

# Systèmes d'exploitation II

## Réseaux et Système Avancés

Moufida Maimour <moufida.maimour@telecomnancy.eu>

TELECOM Nancy – 2ème année

2021-2022

# Conception des systèmes d'exploitation

## (SE ou OS, Operating System)

- ▶ Concepts et algorithmes généraux
- ▶ Etudes de cas (Linux 2.6, 4.4BSD, Système V, SOLARIS)

### Objectifs

- ▶ Comprendre les principes de conception et la structure d'un SE
- ▶ Identifier quelques problèmes à résoudre lors de la conception des SE
- ▶ Comparer et évaluer les solutions proposées à ces problèmes

### Pré-requis :

- ▶ ~~Notions de base sur l'architecture des ordinateurs (PFSI)~~ → ASM ?,
- ▶ Pratique du langage C et du shell UNIX (CSH),
- ▶ Programmation système (RS)

### Évaluation

- ▶ Un écrit à la fin du module
- ▶ Des questionnaires en CM.

# Contenu du module

## 1 Généralités : systèmes informatiques et systèmes d'exploitation

- ▶ Rappel et compléments sur la structure générale des systèmes informatiques, quelques aspects de l'architecture du Pentium ;
- ▶ composants d'un OS, conception d'un OS, organisation générale du noyau Unix.

## 2 Gestion de la mémoire

concepts, impact de la liaison d'adresses, mémoire uniforme et hiérarchisée, le cas linux 2.6.

## 3 Gestion des processus et ordonnancement

espace d'adressage, implémentation des processus, changement de contexte, algorithmes d'ordonnancement, les cas SOLARIS, HP-UX, 4.4BSD, Linux 2.6.

# Bibliographie succincte

- ▶ A. Silberschatz, P. B. Galvin and G. Gagne : [Operating Systems Concepts](#) [www.os-book.com](http://www.os-book.com) (2018).<sup>1</sup>
- ▶ Andrew S Tanenbaum and Herbert Bos : [Modern Operating Systems : Global Edition](#) (2015)
  
- ▶ D. P. Bovet and M. Cesati : [Understanding the Linux Kernel](#) (3rd edition)
- ▶ M. J. Bach : [The Design of the Unix Operating System](#) (Prentice Hall)
- ▶ M. K. McKusik, K. Bostic, M. J. Karels et J. S. Quarterman : [Conception et Implémentation du Système 4.4BSD](#)

---

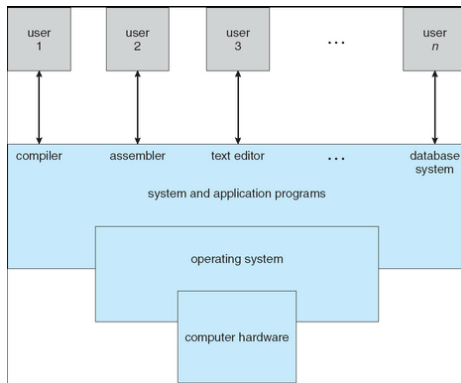
1. Source de nombreuses figures de ce document

# Première partie

## Généralités

- Structure générale d'un système informatique
- Cas du Pentium
- Composants d'un système d'exploitation
  - Système de protection
  - Système d'E/S
- Conception des systèmes d'exploitations
  - Le noyau
    - Structure simple : MS-DOS
    - Structure en couches
    - Systèmes monolithiques
    - Micro-noyau
    - Modules et noyaux modulaires
- Organisation générale du noyau Unix
  - Architecture du noyau Unix
  - Invocation des services système

# Système informatique

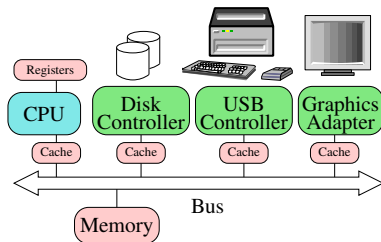


- ▶ **Matériel** : CPU, mémoire, périphériques
- ▶ **OS** : permet l'utilisation du matériel en optimisant ses ressources
- ▶ **Applications** : éditeurs de texte, compilateurs, base de données, jeux video ...
- ▶ **Utilisateurs** : humains, machines, d'autres ordinateurs

# Système informatique : le matériel

## Système informatique moderne

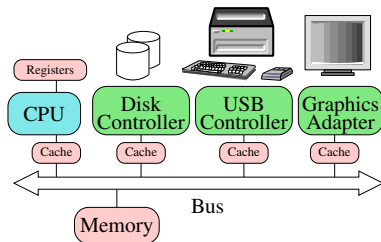
- ▶ CPU (Central Processing Unit)
- ▶ Structures de stockage
- ▶ Périphériques d'E/S
- ▶ Communication via un **bus**



# Système informatique : le matériel

## Système informatique moderne

- ▶ CPU (Central Processing Unit)
- ▶ Structures de stockage
- ▶ Périphériques d'E/S
- ▶ Communication via un **bus**



## Sous-ensembles du bus

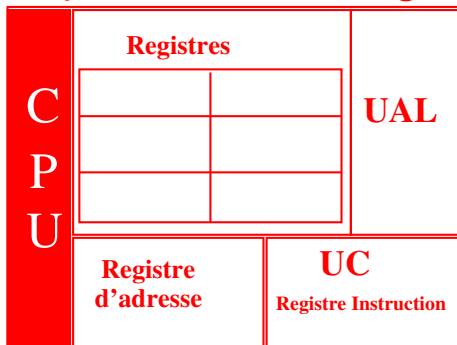
- ▶ **Bus de données** : véhicule les données
- ▶ **Bus d'adresse** : véhicule des adresses et conditionne la taille mémoire adressable
- ▶ **Bus de commande** : véhicule tous les signaux utilisés pour synchroniser les différentes activités : signaux d'horloge, R/W, interruptions ...

**Pentium** : 64 lignes de données (E/S), 32 lignes d'adresses et quelques lignes de contrôle



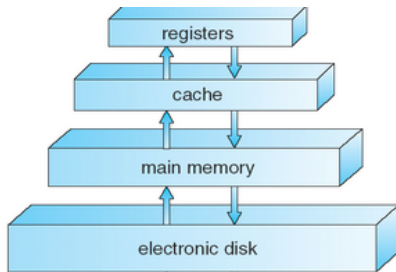
# Système informatique : le matériel

## CPU (Central Processing Unit)



- ▶ **ALU** : opérations arithmétiques et logiques
- ▶ **Registres** : de données, d'adresse, SP, PC, ...
- ▶ **Unité de contrôle (UC)** : A chaque coup d'horloge, elle ouvre et ferme des portes dans le but d'alimenter l'UAL en données et instructions et d'interdire la collision d'informations.

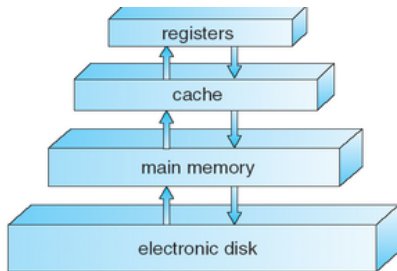
## Hiérarchie de stockage



- ▶ taille,
- ▶ vitesse,
- ▶ coût,
- ▶ volatilité

# Système informatique : le matériel

## Hiérarchie de stockage



- taille,
- vitesse,
- coût,
- volatilité

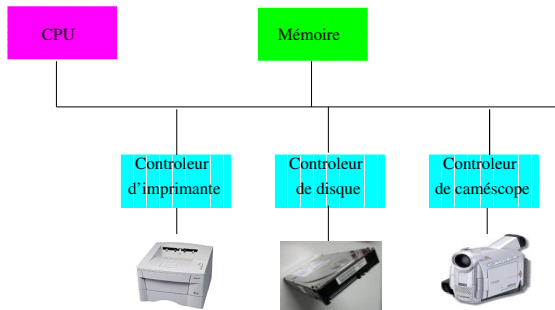
Level	Registers	Cache	Main Memory	Disk storage
Typical size	<1Kb	few Mb	few Gb	100s of Gb
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	5,000,000
Bandwidth (Mb/s)	20k - 100k	5k - 10k	1k - 5k	20 - 150
Volatile ?	Yes	Yes	Yes	No
Managed by	Compiler	Hardware	OS	OS
Backed by	Cache	Main Mem	Disk	CD or tape

# Système informatique : le matériel

## Périphériques d'E/S : device controllers

**Device controller** : la partie électronique du périphérique. Elle maintient un ensemble de registres :

- ▶ command registers (write-only) ;
- ▶ status registers (read-only) ;
- ▶ data registers (read/write).



CPU peut accéder à ces registres via des instructions d'E/S (in/out dans Intel Asm) ou via un mapping mémoire.

## Les registres généraux

General-Purpose Registers							
31	16	15	8	7	0	16-bit	32-bit
			AH		AL	AX	EAX
			BH		BL	BX	EBX
			CH		CL	CX	ECX
			DH		DL	DX	EDX
			BP				EBP
			SI				ESI
			DI				EDI
			SP				ESP

- ▶ les registres 8 bits datent du 8088 ;
- ▶ les registres 16 bits (AX) remontent au vieux 80286 ;
- ▶ les registres 32 bits (EAX) n'existent qu'à partir du 80386.

## Des registres spécialisés

### Le registre d'état EFLAGS

- ▶ indique l'état d'exécution du processeur et la nature du résultat d'une opération arithmétique ou logique ;
- ▶ permet la réalisation des instructions de branchement ;
- ▶ certains bits (ex. bit d'interruption) permet au programmeur de déterminer le comportement du processeur face à certains événements.

## Des registres spécialisés

### Le registre d'état EFLAGS

- ▶ indique l'état d'exécution du processeur et la nature du résultat d'une opération arithmétique ou logique ;
- ▶ permet la réalisation des instructions de branchement ;
- ▶ certains bits (ex. bit d'interruption) permet au programmeur de déterminer le comportement du processeur face à certains événements.

### Les registres de segments

- ▶ **cs (code segment)**, pointe sur le segment contenant les instructions du programme
- ▶ **ss (stack segment)**, pointe sur le segment contenant la pile du programme : gestion des sous-programmes et des interruptions.
- ▶ **ds (data segment)**, pointe sur le segment contenant les données globales et statiques du programme.

## Des registres spécialisés

### Le registre d'état EFLAGS

- ▶ indique l'état d'exécution du processeur et la nature du résultat d'une opération arithmétique ou logique ;
- ▶ permet la réalisation des instructions de branchement ;
- ▶ certains bits (ex. bit d'interruption) permet au programmeur de déterminer le comportement du processeur face à certains événements.

### Les registres de segments

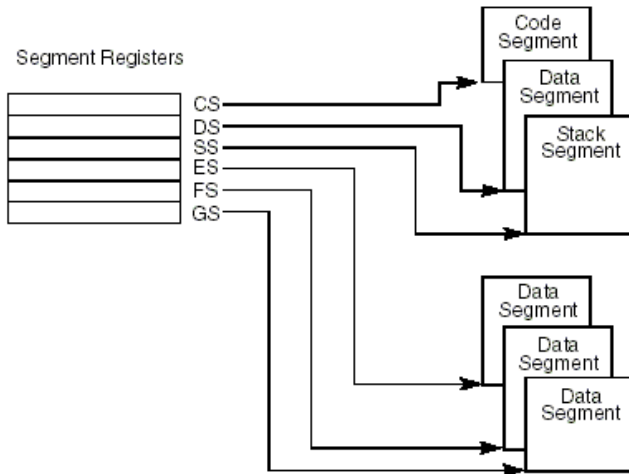
- ▶ **cs (code segment)**, pointe sur le segment contenant les instructions du programme
- ▶ **ss (stack segment)**, pointe sur le segment contenant la pile du programme : gestion des sous-programmes et des interruptions.
- ▶ **ds (data segment)**, pointe sur le segment contenant les données globales et statiques du programme.

Des registres pour la gestion de la mémoire virtuelle seront vus plus tard.



## Les registres de segments

Il y a 3 autres registres plus généraux (ES,FS,GS).



## Principaux modes d'adressage en assembleur

### ► Adressage Registre

```
addl %eax, %edx
```

Ajouter le contenu de EAX à celui de EDX. Résultat dans EDX

## Principaux modes d'adressage en assembleur

### ► Adressage Registre

`addl %eax, %edx`

Ajouter le contenu de EAX à celui de EDX. Résultat dans EDX

### ► Adressage immédiat

`addl $10, %eax`

Ajouter 10 à la valeur contenue dans EAX. Résultat dans EAX

## Principaux modes d'adressage en assembleur

### ► Adressage Registre

`addl %eax, %edx`

Ajouter le contenu de EAX à celui de EDX. Résultat dans EDX

### ► Adressage immédiat

`addl $10, %eax`

Ajouter 10 à la valeur contenue dans EAX. Résultat dans EAX

### ► Adressage indirect

`addl %edx, (%eax)`

EAX contient l'adresse de la valeur à ajouter au contenu de EDX. Résultat dans la case mémoire pointée par EAX.

# Questionnaire 1.1

Durée : 5 minutes

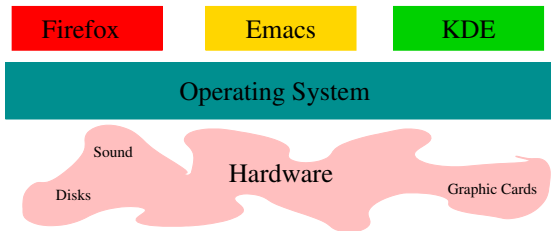
# Première partie

## Généralités

- Structure générale d'un système informatique
- Cas du Pentium
- Composants d'un système d'exploitation
  - Système de protection
  - Système d'E/S
- Conception des systèmes d'exploitations
  - Le noyau
    - Structure simple : MS-DOS
    - Structure en couches
    - Systèmes monolithiques
    - Micro-noyau
    - Modules et noyaux modulaires
- Organisation générale du noyau Unix
  - Architecture du noyau Unix
  - Invocation des services système

# C'est quoi un système d'exploitation

Un **système d'exploitation** est l'ensemble des programmes qui permettent l'exploitation du matériel :

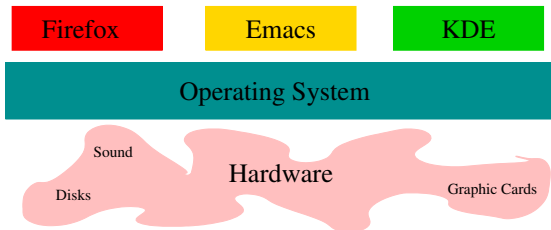


# C'est quoi un système d'exploitation

Un **système d'exploitation** est l'ensemble des programmes qui permettent l'exploitation du matériel :

## Allocateur de ressources

- Gestion de toutes les ressources
- Gestion des conflits pour une utilisation efficace et équitable





# C'est quoi un système d'exploitation

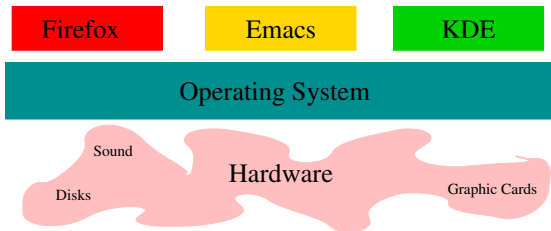
Un **système d'exploitation** est l'ensemble des programmes qui permettent l'exploitation du matériel :

## Allocateur de ressources

- ▶ Gestion de toutes les ressources
- ▶ Gestion des conflits pour une utilisation efficace et équitable

## Contrôleur

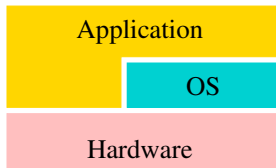
- ▶ Contrôle l'exécution des programmes pour éviter les erreurs et l'utilisation incorrecte de l'ordinateur



# Evolution des SE

## Systèmes monoprogrammés (monojob)

- ▶ Un seul programme, un seul utilisateur
- ▶ MS-DOS, systèmes embarqués
- ▶ **Problème** : usage inefficace des ressources matérielles. Pas de recouvrement des E/S.



# Evolution des SE

## Systèmes monoprogrammés (monojob)

- ▶ Un seul programme, un seul utilisateur
- ▶ MS-DOS, systèmes embarqués
- ▶ **Problème** : usage inefficace des ressources matérielles. Pas de recouvrement des E/S.

## Solution

- ▶ Permettre la coexistence de plusieurs programmes dans le système



# Evolution des SE

## Systèmes multiprogrammés (multijob)

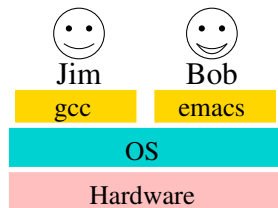
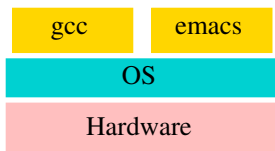
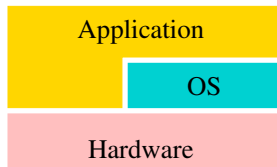
- ▶ Plusieurs programmes peuvent être chargés en mémoire en même temps  
Si un processus se bloque (E/S), un autre pourra se voir attribué le processeur pour être exécuté
- ▶ Problèmes liés à la concurrence entre processus sont à résoudre



# Evolution des SE

## Systèmes multi-utilisateur (multiuser)

- ▶ apparus avec l'arrivée des terminaux
- ▶ Plusieurs utilisateurs sont autorisés à lancer leurs programmes sur la même machine.
- ▶ Problèmes liés à la **gestion** des utilisateurs sont à résoudre



## **Système interpréteur de commandes**

- ▶ interface entre l'utilisateur et le SE.
- ▶ inclu au noyau ou un programme spécial (Unix, MS-DOS)
- ▶ textuel (Unix) ou graphique (Windows)

# Composants des systèmes d'exploitation

## Système interpréteur de commandes

- ▶ interface entre l'utilisateur et le SE.
- ▶ inclu au noyau ou un programme spécial (Unix, MS-DOS)
- ▶ textuel (Unix) ou graphique (Windows)

## Gestion des processus

- ▶ création (**fork**, **exec**) et terminaison (**exit**, **kill**) de processus système et utilisateur
- ▶ suspension et reprise des processus (**wait**, **waitpid**, **sleep**, **pause**)
- ▶ fournir les mécanismes de synchronisation et communication de processus (**signaux**, **tubes**, **sémaphores**, **mémoire partagée**, ...)
- ▶ fournir des mécanismes pour le traitement des interblocages

## Gestion de la mémoire principale

- ▶ Pour être exécuté, un programme doit être chargé en mémoire.
- ▶ La multiprogrammation nécessite une gestion plus avancée de la mémoire afin de permettre la coexistence de plusieurs processus.



## Gestion de la mémoire principale

- ▶ Pour être exécuté, un programme doit être chargé en mémoire.
- ▶ La multiprogrammation nécessite une gestion plus avancée de la mémoire afin de permettre la coexistence de plusieurs processus.

### Le système d'exploitation doit :

- ▶ savoir en permanence, quelles parties de la mémoire sont en cours d'utilisation et par qui ;
- ▶ décider de quels processus doivent être chargés en mémoire lorsque l'on dispose d'espace mémoire ;
- ▶ affecter et désaffecter de l'espace mémoire.

# Composants des systèmes d'exploitation

---

## Gestion des fichiers

l'une des composantes les plus visibles d'un système d'exploitation. Elle fournit :

- ▶ une vue logique uniforme du contenu des différents dispositifs de stockages (fichier, répertoire)

# Composants des systèmes d'exploitation

---

## Gestion des fichiers

l'une des composantes les plus visibles d'un système d'exploitation. Elle fournit :

- ▶ une vue logique uniforme du contenu des différents dispositifs de stockages (**fichier, répertoire**)
- ▶ la correspondance entre les fichiers (concept logique) et la mémoire auxiliaire (physique)

## Gestion des fichiers

l'une des composantes les plus visibles d'un système d'exploitation. Elle fournit :

- ▶ une vue logique uniforme du contenu des différents dispositifs de stockages (fichier, répertoire)
- ▶ la correspondance entre les fichiers (concept logique) et la mémoire auxiliaire (physique)
- ▶ la possibilité de sauvegarde des fichiers sur des supports d'information stables

## Gestion des fichiers

l'une des composantes les plus visibles d'un système d'exploitation. Elle fournit :

- ▶ une vue logique uniforme du contenu des différents dispositifs de stockages (fichier, répertoire)
- ▶ la correspondance entre les fichiers (concept logique) et la mémoire auxiliaire (physique)
- ▶ la possibilité de sauvegarde des fichiers sur des supports d'information stables
- ▶ des primitives de :

# Composants des systèmes d'exploitation

---

## Gestion des fichiers

l'une des composantes les plus visibles d'un système d'exploitation. Elle fournit :

- ▶ une vue logique uniforme du contenu des différents dispositifs de stockages (**fichier, répertoire**)
- ▶ la correspondance entre les fichiers (concept logique) et la mémoire auxiliaire (physique)
- ▶ la possibilité de sauvegarde des fichiers sur des supports d'information stables
- ▶ des primitives de :
  - ▶ création et suppression de fichiers (répertoires) : **creat, mkdir, rmdir, ...**

# Composants des systèmes d'exploitation

## Gestion des fichiers

l'une des composantes les plus visibles d'un système d'exploitation. Elle fournit :

- ▶ une vue logique uniforme du contenu des différents dispositifs de stockages (**fichier, répertoire**)
- ▶ la correspondance entre les fichiers (concept logique) et la mémoire auxiliaire (physique)
- ▶ la possibilité de sauvegarde des fichiers sur des supports d'information stables
- ▶ des primitives de :
  - ▶ création et suppression de fichiers (répertoires) : **creat, mkdir, rmdir, ...**
  - ▶ manipulation de fichiers (répertoires) : **open, read, write, lseek, close, stat, opendir, readdir, closedir, ...**

# Composants des systèmes d'exploitation

---

## Système de protection

Un système d'exploitation (multiprogrammé en particulier) dispose de ressources matérielles et logicielles qui doivent être protégées :



## Système de protection

Un système d'exploitation (multiprogrammé en particulier) dispose de ressources matérielles et logicielles qui doivent être protégées :

- ▶ Protection **logicielle** (système de fichiers)
- ▶ Protection **matérielle** (mémoire, temps CPU, périphérique d'E/S, ...)

# Composants des systèmes d'exploitation

---

## Système de protection

Un système d'exploitation (multiprogrammé en particulier) dispose de ressources matérielles et logicielles qui doivent être protégées :

- ▶ Protection **logicielle** (système de fichiers)
- ▶ Protection **matérielle** (mémoire, temps CPU, périphérique d'E/S, ...)

**La protection matérielle** : le matériel fournit des mécanismes matériels au niveau du processeur permettant :

# Composants des systèmes d'exploitation

## Système de protection

Un système d'exploitation (multiprogrammé en particulier) dispose de ressources matérielles et logicielles qui doivent être protégées :

- ▶ Protection **logicielle** (système de fichiers)
- ▶ Protection **matérielle** (mémoire, temps CPU, périphérique d'E/S, ...)

**La protection matérielle** : le matériel fournit des mécanismes matériels au niveau du processeur permettant :

- ▶ protection **intra-processus** séparant le **mode d'exécution utilisateur** du **mode d'exécution privilégié**  
⇒ **Appels système** pour l'exécution des instructions privilégiées

# Composants des systèmes d'exploitation

## Système de protection

Un système d'exploitation (multiprogrammé en particulier) dispose de ressources matérielles et logicielles qui doivent être protégées :

- ▶ Protection **logicielle** (système de fichiers)
- ▶ Protection **matérielle** (mémoire, temps CPU, périphérique d'E/S, ...)

**La protection matérielle** : le matériel fournit des mécanismes matériels au niveau du processeur permettant :

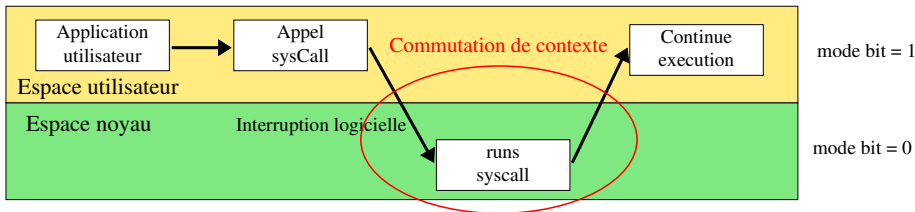
- ▶ protection **intra-processus** séparant le **mode d'exécution utilisateur** du **mode d'exécution privilégié**  
⇒ **Appels système** pour l'exécution des instructions privilégiées
- ▶ protection **inter-processus** garantissant le fonctionnement multi-programmé  
⇒ protection des espaces d'adressage des processus utilisateur et noyau

# Système de protection

## Protection matérielle : Modes d'exécution

Un processeur actuel dispose d'au moins 2 modes d'exécution :

- ▶ mode privilégié (superviseur) associé au mode **kernel** (noyau) des systèmes Unix,
- ▶ mode non privilégié associé au mode **user** (utilisateur) des systèmes Unix

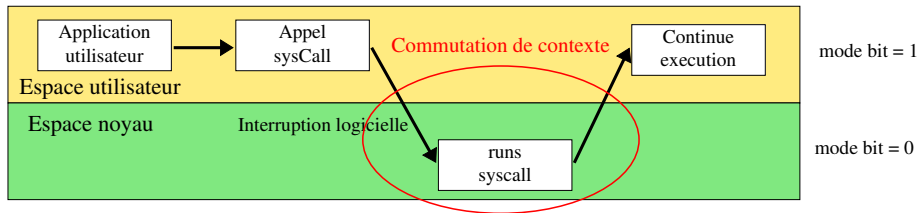


# Système de protection

## Protection matérielle : Modes d'exécution

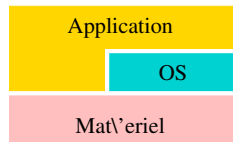
Un processeur actuel dispose d'au moins 2 modes d'exécution :

- ▶ mode privilégié (superviseur) associé au mode **kernel** (noyau) des systèmes Unix,
- ▶ mode non privilégié associé au mode **user** (utilisateur) des systèmes Unix



Intel 80486 dispose de 4 niveaux d'exécutions :

- ▶ MS-Dos écrit pour 8088 ne possède pas de mode protégé. Tous les programmes utilisateur ont accès direct au matériel
- ▶ Les SE les plus récents comme Windows/Nt (Microsoft) et OS/2 (IBM) fournissent une meilleure protection



## Protection des E/S - Appels système

Les instructions d'E/S sont considérées comme privilégiées. Un programme utilisateur doit passer par le système d'exploitation pour effectuer des E/S

## Protection des E/S - Appels système

Les instructions d'E/S sont considérées comme privilégiées. Un programme utilisateur doit passer par le système d'exploitation pour effectuer des E/S

- ▶ Un appel système est déclenché par un programme (événement **synchrone**) afin de réaliser une opération en mode privilégie.
- ▶ Il peut être vu comme un appel d'une fonction classique effectuée par le SE (en mode noyau) pour le compte d'un processus utilisateur.



## Protection des E/S - Appels système

Les instructions d'E/S sont considérées comme privilégiées. Un programme utilisateur doit passer par le système d'exploitation pour effectuer des E/S

- ▶ Un appel système est déclenché par un programme (événement **synchrone**) afin de réaliser une opération en mode privilégie.
- ▶ Il peut être vu comme un appel d'une fonction classique effectuée par le SE (en mode noyau) pour le compte d'un processus utilisateur.
- ▶ L'interface entre le SE et les programmes utilisateurs est définie par l'ensemble des appels système fournis par ce dernier.

## Protection des E/S - Appels système

Les instructions d'E/S sont considérées comme privilégiées. Un programme utilisateur doit passer par le système d'exploitation pour effectuer des E/S

- ▶ Un appel système est déclenché par un programme (événement **synchrone**) afin de réaliser une opération en mode privilégie.
- ▶ Il peut être vu comme un appel d'une fonction classique effectuée par le SE (en mode noyau) pour le compte d'un processus utilisateur.
- ▶ L'interface entre le SE et les programmes utilisateurs est définie par l'ensemble des appels système fournis par ce dernier.
- ▶ Un appel système est généralement réalisé par une **interruption logicielle** déclenchée par un programme à l'aide d'une instruction spéciale (**trap**, **syscall**).

## Protection des E/S - Appels système

Les instructions d'E/S sont considérées comme privilégiées. Un programme utilisateur doit passer par le système d'exploitation pour effectuer des E/S

- ▶ Un appel système est déclenché par un programme (événement **synchrone**) afin de réaliser une opération en mode privilégie.
- ▶ Il peut être vu comme un appel d'une fonction classique effectuée par le SE (en mode noyau) pour le compte d'un processus utilisateur.
- ▶ L'interface entre le SE et les programmes utilisateurs est définie par l'ensemble des appels système fournis par ce dernier.
- ▶ Un appel système est généralement réalisé par une **interruption logicielle** déclenchée par un programme à l'aide d'une instruction spéciale (**trap**, **syscall**).
- ▶ Des informations nécessaires à la requête peuvent être passées par registres, pile ou mémoire.

## Protection de la mémoire centrale

- ▶ Séparation de l'espace d'adressage des différents processus utilisateur ainsi que le noyau

# Système de protection

## Protection de la mémoire centrale

- ▶ Séparation de l'espace d'adressage des différents processus utilisateur ainsi que le noyau
- ▶ Mise en œuvre matérielle :

# Système de protection

## Protection de la mémoire centrale

- ▶ Séparation de l'espace d'adressage des différents processus utilisateur ainsi que le noyau
- ▶ Mise en œuvre matérielle :
  - ▶ utilisation de 2 registres **base** et **limite** qui déterminent la fourchette d'adresse auxquelles un programme peut y accéder

# Système de protection

## Protection de la mémoire centrale

- ▶ Séparation de l'espace d'adressage des différents processus utilisateur ainsi que le noyau
- ▶ Mise en œuvre matérielle :
  - ▶ utilisation de 2 registres **base** et **limite** qui déterminent la fourchette d'adresse auxquelles un programme peut y accéder
  - ▶ seul le SE peut les charger en utilisant une instruction privilégiée spéciale.

# Système de protection

## Protection de la mémoire centrale

- ▶ Séparation de l'espace d'adressage des différents processus utilisateur ainsi que le noyau
- ▶ Mise en œuvre matérielle :
  - ▶ utilisation de 2 registres **base** et **limite** qui déterminent la fourchette d'adresse auxquelles un programme peut y accéder
  - ▶ seul le SE peut les charger en utilisant une instruction privilégiée spéciale.
  - ▶ une tentative d'accès en dehors de cet espace par le programme en mode utilisateur provoque **une exception**.



# Système de protection

## Protection de la mémoire centrale

- ▶ Séparation de l'espace d'adressage des différents processus utilisateur ainsi que le noyau
- ▶ Mise en œuvre matérielle :
  - ▶ utilisation de 2 registres **base** et **limite** qui déterminent la fourchette d'adresse auxquelles un programme peut y accéder
  - ▶ seul le SE peut les charger en utilisant une instruction privilégiée spéciale.
  - ▶ une tentative d'accès en dehors de cet espace par le programme en mode utilisateur provoque **une exception**.

Une **exception** est un signal provoqué par un dysfonctionnement du programme en cours d'exécution : division par zéro, faute de page ...

- ▶ **80x86** : 20 différentes exceptions 0..19.

Chaque exception dispose d'un sous-programme (handler) qui prend en charge l'évènement correspondant provoquant :

- ▶ l'arrêt du programme
- ▶ ou la **réexécution** de l'instruction en cause.

## Protection de l'UC

- ▶ S'assurer que le SE garde le contrôle. Exemple, éviter qu'un programme en boucle infinie ne rende jamais la main
- ▶ Solution : utilisation d'une horloge via une interruption (temps partagé)

# Système de protection

## Protection de l'UC

- ▶ S'assurer que le SE garde le contrôle. Exemple, éviter qu'un programme en boucle infinie ne rende jamais la main
- ▶ Solution : utilisation d'une horloge via une interruption (temps partagé)

**Une interruption** est provoquée par un signal provenant du monde extérieur au processeur, et modifiant le comportement de celui-ci. Le but est de le prévenir de l'occurrence d'un évènement extérieur :

- ▶ fin d'une E/S, top d'horloge ...
- ▶ **80x86** : 32-255. Linux utilise le vecteur 128 (0x80) pour les appels système.

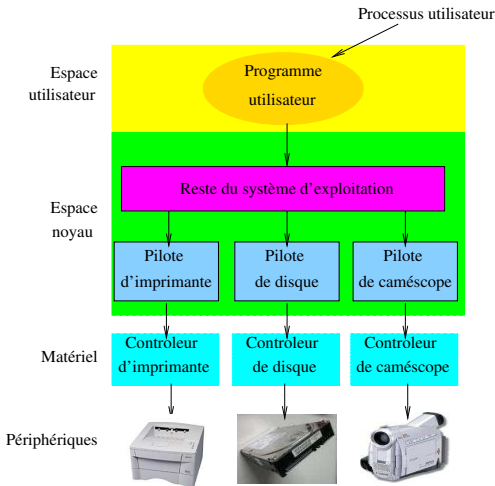
Chaque interruption (comme pour les exceptions) dispose d'un **handler** qui prend en charge l'évènement correspondant : **table de vecteurs d'interruptions** ou **IDT (Interrupt Descriptor Table)** dans le langage Linux.

# Questionnaire 1.2

Durée : 10 minutes

# Composants des systèmes d'exploitation

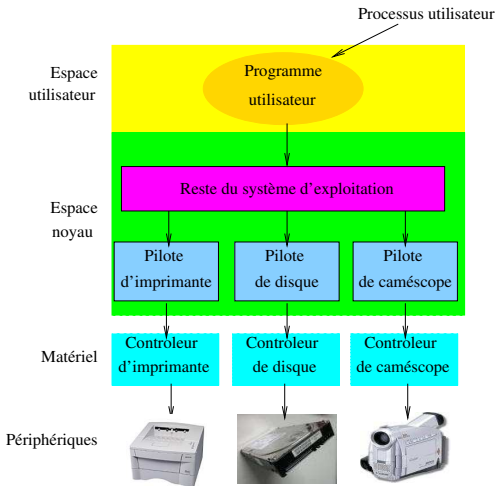
## Gestion du système d'E/S



- Contrôler tous les périphériques d'E/S de l'ordinateur

# Composants des systèmes d'exploitation

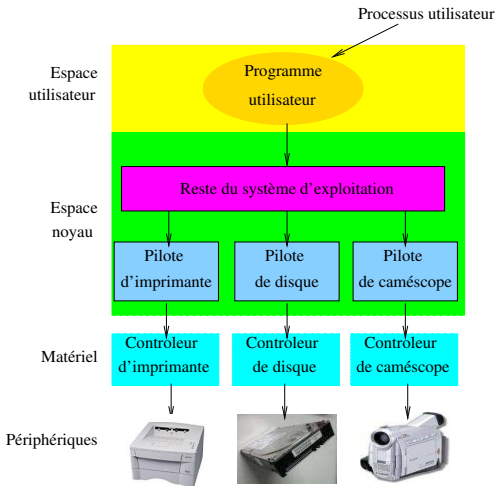
## Gestion du système d'E/S



- ▶ Contrôler tous les périphériques d'E/S de l'ordinateur
- ▶ Fournir une interface entre les périphériques et le reste du système

# Composants des systèmes d'exploitation

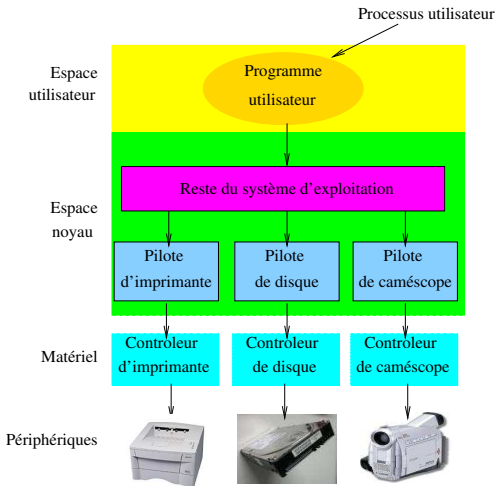
## Gestion du système d'E/S



- ▶ Contrôler tous les périphériques d'E/S de l'ordinateur
- ▶ Fournir une interface entre les périphériques et le reste du système
  - ▶ indépendance du matériel : **sort < in > out**

# Composants des systèmes d'exploitation

## Gestion du système d'E/S



- ▶ Contrôler tous les périphériques d'E/S de l'ordinateur
- ▶ Fournir une interface entre les périphériques et le reste du système
  - ▶ indépendance du matériel : **sort < in > out**
  - ▶ désignation universelle (uniform naming) : **/dev/fd\***  
**/dev/mouse**



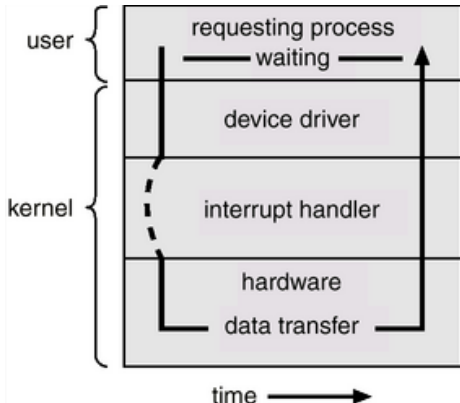
# Mécanismes d'E/S

## E/S synchrones (polling ou PIO)

- ▶ E/S bloquante
- ▶ Le processeur émet une commande à un périphérique et attend de manière active le changement du registre d'état
- ▶ Le périphérique traite la requête et change le registre d'état quand il a terminé et indique le succès ou une erreur

⇒ Utilisation CPU est faible

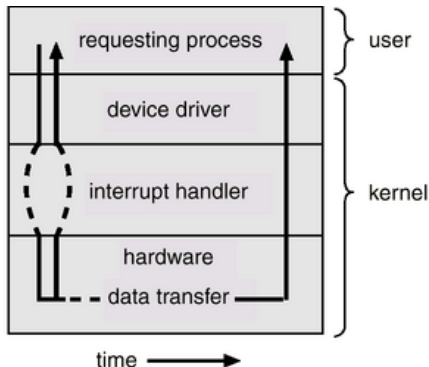
- ▶ **Solution** : interruptions



# Mécanismes d'E/S

## E/S asynchrones

- ▶ E/S non bloquante
- ▶ Le processeur émet une requête et continue son exécution
- ▶ Le périphérique traite la requête, une fois terminée, il émet une interruption
- ▶ Le processeur s'interrompt et interroge le périphérique pour savoir si la requête a été traitée avec succès

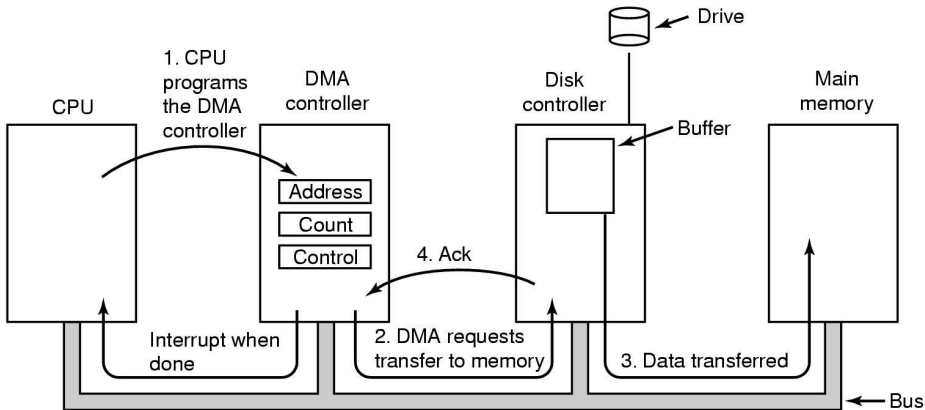


- ▶ Nécessaire dans le système pour multiplexage : ne pas bloquer le système pendant une E/S
- ▶ Peu habituel dans les applications : interface standard bloquante
- ▶ Recouvrement des E/S

## DMA (Direct Memory Access)

- ▶ Un moteur DMA gère le transfert de données entre la mémoire principale et un périphérique
- ▶ Le processeur envoie une requête de transfert à un moteur DMA :
  - ▶ type opération : lecture ou écriture
  - ▶ adresse du périphérique
  - ▶ adresse en mémoire centrale
  - ▶ longueur
- ▶ Le moteur DMA interrompt le processeur lorsque le transfert est terminé.
- ▶ Moteurs DMA placés sur les périphériques ou sur le bus d'entrées-sorties
- ▶ **Avantage** : amélioration de l'utilisation du CPU :
  - ▶ le transfert est juste initié par le CPU
  - ▶ pendant le transfert, le CPU peut faire autre chose

## DMA (Direct Memory Access)



## Quelle stratégie choisir

- ▶ Si le traitement par le périphérique est long  $\Rightarrow$  **interruption** : pas d'attente active
- ▶ Si la quantité de données à transférer est grande  $\Rightarrow$  **DMA** : pas de gaspillage du temps processeur
- ▶ Si la requête est simple à traiter  $\Rightarrow$  **PIO**

# Questionnaire 1.3

Durée : 10 minutes

# Première partie

## Généralités

- Structure générale d'un système informatique
- Cas du Pentium
- Composants d'un système d'exploitation
  - Système de protection
  - Système d'E/S
- Conception des systèmes d'exploitations
  - Le noyau
  - Structure simple : MS-DOS
  - Structure en couches
  - Systèmes monolithiques
  - Micro-noyau
  - Modules et noyaux modulaires
- Organisation générale du noyau Unix
  - Architecture du noyau Unix
  - Invocation des services système

# Conception des systèmes d'exploitations

---

Un système d'exploitation est un gros système et donc a besoin d'être conçu soigneusement pour fonctionner correctement.



# Conception des systèmes d'exploitations

---

Un système d'exploitation est un gros système et donc a besoin d'être conçu soigneusement pour fonctionner correctement.

## Le noyau (kernel)

- ▶ le premier logiciel chargé en mémoire (hors gestionnaire d'amorçage)
- ▶ la partie fondamentale et la plus critique d'un système d'exploitation

# Conception des systèmes d'exploitations

---

Un système d'exploitation est un gros système et donc a besoin d'être conçu soigneusement pour fonctionner correctement.

## Le noyau (kernel)

- ▶ le premier logiciel chargé en mémoire (hors gestionnaire d'amorçage)
- ▶ la partie fondamentale et la plus critique d'un système d'exploitation
- ▶ le gestionnaire des ressources de la machine qui permet aux éléments matériels et logiciels de communiquer entre eux, de fonctionner ensemble et de former un tout.

# Conception des systèmes d'exploitations

---

Un système d'exploitation est un gros système et donc a besoin d'être conçu soigneusement pour fonctionner correctement.

## Le noyau (kernel)

- ▶ le premier logiciel chargé en mémoire (hors gestionnaire d'amorçage)
- ▶ la partie fondamentale et la plus critique d'un système d'exploitation
- ▶ le gestionnaire des ressources de la machine qui permet aux éléments matériels et logiciels de communiquer entre eux, de fonctionner ensemble et de former un tout.

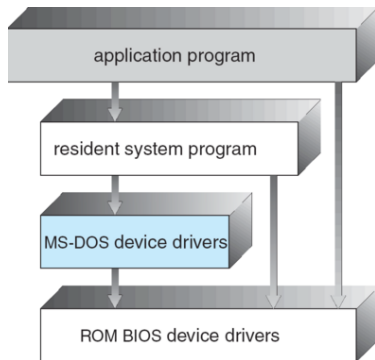
## Fonctions de base d'un noyau :

- ▶ chargement et exécution des processus
- ▶ gestion des entrées/sorties et du matériel en général
- ▶ interface avec les programmes de l'espace utilisateur
- ▶ gestion de la mémoire et ordonnancement

## Structure simple : MS-DOS

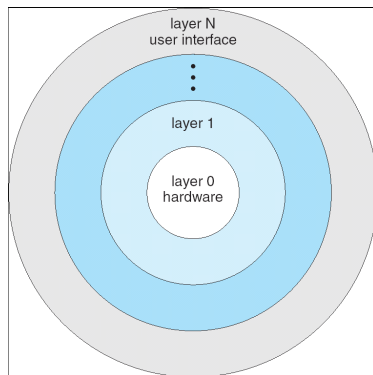
- ▶ Objectif : fournir un maximum de fonctions en un minimum d'espace
- ▶ non structuré en modules
- ▶ séparation non claire entre les différents niveaux

⇒ Structure en couches



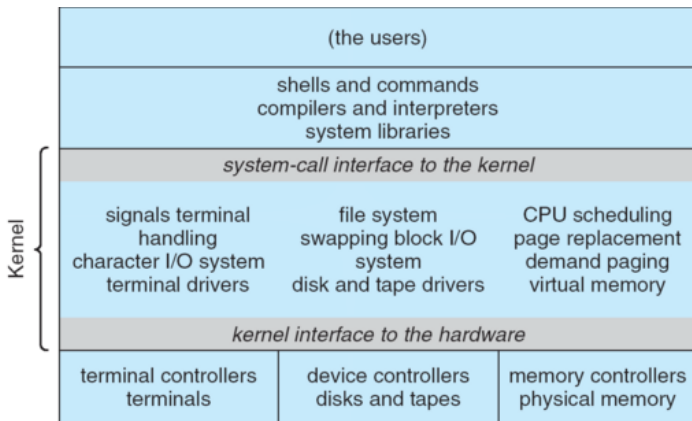
## Structure en couches

- ▶ le SE est divisé en couches
  - ▶ la couche la plus basse : le matériel
  - ▶ la couche la plus haute : l'interface utilisateur
- ▶ les couches sont conçues de telle sorte qu'elles utilisent les services de la couche du niveau inférieur



# Conception des systèmes d'exploitation

## Structure en couches : UNIX



⇒ Unix (premières versions) : système monolithique

# Conception des systèmes d'exploitation

---

## Systèmes monolithiques

L'ensemble des fonctions du système et des pilotes sont regroupés dans un seul bloc de code et un seul bloc binaire généré à la compilation

# Conception des systèmes d'exploitation

---

## Systèmes monolithiques

L'ensemble des fonctions du système et des pilotes sont regroupés dans un seul bloc de code et un seul bloc binaire généré à la compilation

**Exemples :** certains BSD et les anciennes versions de GNU/Linux et d'Unix



# Conception des systèmes d'exploitation

---

## Systèmes monolithiques

L'ensemble des fonctions du système et des pilotes sont regroupés dans un seul bloc de code et un seul bloc binaire généré à la compilation

**Exemples :** certains BSD et les anciennes versions de GNU/Linux et d'Unix

### Avantages

- ▶ facilité de conception et de développement
- ▶ vitesse d'exécution

# Conception des systèmes d'exploitation

---

## Systèmes monolithiques

L'ensemble des fonctions du système et des pilotes sont regroupés dans un seul bloc de code et un seul bloc binaire généré à la compilation

**Exemples :** certains BSD et les anciennes versions de GNU/Linux et d'Unix

### Avantages

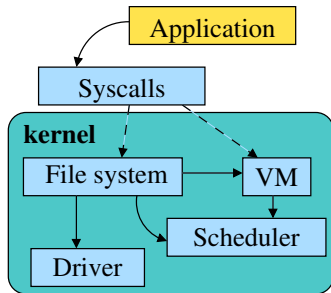
- ▶ facilité de conception et de développement
- ▶ vitesse d'exécution

### Inconvénients

- ▶ code volumineux, difficile à maintenir et à faire évoluer
- ▶ sécurité difficile à intégrer (interactions multiples)
- ▶ utilisation inutile de la mémoire (présence de tous les services même non utilisés)

# Conception des systèmes d'exploitation

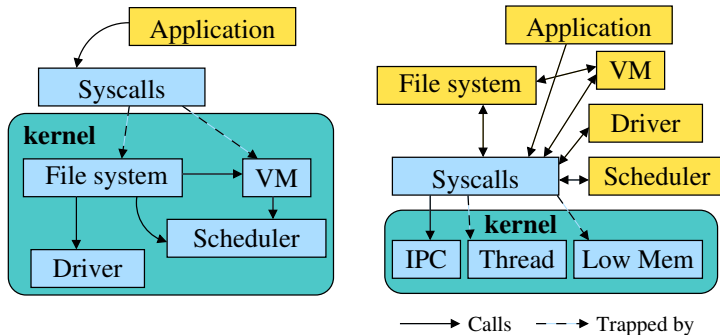
## Systèmes à micro-noyau



- Le noyau ne contient que les fonctions essentielles :

# Conception des systèmes d'exploitation

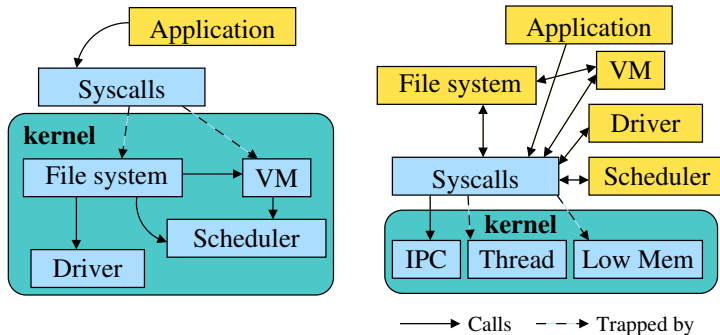
## Systèmes à micro-noyau



- ▶ Le noyau ne contient que les fonctions essentielles :
  - ▶ quelques primitives basiques de la gestion des processus, un ordonnanceur simple, des E/S simples et un mécanisme de communication entre processus (IPC)

# Conception des systèmes d'exploitation

## Systèmes à micro-noyau



- ▶ Le noyau ne contient que les fonctions essentielles :
  - ▶ quelques primitives basiques de la gestion des processus, un ordonnanceur simple, des E/S simples et un mécanisme de communication entre processus (IPC)
- ▶ Les autres services sont fournis par des programmes systèmes qui s'exécutent au-dessus du micro-noyau

# Conception des systèmes d'exploitation

---

## Systèmes à micro-noyau

### Avantages

- ▶ modularité, portabilité, fiabilité et facilité de la maintenance
- ▶ permettent de concevoir des systèmes très généraux
- ▶ plus de sécurité en plaçant de nombreux services "à risque" en espace utilisateur
- ▶ le système est très configurable
- ▶ meilleure utilisation de la mémoire

# Conception des systèmes d'exploitation

## Systèmes à micro-noyau

### Avantages

- ▶ modularité, portabilité, fiabilité et facilité de la maintenance
- ▶ permettent de concevoir des systèmes très généraux
- ▶ plus de sécurité en plaçant de nombreux services "à risque" en espace utilisateur
- ▶ le système est très configurable
- ▶ meilleure utilisation de la mémoire

### Inconvénients

- ▶ performances réduites à causes des mécanismes lourds de communication (IPC)
- ▶ le grand nombre d'appels système (la plupart des services sont à l'extérieur du noyau)
- ▶ difficile de comparer avec les noyaux monolithiques car pas de vrai OS micro-noyau fonctionnel à ce jour

# Conception des systèmes d'exploitation

---

## Les noyaux modulaires

les parties principales du système sont regroupées dans un bloc de code unique (monolithique). Les autres fonctions, les blocs de fonctions auxiliaires sont regroupés en différents **modules** qui peuvent être séparés (code et binaire).



# Conception des systèmes d'exploitation

---

## Les noyaux modulaires

les parties principales du système sont regroupées dans un bloc de code unique (monolithique). Les autres fonctions, les blocs de fonctions auxiliaires sont regroupés en différents **modules** qui peuvent être séparés (code et binaire).

- ▶ bon nombre des avantages théoriques des SE micro-noyau sans pénaliser les performances

# Conception des systèmes d'exploitation

## Les noyaux modulaires

les parties principales du système sont regroupées dans un bloc de code unique (monolithique). Les autres fonctions, les blocs de fonctions auxiliaires sont regroupés en différents **modules** qui peuvent être séparés (code et binaire).

- ▶ bon nombre des avantages théoriques des SE micro-noyau sans pénaliser les performances
- ▶ un module implémente un système de fichiers, un pilote de périphérique, une fonctionnalité spécifique ou toute autre fonction de haut niveau d'un SE.

# Conception des systèmes d'exploitation

## Les noyaux modulaires

les parties principales du système sont regroupées dans un bloc de code unique (monolithique). Les autres fonctions, les blocs de fonctions auxiliaires sont regroupés en différents **modules** qui peuvent être séparés (code et binaire).

- ▶ bon nombre des avantages théoriques des SE micro-noyau sans pénaliser les performances
- ▶ un module implémente un système de fichiers, un pilote de périphérique, une fonctionnalité spécifique ou toute autre fonction de haut niveau d'un SE.
- ▶ Chargement, déchargement dynamique de code

# Conception des systèmes d'exploitation

## Les noyaux modulaires

les parties principales du système sont regroupées dans un bloc de code unique (monolithique). Les autres fonctions, les blocs de fonctions auxiliaires sont regroupés en différents **modules** qui peuvent être séparés (code et binaire).

- ▶ bon nombre des avantages théoriques des SE micro-noyau sans pénaliser les performances
- ▶ un module implémente un système de fichiers, un pilote de périphérique, une fonctionnalité spécifique ou toute autre fonction de haut niveau d'un SE.
- ▶ Chargement, déchargement dynamique de code
- ▶ une fois inséré dans le noyau, un module est équivalent au code lié statiquement. Il est exécuté en mode noyau au nom du processus courant, comme toute fonction liée statiquement au noyau

# Conception des systèmes d'exploitation

## Les noyaux modulaires

les parties principales du système sont regroupées dans un bloc de code unique (monolithique). Les autres fonctions, les blocs de fonctions auxiliaires sont regroupés en différents **modules** qui peuvent être séparés (code et binaire).

- ▶ bon nombre des avantages théoriques des SE micro-noyau sans pénaliser les performances
- ▶ un module implémente un système de fichiers, un pilote de périphérique, une fonctionnalité spécifique ou toute autre fonction de haut niveau d'un SE.
- ▶ Chargement, déchargement dynamique de code
- ▶ une fois inséré dans le noyau, un module est équivalent au code lié statiquement. Il est exécuté en mode noyau au nom du processus courant, comme toute fonction liée statiquement au noyau
- ▶ aucun passage de message n'est nécessaire lorsque les fonctions du module sont invoquées

# Conception des systèmes d'exploitation

## Les noyaux modulaires

les parties principales du système sont regroupées dans un bloc de code unique (monolithique). Les autres fonctions, les blocs de fonctions auxiliaires sont regroupés en différents **modules** qui peuvent être séparés (code et binaire).

- ▶ bon nombre des avantages théoriques des SE micro-noyau sans pénaliser les performances
- ▶ un module implémente un système de fichiers, un pilote de périphérique, une fonctionnalité spécifique ou toute autre fonction de haut niveau d'un SE.
- ▶ Chargement, déchargement dynamique de code
- ▶ une fois inséré dans le noyau, un module est équivalent au code lié statiquement. Il est exécuté en mode noyau au nom du processus courant, comme toute fonction liée statiquement au noyau
- ▶ aucun passage de message n'est nécessaire lorsque les fonctions du module sont invoquées

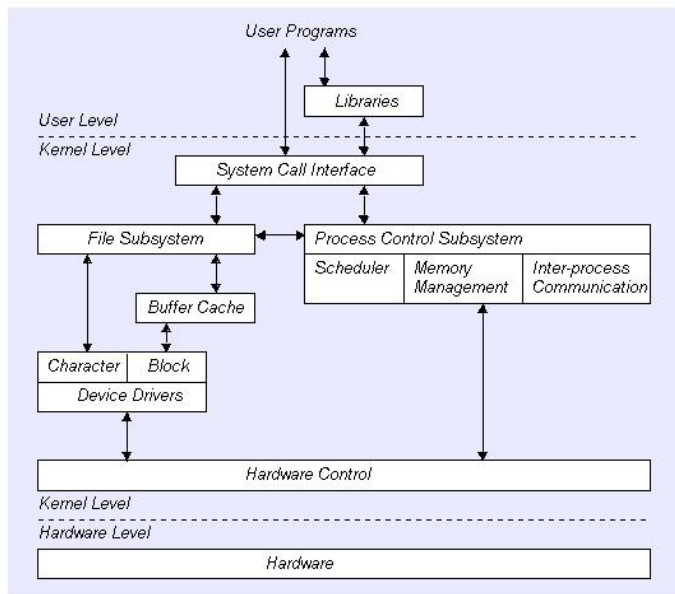
C'est le cas de la majorité des systèmes actuels comme GNU/Linux, Solaris et la plupart des BSD

# Première partie

## Généralités

- Structure générale d'un système informatique
- Cas du Pentium
- Composants d'un système d'exploitation
  - Système de protection
  - Système d'E/S
- Conception des systèmes d'exploitations
  - Le noyau
    - Structure simple : MS-DOS
    - Structure en couches
    - Systèmes monolithiques
    - Micro-noyau
    - Modules et noyaux modulaires
- Organisation générale du noyau Unix
  - Architecture du noyau Unix
  - Invocation des services système

# Architecture du noyau Unix





# Architecture du noyau : approche descriptive

---

Le noyau constitué de 3 grandes parties :

- ▶ **l'interface des appels système**, interface entre les programmes utilisateur et le noyau

# Architecture du noyau : approche descriptive

---

Le noyau constitué de 3 grandes parties :

- ▶ l'interface des appels système, interface entre les programmes utilisateur et le noyau
- ▶ le sous-système de gestion des processus

# Architecture du noyau : approche descriptive

---

Le noyau constitué de 3 grandes parties :

- ▶ **l'interface des appels système**, interface entre les programmes utilisateur et le noyau
- ▶ **le sous-système de gestion des processus**
  - ▶ **gestion des processus** : création, terminaison, suspension, synchronisation et communication.

# Architecture du noyau : approche descriptive

---

Le noyau constitué de 3 grandes parties :

- ▶ **l'interface des appels système**, interface entre les programmes utilisateur et le noyau
- ▶ **le sous-système de gestion des processus**
  - ▶ **gestion des processus** : création, terminaison, suspension, synchronisation et communication.
  - ▶ **ordonnancement** : traite la gestion du partage du temps et des priorités.

# Architecture du noyau : approche descriptive

---

Le noyau constitué de 3 grandes parties :

- ▶ **l'interface des appels système**, interface entre les programmes utilisateur et le noyau
- ▶ **le sous-système de gestion des processus**
  - ▶ **gestion des processus** : création, terminaison, suspension, synchronisation et communication.
  - ▶ **ordonnancement** : traite la gestion du partage du temps et des priorités.
  - ▶ **gestion de la mémoire** : gère le partage des objets, la protection interprocessus, le swapping ou la pagination.

# Architecture du noyau : approche descriptive

---

Le noyau constitué de 3 grandes parties :

- ▶ **l'interface des appels système**, interface entre les programmes utilisateur et le noyau
- ▶ **le sous-système de gestion des processus**
  - ▶ **gestion des processus** : création, terminaison, suspension, synchronisation et communication.
  - ▶ **ordonnancement** : traite la gestion du partage du temps et des priorités.
  - ▶ **gestion de la mémoire** : gère le partage des objets, la protection interprocessus, le swapping ou la pagination.
- ▶ **le sous système de gestion des fichiers**

# Architecture du noyau : approche descriptive

---

Le noyau constitué de 3 grandes parties :

- ▶ **l'interface des appels système**, interface entre les programmes utilisateur et le noyau
- ▶ **le sous-système de gestion des processus**
  - ▶ **gestion des processus** : création, terminaison, suspension, synchronisation et communication.
  - ▶ **ordonnancement** : traite la gestion du partage du temps et des priorités.
  - ▶ **gestion de la mémoire** : gère le partage des objets, la protection interprocessus, le swapping ou la pagination.
- ▶ **le sous système de gestion des fichiers**
  - ▶ **gestion du buffer cache** : gère l'allocation des tampons d'E/S.

# Architecture du noyau : approche descriptive

---

Le noyau constitué de 3 grandes parties :

- ▶ **l'interface des appels système**, interface entre les programmes utilisateur et le noyau
- ▶ **le sous-système de gestion des processus**
  - ▶ **gestion des processus** : création, terminaison, suspension, synchronisation et communication.
  - ▶ **ordonnancement** : traite la gestion du partage du temps et des priorités.
  - ▶ **gestion de la mémoire** : gère le partage des objets, la protection interprocessus, le swapping ou la pagination.
- ▶ **le sous système de gestion des fichiers**
  - ▶ **gestion du buffer cache** : gère l'allocation des tampons d'E/S.
  - ▶ **gestion des fichiers** : traite la protection, l'allocation de l'espace disque, la désignation des fichiers



# Architecture du noyau : approche descriptive

---

Le noyau constitué de 3 grandes parties :

- ▶ **l'interface des appels système**, interface entre les programmes utilisateur et le noyau
- ▶ **le sous-système de gestion des processus**
  - ▶ **gestion des processus** : création, terminaison, suspension, synchronisation et communication.
  - ▶ **ordonnancement** : traite la gestion du partage du temps et des priorités.
  - ▶ **gestion de la mémoire** : gère le partage des objets, la protection interprocessus, le swapping ou la pagination.
- ▶ **le sous système de gestion des fichiers**
  - ▶ **gestion du buffer cache** : gère l'allocation des tampons d'E/S.
  - ▶ **gestion des fichiers** : traite la protection, l'allocation de l'espace disque, la désignation des fichiers
  - ▶ **gestion des périphériques** : gère les fichiers en mode caractère et en mode bloc, l'accès aux périphériques, y compris aux réseaux.

# Architecture du noyau : approche fonctionnelle

---

Le noyau UNIX est découpé en 2 grandes parties qui coopèrent pour le partage des ressources du système et pour la mise-en-oeuvre de certains services :

# Architecture du noyau : approche fonctionnelle

---

Le noyau UNIX est découpé en 2 grandes parties qui coopèrent pour le partage des ressources du système et pour la mise-en-oeuvre de certains services :

## Partie supérieure

fournit des services aux processus utilisateurs en réponse aux appels système et aux exceptions.

# Architecture du noyau : approche fonctionnelle

---

Le noyau UNIX est découpé en 2 grandes parties qui coopèrent pour le partage des ressources du système et pour la mise-en-oeuvre de certains services :

## Partie supérieure

fournit des services aux processus utilisateurs en réponse aux appels système et aux exceptions.

- ▶ exécution synchrone en mode noyau pour pouvoir accéder à la fois aux structure de données du noyau et aux contextes des processus utilisateurs.

# Architecture du noyau : approche fonctionnelle

---

Le noyau UNIX est découpé en 2 grandes parties qui coopèrent pour le partage des ressources du système et pour la mise-en-oeuvre de certains services :

## Partie supérieure

fournit des services aux processus utilisateurs en réponse aux appels système et aux exceptions.

- ▶ exécution synchrone en mode noyau pour pouvoir accéder à la fois aux structure de données du noyau et aux contextes des processus utilisateurs.

## Partie inférieure

composée d'un ensemble de sous-programmes invoqués pour le traitement des interruptions matérielles

# Architecture du noyau : approche fonctionnelle

---

Le noyau UNIX est découpé en 2 grandes parties qui coopèrent pour le partage des ressources du système et pour la mise-en-oeuvre de certains services :

## Partie supérieure

fournit des services aux processus utilisateurs en réponse aux appels système et aux exceptions.

- ▶ exécution **synchrone** en mode noyau pour pouvoir accéder à la fois aux structure de données du noyau et aux **contextes** des processus utilisateurs.

## Partie inférieure

composée d'un ensemble de sous-programmes invoqués pour le traitement des interruptions matérielles

- ▶ des activités se déroulant d'une façon **asynchrone** et s'exécutent en mode noyau

## Traitement d'une exception ou d'une interruption

- ▶ Arrivée de l'interruption/exception

## Traitement d'une exception ou d'une interruption

- ▶ Arrivée de l'interruption/exception
- ▶ Sauvegarde du contexte actuel (PC ...) en utilisant la pile noyau



## Traitement d'une exception ou d'une interruption

- ▶ Arrivée de l'interruption/exception
- ▶ Sauvegarde du contexte actuel (PC ...) en utilisant la pile noyau
- ▶ Accès à la table des vecteurs d'interruptions pour déterminer l'adresse du sous-programme de l'interruption (le **handler**) et chargement du PC avec son adresse

## Traitement d'une exception ou d'une interruption

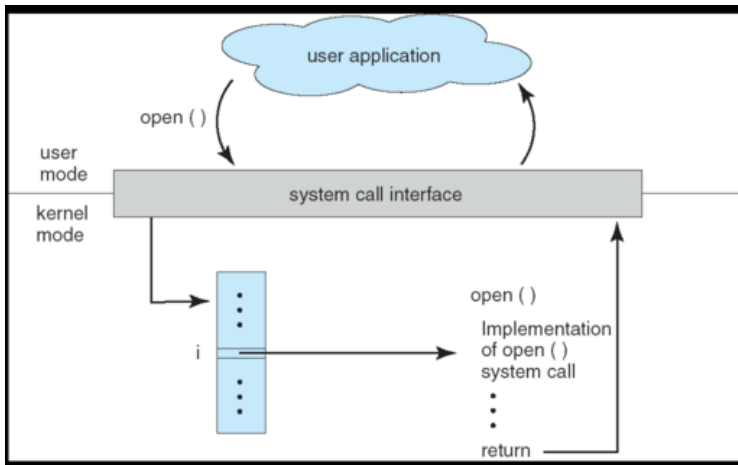
- ▶ Arrivée de l'interruption/exception
- ▶ Sauvegarde du contexte actuel (PC ...) en utilisant la pile noyau
- ▶ Accès à la table des vecteurs d'interruptions pour déterminer l'adresse du sous-programme de l'interruption (le **handler**) et chargement du PC avec son adresse
- ▶ Exécution du sous-programme en **mode noyau**

## Traitement d'une exception ou d'une interruption

- ▶ Arrivée de l'interruption/exception
- ▶ Sauvegarde du contexte actuel (PC ...) en utilisant la pile noyau
- ▶ Accès à la table des vecteurs d'interruptions pour déterminer l'adresse du sous-programme de l'interruption (le **handler**) et chargement du PC avec son adresse
- ▶ Exécution du sous-programme en **mode noyau**
- ▶ Rétablissement de l'ancien contexte et reprise de l'ancien programme en **mode utilisateur**

# Invocation des services système

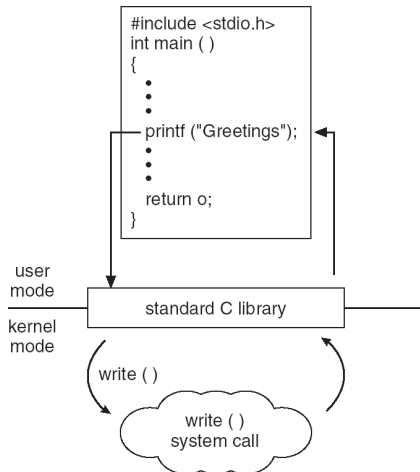
## Implémentation des appels système



# Invocation des services système

## Bibliothèque standard C

- ▶ le code d'un appel système est souvent en assembleur, mais une fonction de bibliothèque de fonctions en C est souvent fournie.





# Invocation des services système


## Un exemple d'appels système : read()

count = read(df, tampon, nbOctets)

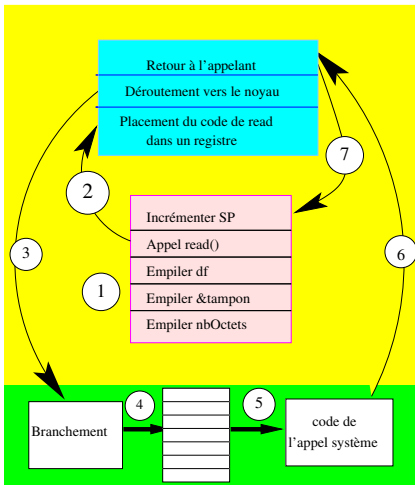
0xFFFFFFFF

 Fonction read()  
de bibliothèque

 Appel à read()  
depuis le programme  
utilisateur

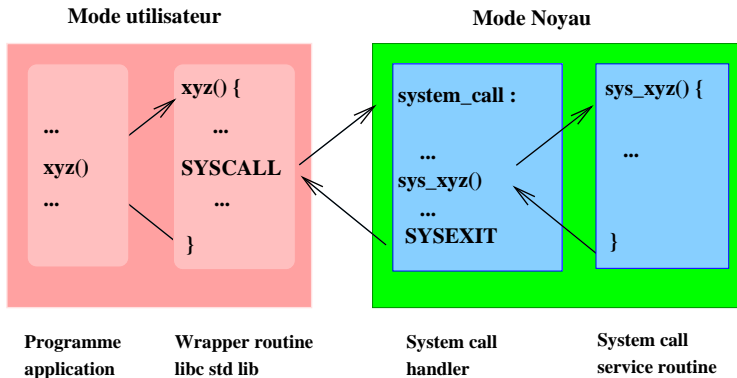
 Espace utilisateur

 Espace noyau



# Invocation des services système

## Appels système sous Linux



# Questionnaire 2

Durée : 10 minutes