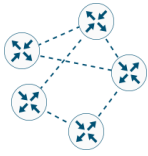


# Routage dynamique des données



Le nombre de routeurs dans un réseau maillé est généralement trop grand pour envisager de configurer les tables de routage à la main. En effet, chaque fois qu'un élément du réseau tombe en panne ou qu'une modification est apportée à sa topologie (ajout d'une nouvelle liaison ou d'un nouveau routeur), il est nécessaire de recalculer toutes les routes et de mettre à jour les tables de routage de chaque routeur. Pour que cela soit possible, il faut également que toutes les données relatives à l'état des liaisons et des routeurs soient envoyées vers un unique opérateur qui doit alors se charger de calculer les nouvelles routes. Outre les inconvénients de centraliser cette tâche, il faut aussi s'assurer que les informations relatives à l'état du réseau puissent être envoyées sans problème à cet opérateur.

Pour toutes ces raisons, on a cherché à automatiser ce processus en laissant les routeurs se charger eux-mêmes de mettre à jour leur table de routage, sans aucune intervention humaine. Ainsi, en plus de la transmission des paquets, les routeurs s'échangent les informations dont ils disposent sur les routes du réseau, en fonction de l'état de leurs voisins et de leurs liens de communication. Les règles à suivre pour réaliser ces échanges sont définies par un protocole de routage.

## 1. Le protocole RIP

Le protocole RIP (**R**outing **I**nformation **P**rotocol) est historiquement le premier algorithme de routage. Le principe du protocole RIP est le suivant : chaque routeur transmet à ses voisins les adresses de ses propres voisins ou celles qu'il a reçu par d'autres routeurs. En plus des adresses, le routeur indique la distance, exprimée en nombre de sauts, qui le sépare d'une machine donnée, c'est-à-dire combien de routeurs il faut traverser pour atteindre cette machine.

### **Propagation des informations**

Chaque routeur a d'abord dans sa table les réseaux directement accessibles sans passer par un autre routeur (donc une distance 0).

Ensuite, périodiquement (toutes les 30 s), chaque machine envoie une requête RIP à ses voisins et reçoit en retour un accusé de réception, cette réponse étant composée de la table de routage de l'émetteur.

Dès réception de la table, le routeur met à jour ses propres tables en suivant les règles suivantes :

- x si des chemins plus courts ou de nouvelles destinations apparaissent. Les distances sont mises à jour ainsi que le nom du premier routeur qu'il faut joindre pour accéder au réseau (distance + direction = vecteur).
- x si un réseau n'apparaît plus dans les annonces, au bout d'un certain temps (3 minutes) il est supprimé des tables.



Les trois caractéristiques principales qui distinguent l'algorithme RIP de l'algorithme alternatif OSPF sont :

- la distance est mesurée en nombre de sauts ;
- chaque routeur n'a d'information que sur ses voisins (en terme de saut : next hop) donc n'a pas de vision globale du réseau (on parle de routing by rumor) ;
- il y a une distance maximale permise de 15 sauts et les tables possèdent 25 entrées maximum .

## 2. Le protocole OSPF

Le protocole RIP n'est pas adapté aux grands réseaux car il ignore les routes de 15 sauts, ceci afin de limiter son délai de convergence et pour éviter des boucles de routage. De plus ce protocole ne tient pas compte des débits des liaisons puisque la distance ne tient compte que du nombre de sauts.

Le protocole OSPF a été mis au point pour pallier à ces faiblesses. Son fonctionnement est plus efficace mais plus complexe.

Retenons seulement quelques grands principes :

- Les routeurs ont une «vision» globale du réseau car ils reçoivent des informations de tout le réseau (mais de manière intelligente et efficace). Tous les routeurs ont donc une connaissance identique du réseau.
- Les distances sont mesurées de manière plus fine : on tient compte du nombre de sauts mais aussi du débit de chaque «câble» reliant deux routeurs par exemple, en général c'est le rapport entre une bande passante de référence et celle du câble exprimées dans la même unité. Les débits binaires sont souvent donnés en kbps (kilobits par seconde) ou Mb/s (megabits par seconde).
- Chaque réseau peut être schématisé par un graphe (vision topologique du réseau). Dans l'exemple ci-dessous (fig 1), les routeurs et les switches sont les sommets, leurs liaisons sont les arêtes, les étiquettes des arêtes sont les coûts.
- Les routes les moins coûteuses et sans cycle sont déterminées en appliquant l'algorithme de **Disjkstra**. Chaque routeur devient alors la racine d'un arbre qui contient les meilleures routes.

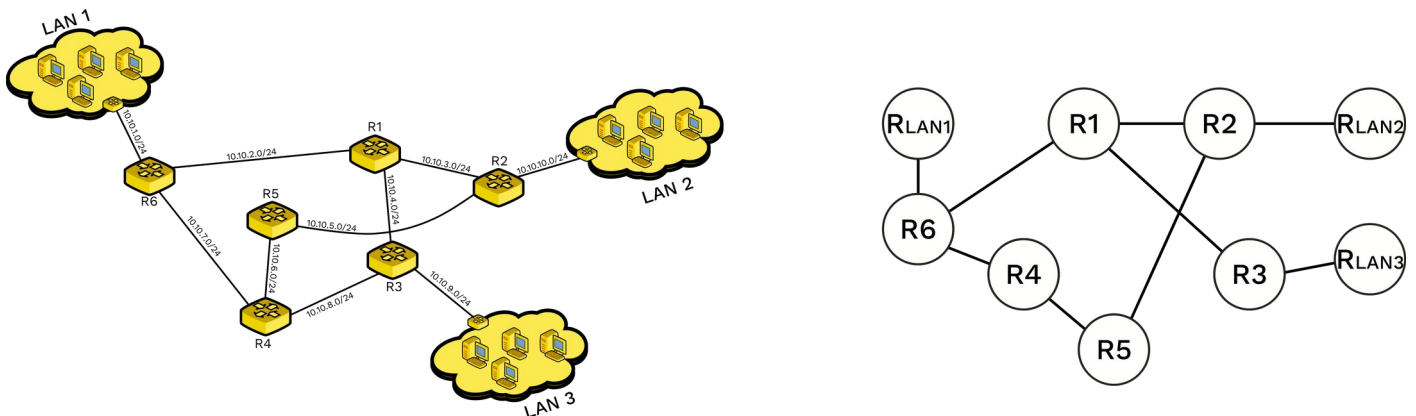


Figure 1: Représentation d'un réseau maillé sous la forme d'un graphe





# L'ordonnanceur

Dans un système multitâches à temps partagé, plusieurs processus peuvent être présents en mémoire centrale en attente d'exécution. Si plusieurs processus sont prêts, le système d'exploitation doit gérer l'allocation du processeur aux différents processus à exécuter. **L'ordonnanceur** s'acquitte de cette tâche.

## 1. Notion d'ordonnancement

Le système d'exploitation d'un ordinateur peut être vu comme un ensemble de processus dont l'exécution est gérée par un processus particulier : l'**ordonnanceur** (scheduler en anglais).

Un ordonnanceur fait face à deux problèmes principaux :

- x le choix du processus à exécuter.
- x le temps d'allocation du processeur au processus choisi.

Les objectifs d'un ordonnanceur d'un système multitâches sont, entre autres :

- x s'assurer que chaque processus en attente d'exécution reçoive sa part de temps processeur ;
- x minimiser le temps de réponse ;
- x utiliser le processeur à 100% ;
- x utiliser d'une manière équilibrée les ressources ;
- x prendre en compte les priorités ;
- x être prédictible.

### Systèmes multitâches préemptifs

Un système d'exploitation multitâche est **préemptif** lorsque celui-ci peut arrêter (réquisition) à tout moment n'importe quelle application pour passer la main à la suivante. Dans les systèmes d'exploitation préemptifs on peut lancer plusieurs applications à la fois et passer de l'une à l'autre, voire lancer une application pendant qu'une autre effectue un travail.

### L'ordonnancement

Il existe plusieurs politiques d'ordonnancement afin d'optimiser la réactivité de la machine ainsi que le taux d'utilisation des processeurs. Parmi les principales techniques, on peut citer :

- x Premier arrivé, premier servi (FIFO) : simple, mais peu adapté à la plupart des situations.
- x Plus court d'abord : très efficace, mais la plupart du temps il est impossible d'anticiper le temps d'exécution d'un processus.
- x Priorité : le système alloue un niveau de priorité aux processus. Cependant des processus de faible priorité peuvent ne jamais être exécutés.
- x Tourniquet : un quantum de temps est alloué à chaque processus. Si le processus n'est pas terminé au bout de ce temps, il est mis en bout de file en état prêt.
- x Un système hybride entre tourniquet et priorité qu'on retrouve dans les systèmes UNIX.



## 2. Interblocage

Le fonctionnement multitâche apporte malheureusement de nouveaux problèmes comme celui de **l'interblocage**.

Imaginez deux processus, le premier s'est accaparé une ressource A et le second une ressource B ; chacun convoite ensuite la ressource de l'autre et n'est prêt à libérer sa propre ressource que si, celle de l'autre devient accessible. On est là face à une situation de blocage mutuel, un interblocage (**deadlock**).

Celle-ci se traduit par la mise en sommeil définitif des deux processus, dans l'attente d'une libération de ressource qui ne surviendra jamais. La figure 1 illustre une situation simple d'interblocage.

Dans un article de 1971, Jr Coffman a établi les conditions d'un interblocage :

- x Au moins une ressource doit être conservée dans un mode non partageable.
- x Un processus doit maintenir une ressource et en demander une autre.
- x Une ressource ne peut être libérée que par le processus qui la détient.
- x Chaque processus doit attendre la libération d'une ressource détenue par un autre qui fait de même.

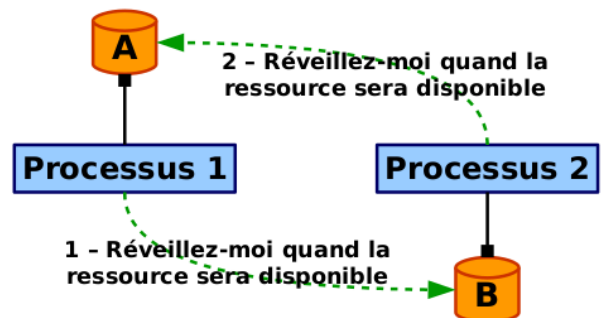


Figure 1: Situation d'interblocage

## 3. Processus Légers (Thread)

Parallèlement aux processus, il existe des processus dits légers qui partagent le même espace d'adressage, hormis la pile qui est propre à chacun. On les appelle des threads. Ils sont d'un usage très courant dans le pilotage des interfaces graphiques, l'encodage vidéo, les calculs scientifiques... car ils permettent de tirer le meilleur parti des micro-processeurs actuels (multithreading) tout en **réduisant le coût de la commutation de contexte**. Du point de vue de l'ordonnanceur ces threads sont considérés comme des processus (tâche) à part entière.



## Les systèmes sur puce (SoC)

Dans un ordinateur "classique" tel qu'un PC de bureau, le « hardware » est organisé autour de 4 éléments principaux :

- x le processeur (CPU – Central Processing Unit) se charge de réaliser les calculs les plus courants, ceux qui permettent par exemple de faire tourner le système d'exploitation ou un navigateur web.
- x la mémoire vive (RAM – Random Access Memory) permet d'enregistrer temporairement les données traitées par le processeur.
- x la carte graphique (ou GPU – Graphics Processing Unit) se charge d'afficher une image, qu'elle soit en 2D ou bien en 3D comme dans les jeux.
- x la carte-mère (Motherboard) permet l'acheminement des données entre les composants (CPU, RAM, GPU, disque dur, SSD, cartes réseau ...) via des « BUS ».

Mais depuis le début de l'ère des smartphones et des tablettes, on assiste à l'émergence de systèmes tout-en-un appelé **SoC (System on a Chip)** afin d'optimiser la miniaturisation et l'intégration des différents composants. Ces derniers sont alors bien mieux interconnectés les uns aux autres, avec par exemple une fréquence processeur qui varie en fonction de la fréquence de la carte graphique du fait de contraintes thermiques et de consommation. Un Soc présente donc une structure complètement inédite par rapport à un ordinateur classique où chaque composant est plus ou moins indépendant .



Figure 1: Puce ARM Exynos - Smartphone Nexus S de Samsung

### 1. Composition d'un SoC

Un système sur une puce (SoC) est un système complet embarqué sur une seule puce ("circuit intégré"), pouvant comprendre de la mémoire, un ou plusieurs microprocesseurs, des périphériques d'interface, ou tout autre composant nécessaire à la réalisation de la fonction attendue. On peut intégrer de la logique, de la mémoire (statique, dynamique, flash, ROM, PROM, EPROM, EEPROM), des dispositifs (capteurs) mécaniques, opto-électroniques, chimiques ou biologiques ou des circuits radio...

Les principaux composants couramment rencontrés dans un SoC sont les suivants :

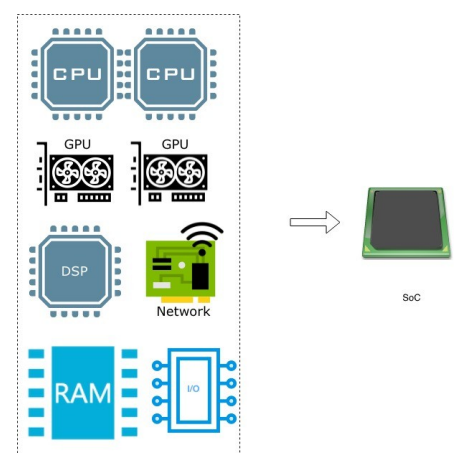


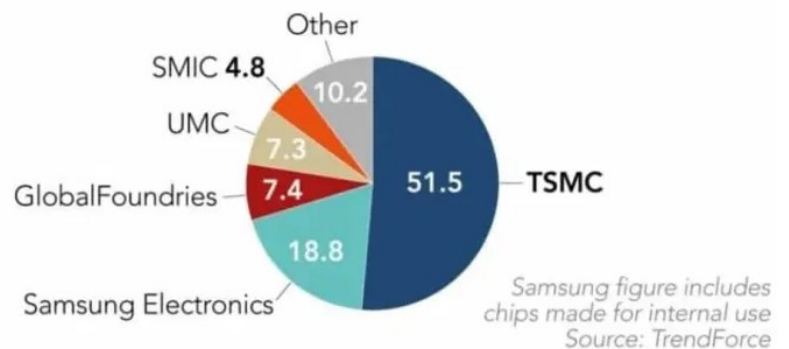
Figure 2: Exemple de composition d'un SoC

Nom	Rôle
<b>CPU</b>	Central Processing Unit : c'est le processeur et chef d'orchestre du SoC comme sur un PC. Il peut être composés de plusieurs cœurs et travaille à une certaine fréquence
<b>GPU</b>	Graphics Processing Unit : en charge de calculer les images affichées à l'écran
<b>ISP</b>	Image Signal Processor : gère les images prises par l'appareil photo
<b>DSP</b>	Digital Signal Processor : gère les signaux en provenance du micro, des accéléromètres, GPS...
<b>Display</b>	Gère l'écran en lien avec le GPU
<b>NPU</b>	Neural Processing Unit : gère tout ce qui est en lien avec le machine learning (reconnaissance vocale, habitudes...)
<b>FPU</b>	Unité de calcul pour nombres flottants (simple ou double précision)
<b>NoC</b>	Gère la communication entre tous les composants
<b>Interface Modem</b>	Interface de communication vers modem 3G/4G/5G, WiFi, Bluetooth...
<b>SPU</b>	Security Processing Unit : gère le cryptage/décryptage des données
<b>Memory</b>	Gère les transferts de données entre CPU et mémoire cache ou mémoire DRAM
<b>Video</b>	Gère le codage/décodage des flux vidéo (MP4)
<b>Audio</b>	Gère le codage/décodage des flux audio (MP3)
<b>Storage</b>	Gère les transferts de données avec la mémoire Flash et/ou la carte SD
<b>GPIO</b>	General Purpose Input Output : entrées/sorties vers boutons, leds

## 2. Marché des SoC



Le marché des SoC est en croissance constante porté par les smartphones. La société TSMC est l'un des principaux fabricants de SoC et fourni notamment : Apple, Broadcom, Qualcomm, MediaTek, AMD, Nvidia...

**Global foundry market share in April-June (In percent)**



### 3. Performances d'un SoC par rapport à système classique

Les systèmes sur puce de part leur conception ont des avantages mais aussi des inconvénients par rapport à une solution traditionnelle type carte mère d'ordinateur

Critère	SoC 	Carte mère PC 
Taille	+++	- - -
Consommation électrique	++	- -
Chaleur dégagée	++	- -
Circulation des données	++	- -
Bruit	++	- -
Adaptation aux besoins spécifiques	+	-
Coût (phase de fabrication)	+	-
Coût (phase de conception)	-	+
Puissance de calcul	-	+
Complexité conception	- -	++
Facilité pour dissiper la chaleur	- -	++
Possibilité de réparation/évolution*	- - -	+++

(\*) concernant l'évolution il existe les PSoc : Programmable SoC qui permettent de pouvoir faire évoluer certaines parties d'un SoC.

#### **Unités utilisées pour comparer les puissances de calcul :**

- ➔ le nombre de transistors
- ➔ le nombre d'instructions exécutées à la secondes (MIPS : Million of Instructions Per Second, GIPS ou TIPS). Souvent utilisé pour les CPU
- ➔ le nombre de calculs effectués par seconde (FLOPS : FLoating-point Operations Per Second). Souvent utilisés pour les GPU
- ➔ les benchmarks





## 4. L'architecture ARM

Dotés d'une architecture relativement plus simple que d'autres familles de processeurs, et bénéficiant d'une faible consommation électrique, les processeurs ARM (Advanced Risc Machine) sont devenus dominants dans le domaine de l'informatique embarquée, en particulier la téléphonie mobile et les tablettes. Les architectures ARM reposent sur des processeurs à **jeux d'instructions réduit RISC** (Reduced Instruction Set Computer) 32 bits (ARMv1 à ARMv7) ou 64 bits (ARMv8).

Aujourd'hui, ARM est surtout connu pour ses systèmes sur puce (SoC), intégrant sur une seule puce : microprocesseur, processeur graphique (GPU), DSP, FPU, SIMD, et contrôleur de périphériques. Ceux-ci sont présents dans la majorité des smartphones et tablettes.

ARM propose des architectures qui sont vendues sous licence de propriété intellectuelle aux concepteurs. Ils proposent différentes options dans lesquelles les constructeurs peuvent prendre ce qui les intéresse pour compléter avec leurs options propres ou de concepteurs tiers. ARM propose ainsi pour les SoC les plus récents les microprocesseurs Cortex (Cortex-A pour les dispositifs portables de type smartphones et tablettes, Cortex-M pour le couplage à un microcontrôleur, Cortex-R pour les microprocesseurs temps réel), des processeurs graphiques (Mali), des bus AMBA sous licence libre, ainsi que les divers autres composants nécessaires à la composition du SoC complet. Certains constructeurs, tels que Nvidia, préfèrent produire leur propre processeur graphique, d'autres, comme Samsung, préfèrent prendre dans certains cas un processeur graphique de prestataire tiers ou d'ARM selon les modèles, et d'autres, comme Apple, modifient certains composants du microprocesseur en mélangeant plusieurs architectures processeur ARM.



Une particularité des processeurs ARM est leur mode de vente. En effet, ARM Ltd. ne fabrique ni ne vend ses processeurs sous forme de circuits intégrés. La société vend les licences de ses processeurs de manière qu'ils soient gravés dans le silicium par d'autres fabricants. Aujourd'hui, la plupart des grands fondeurs de puces proposent de l'architecture ARM.







# 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.

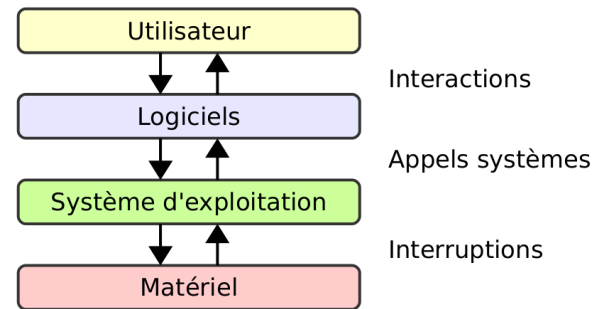


Figure 1: Rôle d'un OS dans une machine

### 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 :

x 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.

x 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

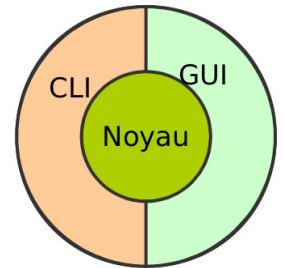
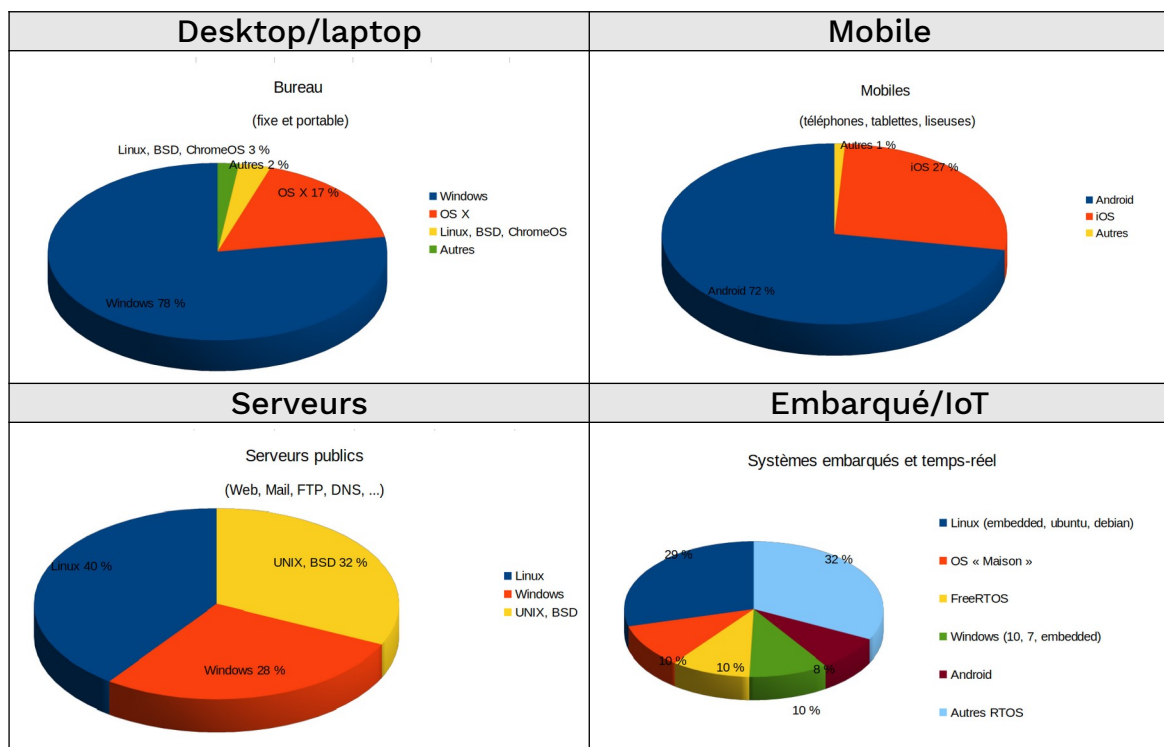


Figure 2:  
Composants d'un  
OS

## 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

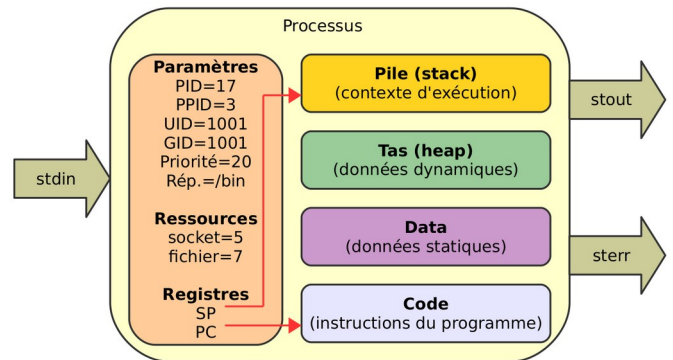


Figure 3: Contexte d'un processus

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 `ps tree` (Process Explorer sous Windows) permet de visualiser cette arborescence :

```
$ ps tree
```

## 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 :



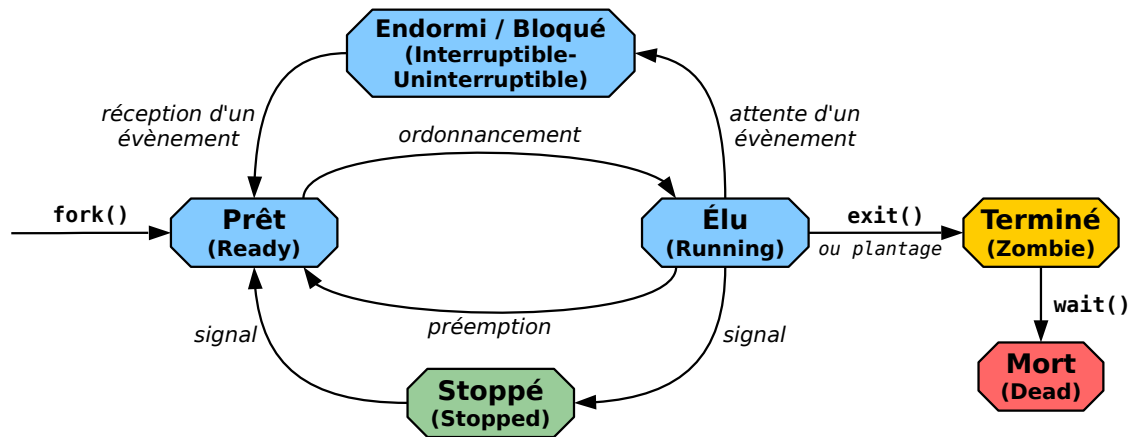


Figure 4: Etats d'un processus

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.