

[SYS1] Systèmes d'exploitation (1)

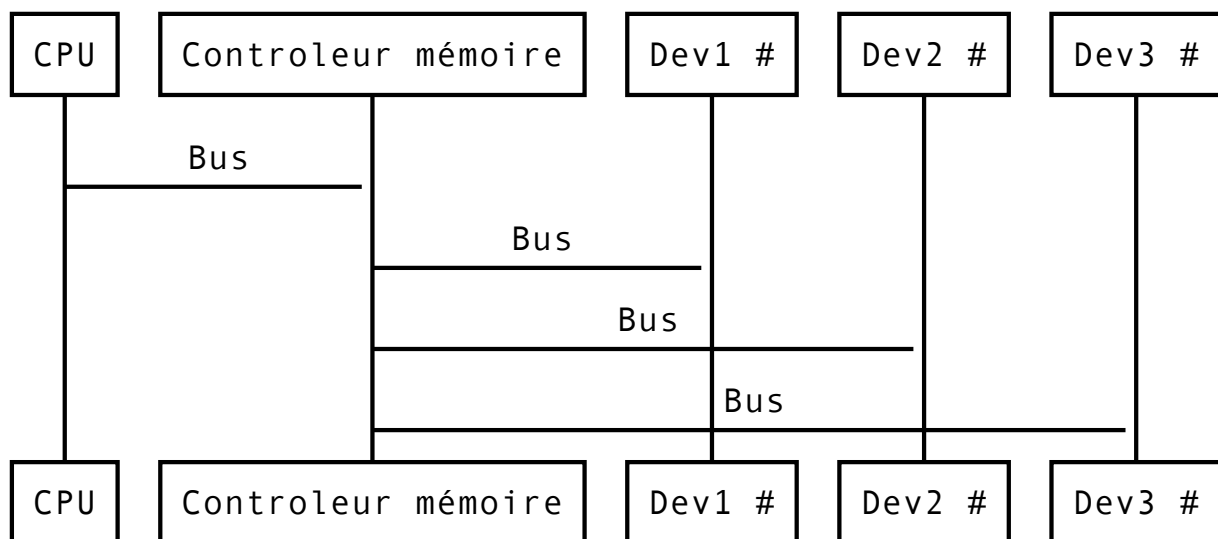
Par Rhaeven, relecture par Find3r

Système d'exploitation

- Gestion des ressources pour les programmes
- Fournir des interfaces et des abstractions pour les programmes utilisateurs

Architecture d'un ordinateur

CPU
RAM
Dev
Dev
...



CPU : Exécute des instructions

- CPU modes
 - *User mode* (ring3)
 - *Supervisor mode* (ring0) : Accès aux registres de controle

Registres : Petites zones de mémoire temporaires (8 bytes)

- *Registres généraux*
- *Registres de controle*
 - permettent de configurer le processeur et la machine en général

Kernel (Noyau du système d'exploitation)

- Fournit des interfaces et des abstractions pour les programmes utilisateurs via des **syscalls (appels systèmes)**
- Quand on fait un syscall, il y a un **context switch** (avec changement de mode, user -> supervisor)
- Syscalls très spécifiques à l'OS, on n'exécute donc pas de syscalls directement mais on utilise des **wrapper** (la plupart du temps exposé par la libc)

Kernel = un seul process qui tourne, comment faire pour reprendre la main pendant l'exécution des programmes et gérer leur scheduling ?

Fichiers

i-node ou inode : Contient toutes les informations d'un fichier

- *Emplacement des datas*
- *Date de création, date de modification*
- *Taille*
- *Permissions, uid, gid*
- Le *type* de fichier
 - regular
 - directory
 - symbolic links
 - devices : char device & block device
 - socket
 - named pipe (FIFO)

Informations récupérables avec **stat/fstat**

Numéro d'inode : Adresse qui permet de retrouver le fichier sur le disque

On se balade dans un directory avec `opendir`, `readdir`, `closedir`

Comment on exécute un programme ?

Programme : fichier qui contient du code

Process : Un programme chose qui s'exécute (la plupart du temps) -> *programme en cours d'exécution*

Sous Linux, on ne sait pas *créer* un nouveau process

syscall fork() : duplique le process courant

syscall wait(), waitpid() : Attend que le process dupliqué se termine

syscall execve() : Conçoit un nouvel espace mémoire dans le process courant

```
pid_t pid = fork();
if (pid < 0)
    err(1, "unable to fork.");

if (pid)
{
    int rc = waitpid(pid/*...*/);
    // error checking/retry/etc
    return;
}

char *argv[] =
{
    "rm",
    "toto.pyc",
    NULL
};

execvp(argv[0], argv);
err(1, "error"); // if execve is successful, this should not be read
```