OS et processus

1. Les systèmes d'exploitation

Définitions

Le **système d'exploitation** (noté SE ou OS, abréviation du terme anglais Operating System), est chargé d'assurer la liaison entre les ressources matérielles, l'utilisateur et les logiciels (traitement de texte, jeu vidéo...).

Le système d'exploitation en dissociant les logiciels et le matériel, mais aussi en proposant une interface conviviale (IHM) permet à l'utilisateur un accès relativement facile à une machine très complexe.

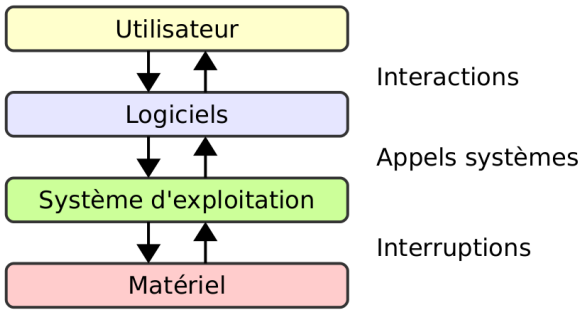


Figure 1: Rôle d'un OS dans une machine

Un peu d'histoire

Deux vidéos d'introduction de Rémi Sharrock (enseignant Telecom ParisTech) sur l'évolution des systèmes d'exploitation et notamment UNIX :

- https://www.youtube.com/watch?v=4OhUDAtmAUo (Histoire des OS)
- https://www.youtube.com/watch?v=bdSWj7Y50VY (Histoire UNIX)

Rôles

Selon son environnement le système d'exploitation peut avoir en charge :

- Gestion du **processeur** : il gère l'allocation du processeur entre les différents programmes grâce à un ordonnanceur.
- Gestion de la **mémoire vive** : il gère l'espace mémoire alloué à chaque application et à lui même. Grâce au MMU (Memory Management Unit) ces zones sont cloisonnées pour éviter qu'un programme écrive dans la zone d'un autre. En cas d'insuffisance de mémoire physique, le système d'exploitation peut créer une zone mémoire sur le disque dur (swap).
- Gestion des **entrées/sorties** : le système d'exploitation permet d'unifier et de contrôler l'accès des programmes aux ressources matérielles par l'intermédiaire des pilotes
- Gestion de l'exécution des **applications** : le système d'exploitation est chargé de la bonne exécution des applications en leur affectant les ressources nécessaires à leur bon fonctionnement. Il permet à ce titre de «tuer» une application ne répondant plus correctement.
- Gestion des **droits** : il est chargé de la sécurité liée à l'exécution des programmes en garantissant que les ressources ne sont utilisées que par les programmes et utilisateurs possédant les droits adéquats (lecture, écriture, exécution, ...).
- Gestion des **fichiers** : le système d'exploitation gère la lecture et l'écriture dans le système de fichiers et les droits d'accès aux fichiers par les utilisateurs et les applications.

Composants

Les différents composants d'un système d'exploitation :

- ✗ Le **noyau** (en anglais kernel) représentant les fonctions fondamentales du système d'exploitation telles que la gestion de la mémoire, des processus, des fichiers, des entrées-sorties principales, et des fonctionnalités de communication.
- ✗ Le **shell** (en anglais shell, traduisez «coquille» par opposition au noyau) permettant la communication avec le système d'exploitation par l'intermédiaire de :
  - ➔ **CLI** (Command Line Interface) : interpréteur de commandes permettant à l'utilisateur via des commandes et un langage de script de dialoguer avec l'OS.
  - ➔ **GUI** (Graphical User Interface) : interface graphique qui avec son pointeur, ses fenêtres, ses icônes, ses boutons permet à l'utilisateur une manipulation aisée et conviviale de la machine

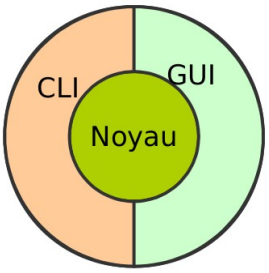
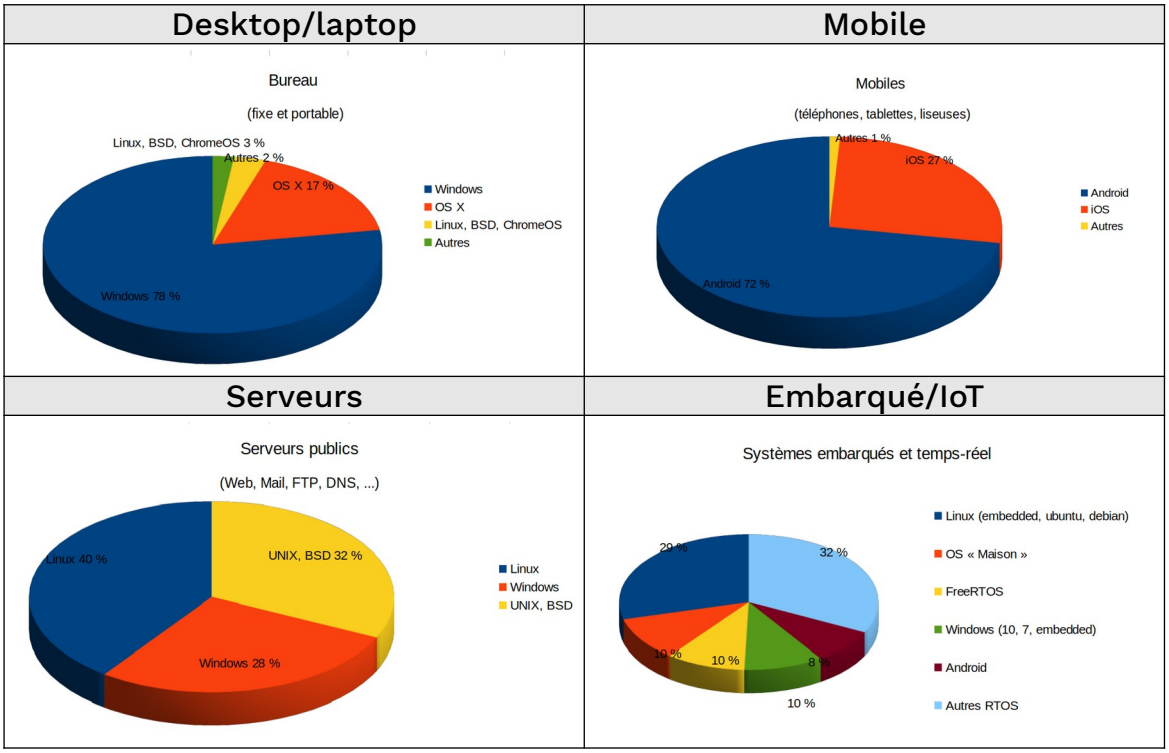


Figure 2: Composants d'un OS

Statistiques d'utilisation

Les systèmes d'exploitation et leurs usages en quelques chiffres :



2.Les processus

Un processus est un programme (exécutable) qui est en cours d'exécution par un ordinateur. Il dispose de ses propres ressources qui sont allouées par le système d'exploitation comme des cycles de processeur, la mémoire vive, les fichiers (entrées-sorties, sockets réseau...).

Programme exécutable

Pour qu'un programme exécutable puisse être exécuté, il doit respecter un format de fichier pris en charge par le noyau, qui est le cœur du système d'exploitation gérant toutes les ressources.

Le format utilisé dépend du système d'exploitation. Parmi les plus connus on trouve :

- ELF (Executable and Linkable Format) sous Linux
- PE (Portable Executable) sous Windows
- Mach-O (Mach-object) sous Mac OS X

Composants d'un processus

Un processus est constitué :

- d'un ensemble d'**instructions** à exécuter (section code)
- d'un espace d'adressage en **mémoire vive** pour travailler (sections pile, tas et data)
- de **ressources** (fichiers ouverts, socket réseaux, connexion bdd...)
- d'un **environnement** (PID, répertoire, utilisateur, droits, priorité, temps, processus parent, variables d'environnement, états des registres CPU...)
- des **flux** d'entrée (stdin) et de sortie (stdout, stderr) pour communiquer avec l'extérieur

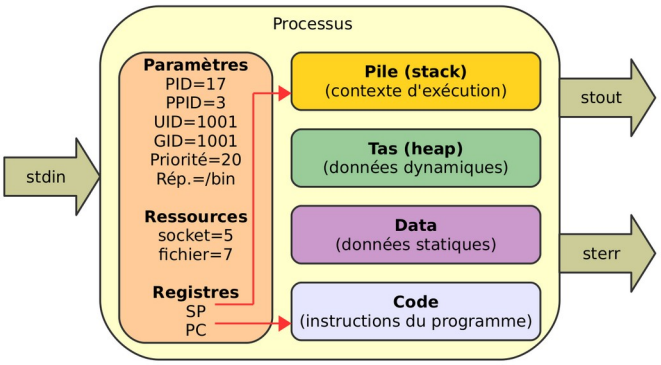


Figure 3: Contexte d'un processus

Le détail d'un processus peut être obtenu sous Linux avec la commande ps

Arborescence de processus

Au lancement du système d'exploitation, un premier processus est créé, il sera l'ancêtre de tous les autres. Il se nomme `init` et son `PID=1`
Ensuite le système d'exploitation va créer des processus « fils » à partir du processus « père » init de 2 types :

- les processus **démon** (service sous Windows) qui tournent en continu
- les processus **utilisateurs** lancés à partir du shell

La commande `pstree` (Process Explorer sous Windows) permet de visualiser cette arborescence :

```
$ pstree
```

État d'un processus

Lorsqu'un processus sollicite une ressource (par exemple la lecture d'un fichier) et que celle-ci n'est pas immédiatement accessible (le disque dur est déjà en cours d'utilisation), le processus va être mis en sommeil un certain temps pour être ensuite réveillé. Un processus passe donc par différents états illustrés par le diagramme suivant :

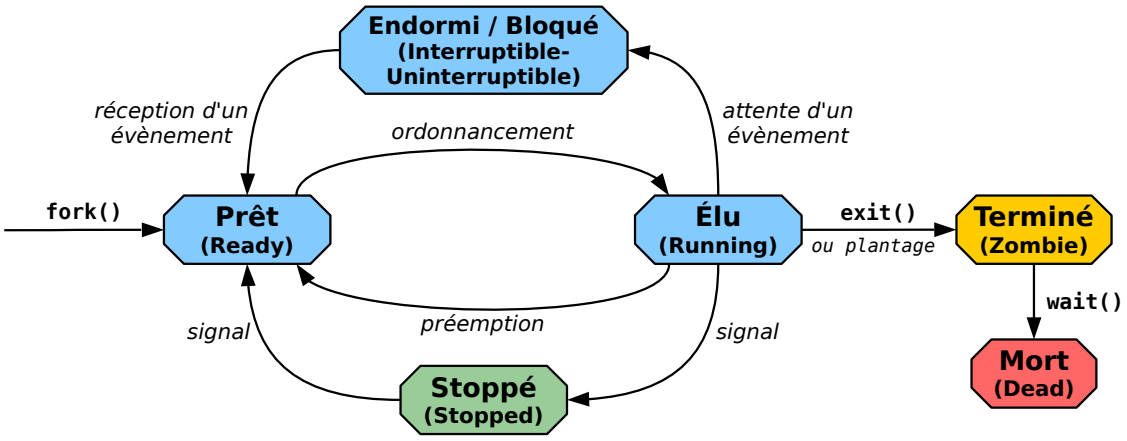


Figure 4: Etats d'un processus

Au début, le processus nouvellement créé est placé dans l'état **Prêt**, il est alors inséré dans une file d'attente par l'ordonnanceur chargé de gérer l'ordre d'exécution des processus.

Lorsque vient son tour, le processus passe à l'état **Élu**, il est alors exécuté un certain temps (conséquence du multitâche) avant d'être replacé dans l'état **Prêt** ou dans l'état **Endormi / Bloqué** s'il sollicite le système d'exploitation (entrée/sortie, libération de ressource...) auquel cas son réveil interviendra lorsque la réponse sera reçue.

Lorsque le processus se termine normalement ou non, il passe à l'état **Terminé** et un code est envoyé à son processus parent. Ce n'est que lorsque le processus parent relève ce code de retour que le processus fils est détruit : ses ressources sont libérées et il est supprimé de la mémoire.

Remarque L'état **Stoppé** apparaît lorsqu'un processus est par exemple tracé par un débogueur ou lorsqu'il reçoit un signal STOP comme lors d'un CTRL+Z.