

# Systèmes d'exploitation

Introduction: shell, noyau, syscalls

Guillaume Salagnac

Insa de Lyon – Informatique

## IFA-3-SYS : Systèmes d'Exploitation

Guillaume Salagnac <guillaume.salagnac@insa-lyon.fr>

- enseignement : architecture, compilation, système, algo
- recherche : numérique pour l'anthropocène, logiciel soutenable

### Objectifs de ce cours

- Comprendre les «concepts clés» des systèmes d'exploitation
  - quel est le problème ? pourquoi se pose-t-il ? toujours ?
  - quelle sont la/les solutions ? pourquoi ça marche ?
    - ▶ prenez des notes en cours !
- Pratiquer leur usage
  - TP de programmation en C sous Linux (4×2h)
  - TD sur papier (3×2h, dont 1 contrôle continu)
  - ▶ pas de CR ; prenez des notes aussi !

### Ressources

- <http://moodle.insa-lyon.fr> > Informatique > IFA-3
  - ▶ vidéos et diapos ; sujets de TD et de TP ; annales d'examen
- Permanence en salle 208 tous les jeudi de 13h à 14h

2/48

## Plan du cours

- **Chap 1** Noyau, processus, appels système 28/01
    - TP manipulation de processus : `fork()`, `exec()`, `waitpid()`
  - **Chap 2** Multitâche, temps partagé, ordonnancement
    - TD ordonnancement de processus
  - **Chap 3** Mémoire virtuelle, isolation, pagination à la demande
    - TP allocation et entrées-sorties fichier avec `mmap()`
- Contrôle continu (30 min, coeff 1) 24/02 à 8h
- **Chap 4** Allocation dynamique
    - TP implémentation de `malloc()` et `free()`
  - **Chap 5** Concurrency et synchronisation
    - TD algorithmes concurrents avec mutex et moniteurs
    - TP programmation avec `pthread`s

Examen final (1h30, coeff 2) 18/03

3/48

## Évaluation

### Contrôle «continu»

- sous forme de QCM papier (30 min, le 24/02)
- questions de compréhension et petits exercices
  - dont des questions sur les TD/TP
- sans documents **sauf une feuille A4 recto-verso manuscrite**
- plusieurs contrôles au fil de l'eau pendant le semestre
- anciens sujets sur Moodle

### Examen final

- épreuve écrite de 1h30 le 18/03
- sans documents **sauf une feuille A4 recto-verso manuscrite**
- anciens sujets sur Moodle

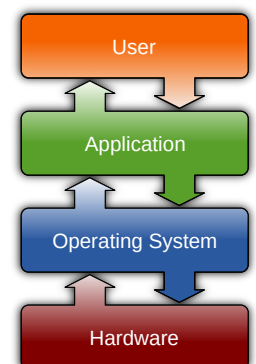
4/48

## Plan

1. Introduction : définition du terme «Système d'exploitation»
2. Interface entre OS et utilisateur : le shell
3. Interface entre logiciel et matériel : l'architecture
4. Isolation entre noyau et applications : les syscalls
5. Quelques syscalls UNIX incontournables

5/48

## Vous avez dit «Système d'exploitation» ?



6/48

## Quelques définitions

**Utilisateur** = l'humain devant la machine

- suivant le contexte : utilisateur final, ou développeur
- interagit directement... avec le matériel !

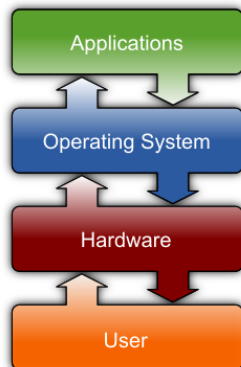
**Applications** = les logiciels avec lesquels veut interagir l'utilisateur final

- messagerie, traitement de texte, lecteur de musique, etc

**Matériel** = la machine physique

Et donc : **Operating System** = tout le reste

- logiciel d'infrastructure : «noyau», «pilotes», «services», etc
- «entre le matériel et les applications»



7/48

## Rôle de l'OS : les deux fonctions essentielles

et largement interdépendantes !

### Machine virtuelle

- cacher la complexité sous une interface «plus jolie»
- fournir certains **services de base** aux applications
  - IHM, stockage persistant, accès internet, gestion du temps
- permettre la **portabilité** des programmes
  - pouvoir lancer un même exécutable sur différents matériels

### Gestionnaire de ressources

- **partager** chaque ressource entre les applications
- **exploiter «au mieux»** les ressources disponibles
- assurer la **protection** des applications (et du système)

8/48

## Plan

1. Introduction : définition du terme «Système d'exploitation»
2. Interface entre OS et utilisateur : le shell
3. Interface entre logiciel et matériel : l'architecture
4. Isolation entre noyau et applications : les syscalls
5. Quelques syscalls UNIX incontournables

9/48

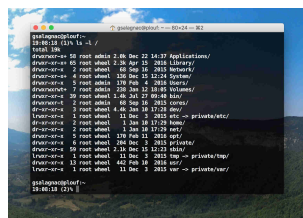
## Interface entre OS et utilisateur : le shell

Les services offerts par un shell :

- **exécution** de programmes
  - charger un programme en mémoire, le lancer, l'arrêter
  - choisir quel programme est au premier-plan
- exploration et administration des **espaces de stockage**
  - naviguer dans les fichiers, copier, supprimer
- confort et **ergonomie**
  - presse-papiers, drag-and-drop, corbeille
- ...

10/48

## Différents types de shell : l'interpréteur de commandes



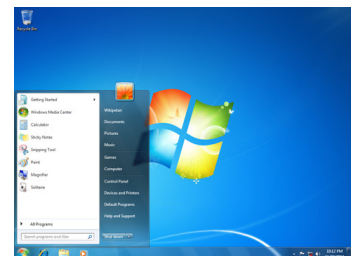
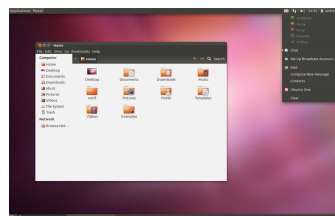
Attention : **terminal** (ou émulateur de terminal)  $\neq$  **shell** !

Interface textuelle = **Command-Line Interface** = CLI

exemples : Bourne shell (1977), bash, zsh...

11/48

## Différents types de shell : le bureau graphique

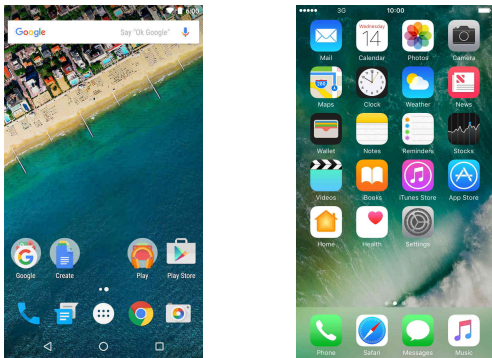


Interface graphique = **Graphical User Interface** = GUI

exemples : Gnome, bureau de Windows, Aqua (Mac OSX)...

12/48

## et encore : l'écran d'accueil du smartphone



exemples : Android Launcher, Nova Launcher, Square Home...

13/48

## Conclusion : le shell

différents types de shell : CLI vs GUI à souris vs GUI tactile

- ▶ fonctionnalités similaires
- ▶ pour l'OS : une «application» comme les autres !

votre OS contient volontiers des applications **pré-installées**...

- shell, navigateur web, explorateur de fichiers, messagerie, lecteur multimedia, app store, etc

... et aussi plein de «programmes système» :

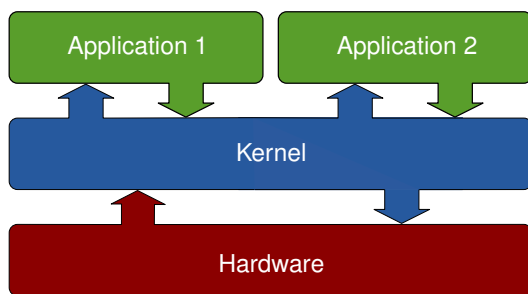
- développement : compilateur, assembleur, linker, etc
- sécurité : antivirus, pare-feu, sauvegarde
- maintenance : mise à jour, panneau de configuration
- services réseau : web, base de données, accès distant

Remarque :

dorénavant je vais appeler tous ces programmes des **applications**

14/48

## Positionnement de l'OS



Définition : **Noyau**, ou en VO kernel

Le noyau c'est la partie de l'OS qui n'est pas une application

15/48

## Plan

1. Introduction : définition du terme «Système d'exploitation»
2. Interface entre OS et utilisateur : le shell
3. Interface entre logiciel et matériel : l'architecture
4. Isolation entre noyau et applications : les syscalls
5. Quelques syscalls UNIX incontournables

16/48

## Vous avez dit «une interface plus jolie» ?

et c'est vraiment cette formule qui est donnée dans les livres :

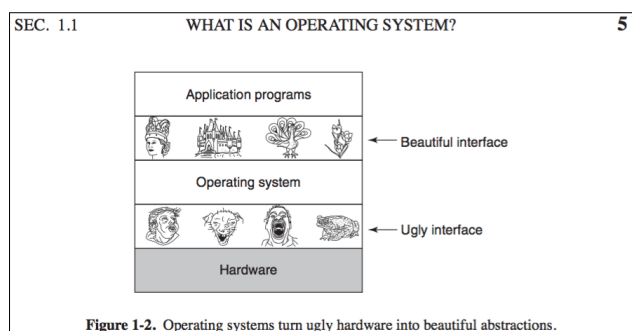


Figure 1-2. Operating systems turn ugly hardware into beautiful abstractions.

source : Tanenbaum. *Modern Operating Systems* (4th ed, 2014). page 5

17/48

## Un exemple de programme : la commande cat

```
int main() {
    char buffer[100];
    int n;

    int fd=open("filename.txt", O_RDONLY);
    if(fd == -1)
        exit(1);

    n=read(fd, buffer, 100);
    while( n > 0 ) {
        write(STDOUT, buffer, n);
        n=read(fd, buffer, 100);
    }
    exit(0);
}
```

18/48

# Architecture d'une machine typique

The diagram illustrates a hierarchical computer architecture. At the top, multiple CPUs (CPU1, CPU2, CPU3, ...) are connected to a horizontal 'System bus'. Below the system bus, 'main memory' and an 'I/O bridge' are connected. The 'I/O bridge' connects the system bus to an 'I/O bus'. Various I/O controllers are connected to the I/O bus: a 'USB controller' (which connects to a 'USB bus' and then to a 'mouse' and 'keyboard'), a 'disk controller' (which connects to a 'disk'), and a 'network adapter' (which connects to a 'network'). Ellipses (...) indicate additional components and connections.



# Langage de programmation vs langage machine

```
00401be5 <main>:
401be5: 55                push    %rbp
401be6: 48 89 e5          mov     %rsp,%rbp
401be9: 48 83 ec 70       sub     $0x70,%rsp
401bed: be 00 00 00 00    mov     $0x0,%esi
401bf2: bf 10 20 48 00    mov     $0x482010,%edi
401bf7: b8 00 00 00 00    mov     $0x0,%eax
401bfc: e8 7f c5 03 00    callq   43e180 <__libc_open>
401c01: 89 45 f8          mov     %eax,-0x8(%rbp)
401c04: 83 7d f8 ff       cmpl    $0xffffffff,-0x8(%rbp)
401c08: 75 0a            jne     401c14 <main+0x2f>
401c0a: bf 01 00 00 00    mov     $0x1,%edi
401c0f: e8 7c 6a 00 00    callq   408690 <exit>
401c14: 48 8d 4d 90       lea     -0x70(%rbp),%rcx
401c18: 8b 45 f8          mov     -0x8(%rbp),%eax
401c1b: ba 64 00 00 00    mov     $0x64,%edx
401c20: 48 89 ce          mov     %rcx,%rsi
401c23: 89 c7            mov     %eax,%edi
401c25: e8 86 c6 03 00    callq   43e2b0 <__libc_read>
401c2a: 89 45 fc          mov     %eax,-0x4(%rbp)
401c2d: eb 30            jmp     401c5f <main+0x7a>
401c2f: 8b 45 fc          mov     -0x4(%rbp),%eax
401c32: 48 89 e5          mov     %rsp,%rbp
401c35: 5d                pop     %rbp
401c36: c3                retq    0
401c37: 90                nop     0
401c38: 90                nop     0
401c39: 90                nop     0
401c3a: 90                nop     0
401c3b: 90                nop     0
401c3c: 90                nop     0
401c3d: 90                nop     0
401c3e: 90                nop     0
401c3f: 90                nop     0
401c40: 90                nop     0
401c41: 90                nop     0
401c42: 90                nop     0
401c43: 90                nop     0
401c44: 90                nop     0
401c45: 90                nop     0
401c46: 90                nop     0
401c47: 90                nop     0
401c48: 90                nop     0
401c49: 90                nop     0
401c4a: 90                nop     0
401c4b: 90                nop     0
401c4c: 90                nop     0
401c4d: 90                nop     0
401c4e: 90                nop     0
401c4f: 90                nop     0
401c50: 90                nop     0
401c51: 90                nop     0
401c52: 90                nop     0
401c53: 90                nop     0
401c54: 90                nop     0
401c55: 90                nop     0
401c56: 90                nop     0
401c57: 90                nop     0
401c58: 90                nop     0
401c59: 90                nop     0
401c5a: 90                nop     0
401c5b: 90                nop     0
401c5c: 90                nop     0
401c5d: 90                nop     0
401c5e: 90                nop     0
401c5f: 90                nop     0
401c60: 90                nop     0
401c61: 90                nop     0
401c62: 90                nop     0
401c63: 90                nop     0
401c64: 90                nop     0
401c65: 90                nop     0
401c66: 90                nop     0
401c67: 90                nop     0
401c68: 90                nop     0
401c69: 90                nop     0
401c6a: 90                nop     0
401c6b: 90                nop     0
401c6c: 90                nop     0
401c6d: 90                nop     0
401c6e: 90                nop     0
401c6f: 90                nop     0
401c70: 90                nop     0
401c71: 90                nop     0
401c72: 90                nop     0
401c73: 90                nop     0
401c74: 90                nop     0
401c75: 90                nop     0
401c76: 90                nop     0
401c77: 90                nop     0
401c78: 90                nop     0
401c79: 90                nop     0
401c7a: 90                nop     0
401c7b: 90                nop     0
401c7c: 90                nop     0
401c7d: 90                nop     0
401c7e: 90                nop     0
401c7f: 90                nop     0
401c80: 90                nop     0
401c81: 90                nop     0
401c82: 90                nop     0
401c83: 90                nop     0
401c84: 90                nop     0
401c85: 90                nop     0
401c86: 90                nop     0
401c87: 90                nop     0
401c88: 90                nop     0
401c89: 90                nop     0
401c8a: 90                nop     0
401c8b: 90                nop     0
401c8c: 90                nop     0
401c8d: 90                nop     0
401c8e: 90                nop     0
401c8f: 90                nop     0
401c90: 90                nop     0
401c91: 90                nop     0
401c92: 90                nop     0
401c93: 90                nop     0
401c94: 90                nop     0
401c95: 90                nop     0
401c96: 90                nop     0
401c97: 90                nop     0
401c98: 90                nop     0
401c99: 90                nop     0
401c9a: 90                nop     0
401c9b: 90                nop     0
401c9c: 90                nop     0
401c9d: 90                nop     0
401c9e: 90                nop     0
401c9f: 90                nop     0
401ca0: 90                nop     0
401ca1: 90                nop     0
401ca2: 90                nop     0
401ca3: 90                nop     0
401ca4: 90                nop     0
401ca5: 90                nop     0
401ca6: 90                nop     0
401ca7: 90                nop     0
401ca8: 90                nop     0
401ca9: 90                nop     0
401caa: 90                nop     0
401cab: 90                nop     0
401cac: 90                nop     0
401cad: 90                nop     0
401cae: 90                nop     0
401caf: 90                nop     0
401cb0: 90                nop     0
401cb1: 90                nop     0
401cb2: 90                nop     0
401cb3: 90                nop     0
401cb4: 90                nop     0
401cb5: 90                nop     0
401cb6: 90                nop     0
401cb7: 90                nop     0
401cb8: 90                nop     0
401cb9: 90                nop     0
401cba: 90                nop     0
401cbb: 90                nop     0
401cbc: 90                nop     0
401cbd: 90                nop     0
401cbe: 90                nop     0
401cbf: 90                nop     0
401cc0: 90                nop     0
401cc1: 90                nop     0
401cc2: 90                nop     0
401cc3: 90                nop     0
401cc4: 90                nop     0
401cc5: 90                nop     0
401cc6: 90                nop     0
401cc7: 90                nop     0
401cc8: 90                nop     0
401cc9: 90                nop     0
401cca: 90                nop     0
401ccb: 90                nop     0
401ccc: 90                nop     0
401ccd: 90                nop     0
401cce: 90                nop     0
401ccf: 90                nop     0
401cd0: 90                nop     0
401cd1: 90                nop     0
401cd2: 90                nop     0
401cd3: 90                nop     0
401cd4: 90                nop     0
401cd5: 90                nop     0
401cd6: 90                nop     0
401cd7: 90                nop     0
401cd8: 90                nop     0
401cd9: 90                nop     0
401cda: 90                nop     0
401cdb: 90                nop     0
401cdc: 90                nop     0
401cdd: 90                nop     0
401cde: 90                nop     0
401cdf: 90                nop     0
401ce0: 90                nop     0
401ce1: 90                nop     0
401ce2: 90                nop     0
401ce3: 90                nop     0
401ce4: 90                nop     0
401ce5: 90                nop     0
401ce6: 90                nop     0
401ce7: 90                nop     0
401ce8: 90                nop     0
401ce9: 90                nop     0
401cea: 90                nop     0
401ceb: 90                nop     0
401cec: 90                nop     0
401ced: 90                nop     0
401cee: 90                nop     0
401cef: 90                nop     0
401cf0: 90                nop     0
401cf1: 90                nop     0
401cf2: 90                nop     0
401cf3: 90                nop     0
401cf4: 90                nop     0
401cf5: 90                nop     0
401cf6: 90                nop     0
401cf7: 90                nop     0
401cf8: 90                nop     0
401cf9: 90                nop     0
401cfa: 90                nop     0
401cfb: 90                nop     0
401cfc: 90                nop     0
401cfd: 90                nop     0
401cfe: 90                nop     0
401cff: 90                nop     0
401d00: 90                nop     0
401d01: 90                nop     0
401d02: 90                nop     0
401d03: 90                nop     0
401d04: 90                nop     0
401d05: 90                nop     0
401d06: 90                nop     0
401d07: 90                nop     0
401d08: 90                nop     0
401d09: 90                nop     0
401d0a: 90                nop     0
401d0b: 90                nop     0
401d0c: 90                nop     0
401d0d: 90                nop     0
401d0e: 90                nop     0
401d0f: 90                nop     0
401d10: 90                nop     0
401d11: 
```

20/48

The diagram illustrates the Von Neumann architecture and its restricted mode. At the top, a red box contains the text 'Applications = CPU en «mode restreint»'. Below this, a diagram shows a 'CPU' box and a 'memory' box (represented as a grid) connected by a thick grey arrow that forms a loop, indicating the flow of data and instructions between them. To the right, a light pink box titled 'Rappel : le cycle de Von Neumann' lists the steps of the cycle: 'while True do :', 'charger une instruction depuis la «mémoire»', 'décoder ses bits : quelle opération, quelles opérandes, etc', 'exécuter l'opération et enregistrer le résultat', and 'repeat'. Below this, a green box titled 'Définition : restricted mode = slave mode = ring 3 = user mode' lists three points: 'vue partielle de la machine : 1 CPU + 1 mémoire', 'certaines instructions interdites, certaines adresses interdites', and 'utile pour exécuter sereinement du code applicatif'. At the bottom, a light blue box lists 'instructions disponibles : opérations ALU, accès mémoire, sauts' and provides three example instructions in boxes: 'ADD R1 <- R3, R4', 'WRITE [R8] <- R5', and 'CALL 123456'.

Applications = CPU en «mode restreint»

**Rappel : le cycle de Von Neumann**

while True do :

- charger une instruction depuis la «mémoire»
- décoder ses bits : quelle opération, quelles opérandes, etc
- exécuter l'opération et enregistrer le résultat

repeat

**Définition : restricted mode = slave mode = ring 3 = user mode**

- vue partielle de la machine : 1 CPU + 1 mémoire
- certaines instructions interdites, certaines adresses interdites
- utile pour exécuter sereinement du code applicatif

instructions disponibles : opérations ALU, accès mémoire, sauts

ADD R1 <- R3, R4      WRITE [R8] <- R5      CALL 123456



The diagram illustrates a computer system architecture. At the top, three CPU boxes (CPU1, CPU2, CPU3) and an ellipsis are connected to a horizontal 'System bus'. Below the system bus, a vertical line connects to a 'main memory' box and an 'I/O bridge' box. The 'I/O bridge' is connected to a horizontal 'I/O bus'. Below the I/O bus, three boxes are shown: 'USB controller', 'disk controller', and 'network adapter', followed by an ellipsis. The 'USB controller' is connected to a 'USB bus', which in turn connects to a 'mouse' and a 'keyboard'. The 'disk controller' is connected to a 'disk'. The 'network adapter' is connected to a 'network'. Arrows indicate the direction of data flow between components.

**Noyau = CPU en «mode superviseur»**

Diagram illustrating the system architecture:

- CPUs (CPU1, CPU2, CPU3, ...) are connected to the **System bus**.
- The **System bus** connects to **main memory** and the **I/O bridge**.
- The **I/O bridge** connects to the **I/O bus**.
- The **I/O bus** connects to various I/O devices: **USB controller**, **disk controller**, **network adapter**, and others.
- The **USB controller** connects to the **USB bus**, which connects to **mouse** and **keyboard**.
- The **disk controller** connects to the **disk**.
- The **network adapter** connects to the **network**.

**Définition : supervisor mode = ring 0 = kernel mode = privileged mode**

- accès **direct** au matériel : nécessaire pour le noyau et les **drivers**
- SW  $\rightarrow$  HW = Memory-mapped I/O      HW  $\rightarrow$  SW = Interruptions
- mode par défaut au démarrage de la machine (boot)

22/48

[illegible]

Diagram illustrating a computer system architecture for a disk request:

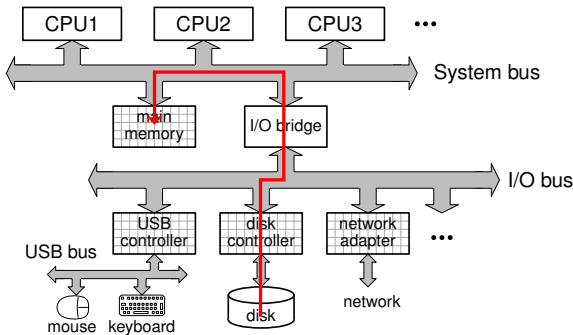
- System bus:** Connects multiple CPUs (CPU1, CPU2, CPU3, ...) to the main memory and the I/O bridge.
- I/O bridge:** Connects the System bus to the I/O bus.
- I/O bus:** Connects the I/O bridge to various controllers:
  - USB controller:** Connected to a mouse and keyboard.
  - disk controller:** Connected to a disk.
  - network adapter:** Connected to a network.

A red line traces the path of a request from CPU1, through the System bus, I/O bridge, and I/O bus to the disk controller and then to the disk.



### Exemple : lecture sur le disque 2/3

DMA

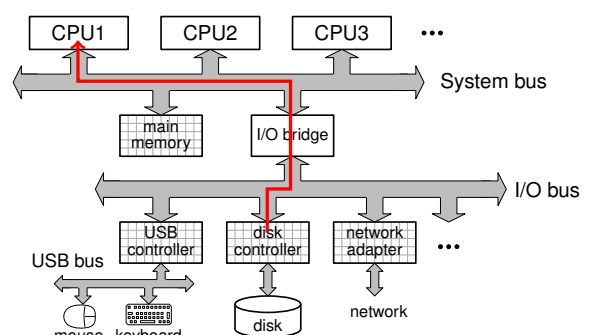


Le contrôleur de disque lit le secteur demandé et transfère les données directement en mémoire vive à l'adresse voulue : c'est un **transfert DMA** (Direct Memory Access)

25/48

### Exemple : lecture sur le disque 3/3

IRQ



À la fin du transfert DMA, le contrôleur du périphérique notifie le CPU en lui envoyant une **Requête d'Interruption (IRQ)**

26/48

### Un processeur avec support des interruptions

#### Le cycle de Von Neumann avec interruptions

while True do :

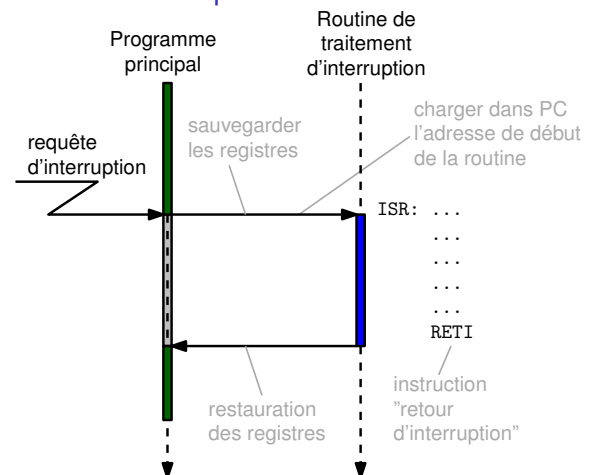
```

charger une instruction depuis la mémoire
décoder ses bits : quelle opération, quelles opérandes, etc
exécuter l'opération et enregistrer le résultat
if interruption demandée then :
    sauvegarder le contenu des registres
    déterminer l'adresse de la routine de traitement
    passer en mode superviseur
    sauter à la routine = écrire son adresse dans le compteur ordinal
endif
repeat
    
```

Note : à la fin de la routine de traitement, une instruction RETI repassera le CPU en *mode restreint*.

27/48

### Mécanisme d'interruptions : déroulement



28/48

### Mécanisme d'interruptions : vocabulaire

- IRQ = **Interrupt Request**
  - un «message» envoyé par un périphérique vers le processeur
  - de façon asynchrone (vs polling, inefficace)
  - chaque IRQ porte un numéro identifiant le périphérique d'origine
- ISR = **Interrupt Service Routine**
  - un fragment de programme (= séquence d'instructions) exécuté à chaque occurrence de l'évènement matériel associé
  - termine toujours par une instruction RETI «retour d'interruption»
  - pendant une ISR : nouvelles IRQ temporairement mises en attente (permet au programmeur d'être «seul au monde»)
- Table des Vecteurs d'Interruptions
  - tableau de pointeurs indiquant l'adresse de chaque ISR
  - le CPU utilise le numéro d'IRQ pour savoir où sauter

#### Définition : **Noyau**, ou en VO kernel

Le noyau c'est (exactement) l'ensemble des ISR de la machine

- et de toutes les fonctions que celles-ci appellent

29/48

### Différentes sources d'interruptions

- Périphériques d'**entrées-sorties**
  - clavier, souris, disque, GPU, réseau, etc
- Pannes matérielles
  - température excessive, coupure de courant, etc
- Minuteur système, ou en VO **System Timer**
  - aka *Programmable interval timer*
  - interruptions périodiques, typiquement 100Hz ou 1000Hz
  - permet à l'OS de **percevoir le passage du temps**
  - bonus : permet au noyau de **reprenre la main** sur les applications (cf chap. 2)
- Évènements logiciels exceptionnels
  - **erreurs** fatales : division par zéro, instruction invalide, etc
  - **trappes** volontaires : appels système (cf diapos suivantes)
  - fautes de pages : constatées par la MMU (cf chap 3)

30/48

## Plan

1. Introduction : définition du terme «Système d'exploitation»
2. Interface entre OS et utilisateur : le shell
3. Interface entre logiciel et matériel : l'architecture
4. Isolation entre noyau et applications : les syscalls
5. Quelques syscalls UNIX incontournables

31/48

## Changement de mode d'exécution : trappes

Problème : comment une application peut-elle invoquer une méthode du noyau ?

**Mauvaise** solution : autoriser les applications à sauter vers les fonctions situées dans le noyau

- destination arbitraire ► failles de sécurité
- passage en mode superviseur ► quand ? comment ?

Solution : se donner une instruction spécialisée pour cet usage

- exemples : TRAP (68k), INT (x86), SWI (ARM), SYSCALL (x64)
- **interruption logicielle** = **trappe** = **exception**
- fonctionnement : similaire aux autres types d'interruption
  - sauvegarde en mémoire l'état du CPU
  - bascule vers mode superviseur
  - branche dans le noyau vers une adresse bien connue

32/48

## Appel système : principe

Appel système, ou en VO system call = syscall

Fonction située dans le noyau, invoquée par un processus utilisateur via une interruption logicielle

**Côté application :**

- l'appel est invoqué avec une instruction TRAP
- indifférent au langage de programmation utilisé
- encapsulation dans des fonctions de bibliothèque (ex : libc)

**Côté noyau :**

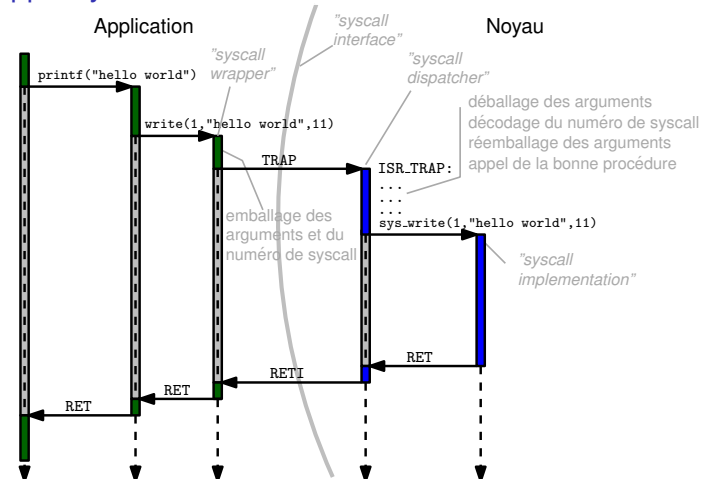
- on passe à chaque fois par l'ISR de TRAP
- qui appelle la bonne fonction dans le noyau,
- puis qui rend la main à l'application avec RETI

Exemples :

- read(), write(), fork(), gettimeofday()...
- plusieurs centaines en tout sous Linux

33/48

## Appel système : déroulement



34/48

## Appel système : langage machine

```
00410700 <__libc_read>:
410700: f3 0f 1e fa      endbr64
410704: 80 3d 6d f3 08 00 00  cmpb $0x0,0x8f36d(%rip)
41070b: 74 13            je 410720 <__libc_read+0x20>
41070d: 31 c0            xor %eax,%eax
41070f: 0f 05            syscall
410711: 48 3d 00 f0 ff ff  cmp $0xfffffffffff000,%rax
410717: 77 4f            ja 410768 <__libc_read+0x68>
410719: c3              ret
41071a: 66 0f 1f 44 00 00  nopw 0x0(%rax,%rax,1)
410720: 55              push %rbp
...

004107a0 <__libc_write>:
4107a0: f3 0f 1e fa      endbr64
4107a4: 80 3d cd f2 08 00 00  cmpb $0x0,0x8f2cd(%rip)
4107ab: 74 13            je 4107c0 <__libc_write+0x20>
4107ad: b8 01 00 00 00    mov $0x1,%eax
4107b2: 0f 05            syscall
4107b4: 48 3d 00 f0 ff ff  cmp $0xfffffffffff000,%rax
4107ba: 77 54            ja 410810 <__libc_write+0x70>
4107bc: c3              ret
```

35/48

## Notion de processus

Applications exécutées sur la «**machine virtuelle userland**» :

- jeu d'instructions restreint (CPU en mode utilisateur)
  - pas accès au mécanisme d'interruptions
- accès **interdit** à certaines adresses mémoire
  - ex : code et données du noyau, périphériques matériels

Protection par «**sandboxing**» : une nouvelle instance de cette machine virtuelle pour chaque application en cours d'exécution

- **CPU virtuel** (cf chap 2), **mémoire virtuelle** (cf chap 3)
- périphériques : accessibles seulement au travers du noyau

**Notion de processus (ou en VO process)**

« Un programme en cours d'exécution »

Système d'exploitation = illusionniste (VM) + sous-traitant (HW)

36/48

## Notion de processus : remarques

### Intuitions :

- un processus = un programme + son état d'exécution
- état d'exécution = valeurs des registres + contenu mémoire + «contexte d'exécution»

### Le noyau :

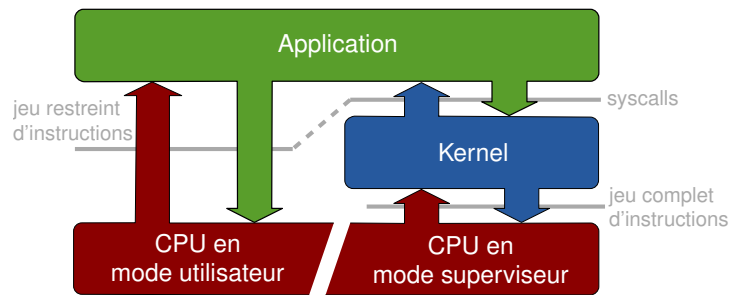
- partage les ressources matérielles entre les processus
- crée/recycle les processus lorsqu'on lui demande
  - dans le noyau : un **Process Control Block** par processus vivant
  - PCB = carte d'identité du processus ► stocke le **contexte** : numéro (**PID**), répertoire courant, liste des fichiers ouverts...

### À faire chez vous :

- essayer les commandes `ps aux` et `top`
  - puis `man ps` et `man top`
- et aussi `strace ./monprogramme`

37/48

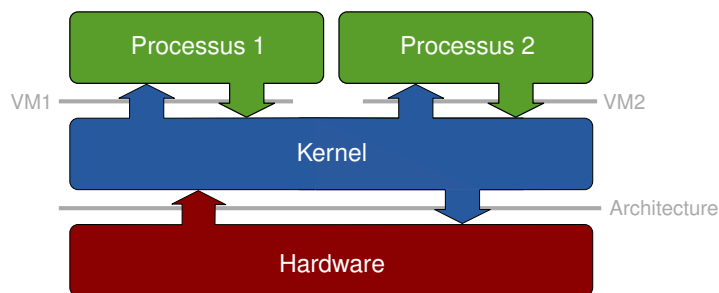
## La VM userland : en résumé



- code applicatif exécuté par CPU en mode utilisateur
- pour faire appel au noyau : interface des appels système

38/48

## Positionnement de l'OS



- chaque application qui s'exécute est un **processus** userland
- le **noyau** virtualise et arbitre les accès au matériel

39/48

## Plan

1. Introduction : définition du terme «Système d'exploitation»
2. Interface entre OS et utilisateur : le shell
3. Interface entre logiciel et matériel : l'architecture
4. Isolation entre noyau et applications : les syscalls
5. Quelques syscalls UNIX incontournables

40/48

## Appels système : exemples

EXAMPLES OF WINDOWS AND UNIX SYSTEM CALLS		
	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

source : Silberschatz. *Operating Systems Concepts Essentials* (2011). p 59

41/48

## Une fonction qui cache un syscall : getpid()

Pour connaître notre numéro de processus

```
#include <unistd.h>

int getpid(void);
```

### Remarques :

- le noyau donne un **numéro unique** à chaque processus
- attribué à la **création** du processus. ne change jamais ensuite.
- stocké dans le PCB du processus

42/48



## Une fonction qui cache un syscall : exit()

Pour cesser définitivement l'exécution du programme

```
#include <stdlib.h>

void exit(int status);
```

Remarques :

- l'exécution ne «revient jamais» d'un appel à exit()
- exit(n) équivalent à un return n depuis le main()
- le «exit status» n est transmis au processus parent
  - convention : 0=OK, 1-255=erreur
  - stocké dans le PCB en attendant un syscall wait() du parent

43/48

## Une fonction qui cache un appel système : fork()

Pour dupliquer le processus en cours

```
#include <unistd.h>

int fork(void);
```

Remarques :

- sous unix : créer un processus ≠ changer de programme
- fork() duplique le processus qui a invoqué le syscall
  - processus d'origine = «parent», nouveau = «enfant»
- **duplication** intégrale de la machine virtuelle userland
  - CPU virtuel : les deux processus s'exécutent **en concurrence**
  - mémoire virtuelle : chacun s'exécute dans un **espace privé**
  - contexte : même répertoire courant, mêmes fichiers ouverts...

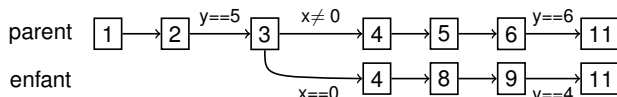
Paradigme «Call once, return twice»

- dans le nouveau processus fork() renvoie 0
- dans le parent, fork() renvoie le PID de l'enfant

44/48

## Appel système fork : illustration

```
1 // only one process
2 int y = 5 ;
3 int x = fork();
4 if ( x != 0 ) {
5     // parent only
6     y = y + 1;
7 } else {
8     // child only
9     y = y - 1;
10 }
11 // both processes
```



45/48

## Mon premier interpréteur de commandes

```
char command[...];
char params[...];
main() {
    while(true) {
        print_prompt();
        read_command(&command, &params);
        x = fork();

        if (x == 0) { // we are the child process
            exec(command, params);
        } else { // we are the parent process
            wait(&status);
        }
    }
}
```

46/48

## Encore des appels système

D'autres exemples, qu'on reverra en TP :

exec(filename) charger un autre programme (exécutable)  
dans mon processus

sleep(int num) suspendre l'exécution de mon processus  
pendant num secondes

wait(...) «attendre» qu'un de mes processus enfants ait  
terminé son exécution

getppid() connaître le PID de son processus parent

open(), read(), write() accéder aux fichiers

47/48

## À retenir

### Architecture

- cycle de Von Neumann, MMIO, DMA, interruptions
- CPU avec *dual-mode operation* : mode restreint vs privilégié
- instruction TRAP pour lever une interruption

### Noyau

- l'ensemble des routines de traitement d'interruption (ISR)
  - en particulier le *syscall dispatcher*
- et des fonctions appelées par les ISR
  - en particuliers les *drivers* et les impléms des syscalls

### Processus

- une «machine virtuelle» offerte aux applications
- vue simplifiée et restreinte de l'architecture

### Appels système

- interface entre les processus et le noyau
- accessibles via des fonctions de bibliothèque

OS = noyau + bibliothèques + programmes utilitaires

48/48