Appels systèmes

François Trahay

Qu’est ce qu’un système d’exploitation ?

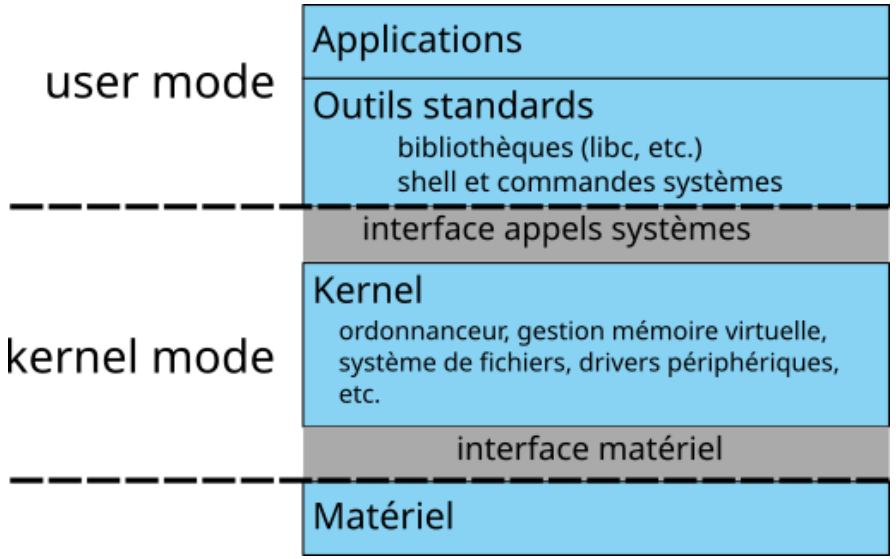
Rôles d’un système d’exploitation:

- Abstraire le matériel pour le programmeur
  - Cacher la complexité du matériel
  - Fournir une interface virtuelle de la machine
- Protéger
  - Protection entre utilisateurs (droits d’accès aux fichiers, espaces mémoires des processus séparés)
  - Protection du matériel
- Partager les ressources
  - Partage du CPU (ordonnancement des processus)
  - Accès concurrents à un périphérique

User mode vs. Kernel mode

Cloisonnement entre le mode utilisateur et le mode noyau

- *User mode*
  - certaines instructions sont interdites
  - pas d'accès aux périphériques
  - accès à l'espace mémoire virtuel du processus
- *Kernel mode*
  - accès aux périphériques
  - accès à la mémoire physique



Comment passer en mode noyau ?

2 méthodes:

- interruption
  - interruption logicielle
    - Générée par le processeur en exécutant une instruction
    - division par zéro, accès mémoire illicite
  - interruption générée par le matériel (IRQ)
    - “Je viens de recevoir un message.” – la carte réseau
    - “J’ai fini de copier les données sur le disque dur.” – le moteur DMA
- appel système
  - l’utilisateur demande à l’OS un service

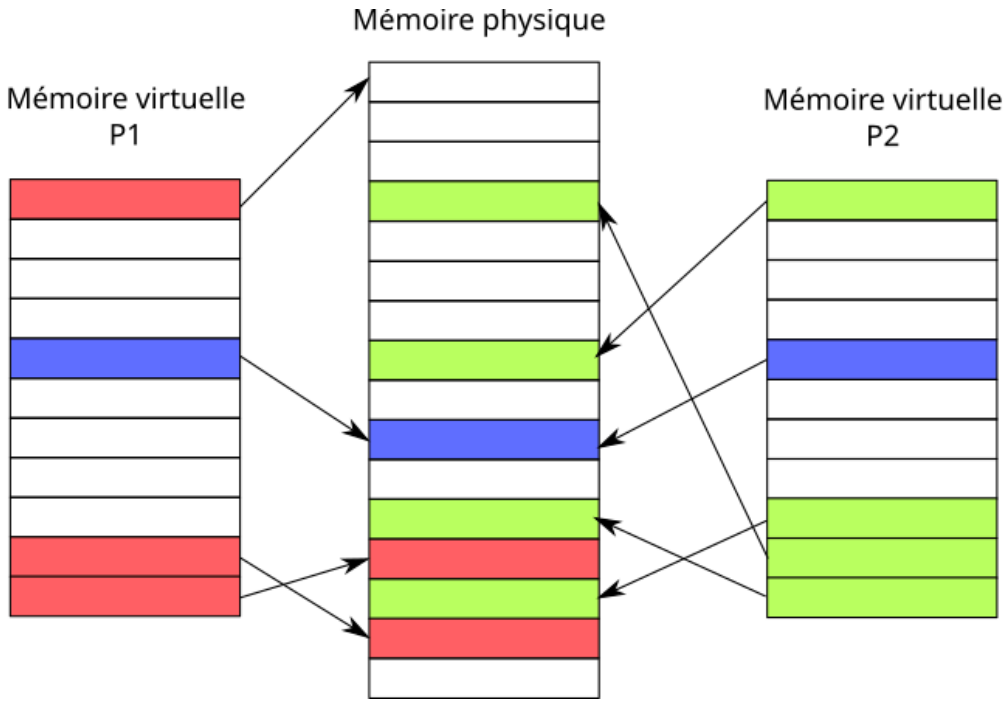
Observer les appels systèmes

La commande `strace` intercepte et affiche les appels systèmes d'un programme:

```
$ strace echo "coucou"
execve("/bin/echo", ["echo", "coucou"], [/* 54 vars */]) = 0
brk(NULL)                               = 0x25d2000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f619cc01000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("tls/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
open("tls/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
open("x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
open("libc.so.6", O_RDONLY|O_CLOEXEC)   = -1 ENOENT (No such file or directory)
open("/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
[...]
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 2), ...}) = 0
write(1, "coucou\n", 7coucou
)                 = 7
close(1)                                = 0
close(2)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

Gestion de la mémoire

- mémoire virtuelle des processus découpées en *pages*
- mémoire physique (RAM) découpée en *cadres de pages*
- *pages* projetées sur des *cadres de pages*



Primitives de synchronisation: les sémaphores

Sémaphore:

- distributeur de “*jetons*”
- 2 opérations:
  - **P** (*“Puis-je”*): prendre un jeton (et attendre si pas de jeton)
  - **V** (*“Vas-y”*): ajouter un jeton (et débloquer un processus)
- Exemple d’utilisation: exclusion mutuelle entre processus

Sémaphore: mise en oeuvre

- Création: `sem_open("/CLE", O_CREAT, S_IRWXU, nb_jetons);`
  - retourne un `sem_t*`
  - CLE est une chaîne commençant par `/`
- Ouverture: `sem_open("/CLE", 0);`
  - retourne un `sem_t`
- Destruction : `sem_unlink(const char *name)`
- Opération **P** : `sem_wait(sem_t* sem)`
- Opération **V** : `sem_post(sem_t* sem)`