répertoire `/home/frico/SujetsExam/` ?

Puisque le système est une interface entre les applications et le matériel, il a aussi un rôle de protection :

- Du matériel
  - Monitoring *e.g.*, `/proc/acpi/thermal_zone/THM/temperature`
  - Actions automatiques (*e.g.*, gestion de l'énergie)
  - Zones critiques. Certaines actions, comme par exemple le pilotage du disque dur ne doivent pas être interrompues
- Des données
  - Systèmes de fichiers journalisés (protection contre l'arrêt brutal)
  - Utilisateur, droits, authentification
- Des programmes
  - Séparation des tâches. Le système fait en sorte qu'un programme ne puisse normalement pas perturber le fonctionnement d'un autre.
  - Virtualisation pour assurer cette protection, un programme est exécuté dans un environnement protégé avec un jeu d'instructions réduit, il n'accède pas directement à la mémoire ...
  - Communication à cause de cette protection il faut fournir méthodes fiables de communication entre programmes.

## 5 Utilisateur

**Que voit l'utilisateur ?**

- Le logo au démarrage, et quelques bizarreries



- Un système de configuration
- Un système d'installation

### Pourquoi étudier le système

- Le système impose des limites
  - Droits d'accès (site web, installation XP ou Vista)
  - Système de fichiers (clé USB, racine)
- Programmation « avancée » (*e.g.*, client/serveur, multi-thread)
- Administration

Le système idéal selon la publicité devrait être à la fois efficace et invisible, c'est-à-dire faire exactement ce qu'on veut, au moment où on le veut, sans n'avoir rien à savoir de son fonctionnement. C'est bien entendu impossible. Surtout pour les informaticiens qui doivent aller plus loin (programmation multithread, programmation réseau, configuration de machines *etc.*).

De plus, les techniques utilisées pour « simplifier » l'utilisation du système ne simplifient que les utilisations les plus courantes, mais elles compliquent les utilisations plus spécifiques car :

- Les interfaces de configurations sont incomplètes et très instables (*e.g.*, assistant).
- Le système fait un grand nombre d'actions de façon implicite qui perturbent le fonctionnement (*e.g.*, mise en cache)
- En cas de mauvaise utilisation, le système met en place des méthodes de secours en mode dégradé ce qui empêche de voir les erreurs

Le système doit

- Différencier les utilisateurs
  - système d'authentification
  - base de données des utilisateurs
- Être configurable
  - interface de configuration
  - base de données des configurations
- Avoir un système d'installation de programmes
  - comment installer ?
  - notion de packages

### Objectifs de l'UE

Le système d'exploitation peut être abordé selon 3 points de vue :

- Conception et théorie : les problèmes posés par les systèmes et les moyens de les résoudre.
- Utilisation et programmation : les outils fournis par les systèmes pour mieux utiliser les possibilités des ordinateurs (programmation multiprocesseur, multithread, réseau par exemple)
- Administration : comment configurer et gérer le système ?

### Bibliographie

#### Les livres

- Andrew TANENBAUM. *Systèmes d'exploitation*. Pearson, 2008.
- Joffroy BEAUQUIER et Béatrice BÉRARD. *Systèmes d'exploitation*. McGraw-Hill, 1991.

#### Les sites

- F. PELLEGRINI et D. SHERMAN. « *Système d'exploitation* ». ENSEIRB, <http://uuu.enseirb.fr/~pelegrin/enseignement/>, 2001.
- D. REVUZ. « *Cours Système* ». Université de Marne-la-vallée, <http://www-igm.univ-mlv.fr/~dr/NCS/>, feb 2005.
- Cyril DROCOURT. « *Programmation Systeme* ». IUT d'Amiens, <http://info.iut-amiens.fr/~drocourt/cours/>.
- David DECOTIGNY et Thomas PETAZZONI. « *SimpleOS* ». <http://sos.enix.org/fr/PagePrincipale>, Gnu Linux Magazine - diamond editions, 2004-2007.

### En conclusion

#### À retenir

- Rôle d'interface du système
- Notion de niveaux de fonctionnement (utilisateur ou noyau)
- Appels systèmes et leur rôle

