ARCHITECTURE MATERIELLE

ET

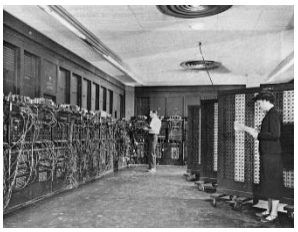
SYSTEMES D'EXPLOITATION

1. Un peu d'histoire

-500 : apparition du boulier

1642 : Pascal invente la Pascaline

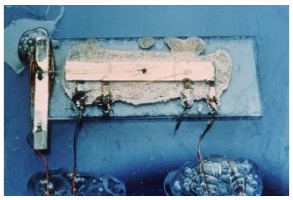
1937 : article d'Alan Turing sur la calculabilité : machines de Turing
1942 : Alan Turing et son équipe mettent au point la machine qui a permis de décoder les messages d'Enigma.



1945 : Définition de l'architecture de Von NEUMANN

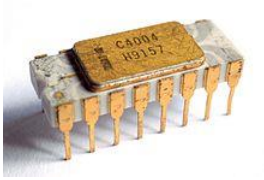
1946 : construction de l' ENIAC

1947 : invention du transistor (Bell)



1958 : premier circuit intégré (Texas Instrument)

1965 : G. Moore énonce la loi qui porte son nom : loi de Moore



1971 : premier microprocesseur : 4004 d'Intel
(4 bits, 108 KHz, 2300 transistors)

1981 : IBM sort le premier PC

1981 : Premier système d'exploitation pour PC : MS-DOS
1991 : Création de LINUX, système d'exploitation libre

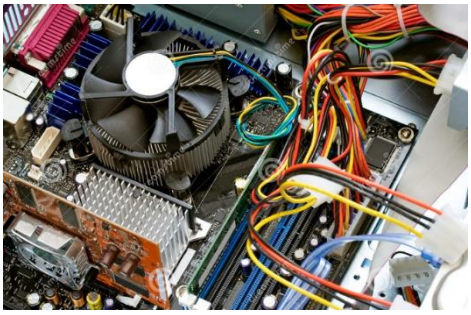
2007 : Création d'Android pour les appareils mobiles

2018 : record de processeur à 7,6 GHz ….probablement dépassé aujourd'hui….

D'après : Pixees, NSI-Ellipses, Irif.fr, Wikipedia, Courstechinfo.be

2. Microprocesseur et mémoire

2.1 L'architecture de base :



Dans l'intérieur d'un PC, on remarque la carte mère qui accueille tous les éléments fondamentaux au bon fonctionnement d'un ordinateur :

- Le **microprocesseur** (CPU) (caché sous un système de refroidissement),
- Les barrettes de **mémoire RAM**,
- La **carte graphique** (qui permet de gérer l'affichage).
- On trouve aussi sur la **carte mère** les **puces** qui gèrent les interfaces réseau (Wifi et Ethernet) et bien d'autres choses…

2.1.1 Le processeur

Un **processeur** (UTC : Unité Centrale de Traitement) ou en anglais (CPU : Central Processing Unit) est un composant qui exécute les instructions des programmes informatiques.

2.1.2 La mémoire

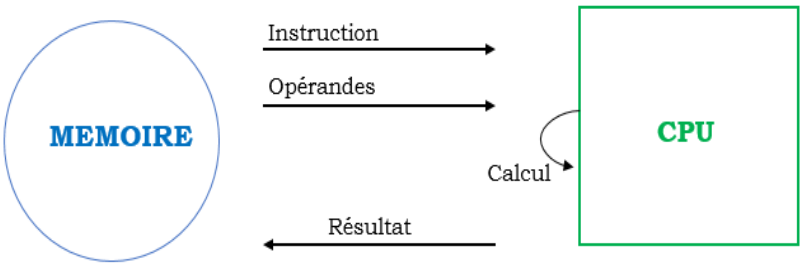
La **mémoire** est un dispositif électronique numérique qui sert à stocker des données de manière plus ou moins statique.

2.1.3 La carte graphique

Une **carte graphique** ou carte vidéo, est une carte d 'extension d'ordinateur dont le rôle est de produire une image affichable sur un écran.
La carte graphique envoie à l'écran les images stockées dans sa propre mémoire, à une fréquence et dans un format qui dépendent d'une part de l'écran branché et du port sur lequel il est branché.

Tout programme est une **suite d'opérations simples** qui ont toutes la même forme. En utilisant les deux éléments (processeur et mémoire), ce principe de base permet l'activité numérique.

(1) Une instruction élémentaire à effectuer est chargée de la mémoire vers le processeur.
(2) Les opérandes (données sur lesquelles va être fait le calcul), sont aussi chargées de la mémoire vers le processeur.
(3) Le calcul de l'opération élémentaire est effectué.
(4) Le résultat de l'opération est stocké en mémoire.



D'après : Pixees, NSI-Ellipses, Irif.fr, Wikipedia, Courstechinfo.be

2.2 Les microprocesseurs

Au niveau technique, le microprocesseur est un circuit électronique intégré qui effectue des opérations. Sa taille est de plus en plus réduite.
La vitesse du microprocesseur est définie par son horloge qui fournit le rythme des opérations élémentaires à effectuer, exprimée en Hz (nombre d'opérations par seconde).

Deux familles de processeurs sont disponibles sur le marché :

- Les processeurs **RISC** (Reduced Instruction Set Computer) qui proposent un nombre restreint d'instructions, qu'il est donc possible d'effectuer efficacement et rapidement.
- Les processeurs **CISC** (Complex Instruction Set Computer) qui disposent d'un nombre d'instructions plus important et plus élaborées, mais qui sont donc moins rapides.

CISC	RISC
S/360 (IBM) VAX (DEC) 68xx, 680x0 (Motorola) x86, Pentium (Intel)	Alpha (DEC) PowerPC (Motorola) MIPS PA-RISC (Hewlett-Packard) SPARC

Si le choix du processeur selon les besoins a donc une importance, mais les évolutions en termes de rapidité permettent désormais de créer des RISC plus puissants dont les performances se rapprochent de celles des CISC.

2.3 Les mémoires

La mémoire est le support, principalement magnétique(disques durs) ou électroniques (RAM, clés usb, disques SSD, ROM) qui contient les programmes et les données.

La mémoire peut être :

a) **Permanente**, les données sont conservées lorsque la machine est éteinte.
Disques durs, clés usb, disques SSD, CD, DVD)

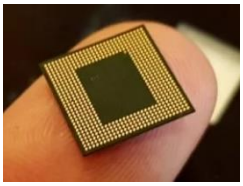
b) **Volatile**, les données ne sont conservées que pour le durée de leur utilisation.
- La **mémoire** vive ou RAM (Random Access Memory : **mémoire** à accès aléatoire) est la **mémoire** la plus connue de l'**ordinateur**. Le processeur l'utilise pour stocker toutes sortes de données dont il a besoin rapidement. Sans **mémoire** vive, un **ordinateur** ne peut pas fonctionner.
- La **mémoire cache** ou antémémoire est une **mémoire** qui enregistre temporairement des copies de données provenant d'une source, afin de diminuer le temps d'accès ultérieur du processeur à ces données.

D'après : Pixees, NSI-Ellipses, Irif.fr, Wikipedia, Courstechinfo.be

- Le **registre** est un emplacement de mémoire interne à un processeur. Les registres se situent au sommet de la hiérarchie mémoire : il s'agit de la mémoire la plus rapide d'un ordinateur, mais dont le coût de fabrication est le plus élevé, car la place dans un microprocesseur est limitée.

On entend souvent dire que les téléphones portables (smartphones) sont de véritable ordinateur, ce qui est vrai. On peut s'interroger sur la taille d'un smartphone par rapport à la taille d'un PC (la carte mère d'un PC mesure environ 25 cm sur 30 cm, soit bien plus qu'un smartphone).
Pourtant on doit obligatoirement trouver dans un smartphone les mêmes composants que dans un PC : CPU, RAM, carte graphique et interfaces réseau (Wifi et Bluetooth dans le cas d'un smartphone) !

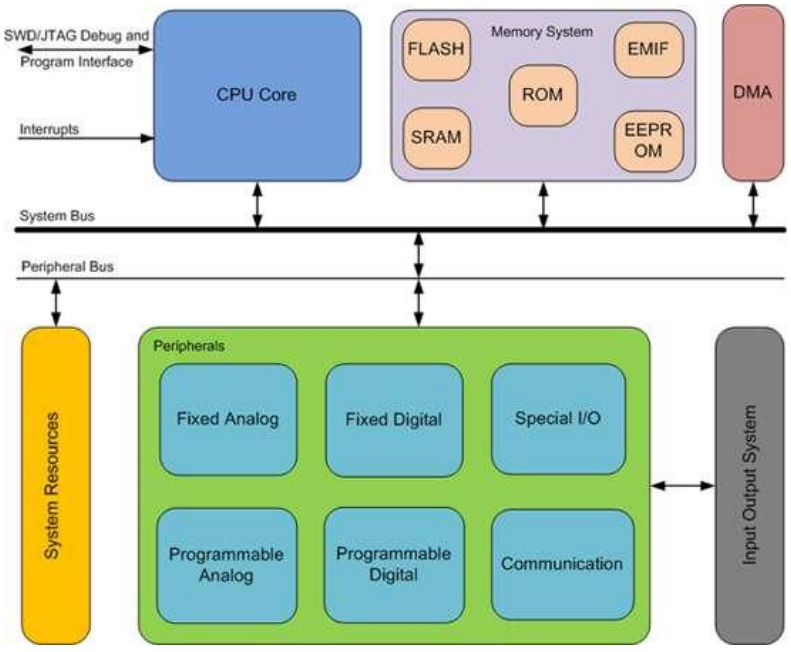
La solution ? Placer tous ces composants dans une puce unique d'une centaine de mm^2



Ces puces accueillant CPU, RAM, circuit graphique (GPU) (équivalent à la carte graphique dans un PC) et circuits radio (Wifi et Bluetooth), sont souvent appelées "système sur puce", "system on a chip" en anglais (abréviation "**SoC**").

3. Les SoCs : Systèmes sur puce

Rendu possible par la miniaturisation permanente des composants électroniques, l'idée de base est d'intégrer dans une seule puce plusieurs éléments, de nature et de rôles différents, pour créer un système autonome capable d'effectuer une tâche spécifique. Ces dispositifs partent le nom de systèmes sur puce : **SoCs**.



Les éléments qui peuvent être intégrés dans un SoC sont nombreux : des microprocesseurs, de la mémoire, des dispositifs de communication sans fil, des dispositifs d'entrée/sortie des capteurs…

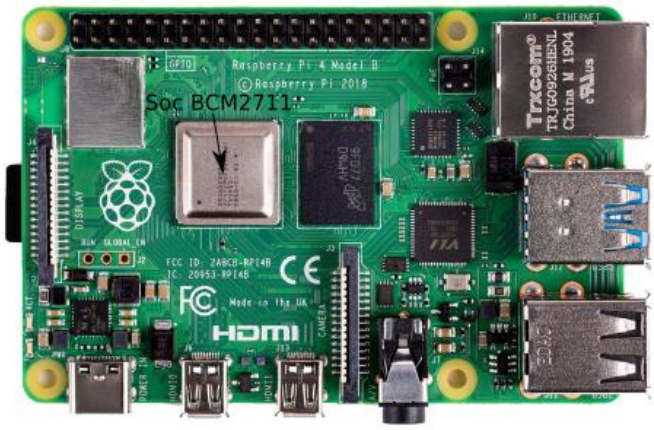
D'après : Pixees, NSI-Ellipses, Irif.fr, Wikipedia, Courstechinfo.be

On peut remarquer que l'on retrouve bien sur le schéma du SoC ci-dessus, un CPU et de la mémoire.

Outre leur taille, les SoCs ont d'autres avantages par rapport aux systèmes "classiques" (carte mère + CPU + carte graphique...) :

- Les SoCs sont conçus pour consommer beaucoup moins d'énergie qu'un système classique (à puissance de calcul équivalente).
- Cette consommation réduite permet dans la plupart des cas de s'affranchir de la présence de système de refroidissement actif comme les ventilateurs ce qui le rend plus silencieux.
- Vu les distances réduites entre, par exemple, le CPU et la mémoire, les données circulent beaucoup plus vites, ce qui permet d'améliorer les performances. En effet, dans les systèmes "classiques" les bus (si nécessaire voir le cours de première : Modèle d'architecture de von Neumann à propos des bus) sont souvent des "goulots d'étranglement" en termes de performances à cause de la vitesse de circulation des données.

On trouve aussi ce système de SoC sur des nano-ordinateurs comme le Rasberry Pi :



Dans sa version 4, le Rasberry Pi est équipé du Soc BCM2711 de la société Broadcom

Voici les noms des SoC utilisés par quelques produits succès :

- Apple iPhone 11 : SoC A13 Bionic
- Samsung Galaxy S11 : SoC Exynos 990
- Nintendo Switch : SoC Nvidia Tegra

Rien qu'à l'évocation des noms cités ci-dessus, vous vous doutez bien que le marché des SoC a aujourd'hui un poids économique très important. On commence même à trouver des ordinateurs portables équipés de SoC à la place des cartes mères "classiques". Les SoC commencent doucement, mais surement à rattraper les systèmes "classiques" en termes de puissance (pour les systèmes "classiques" d'entrées et de moyennes gammes).

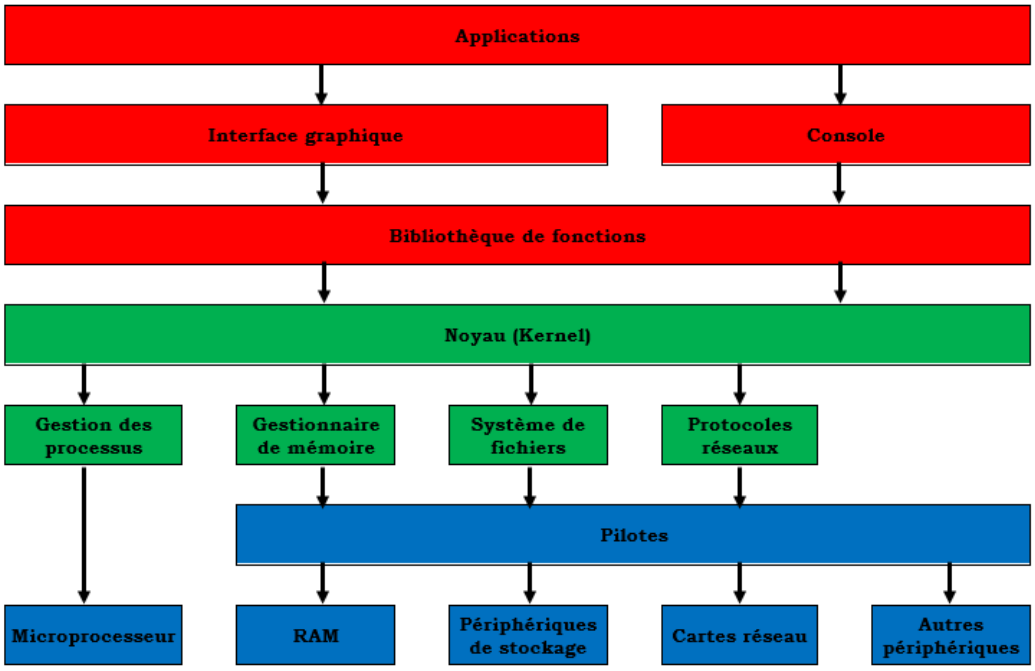
D'après : Pixees, NSI-Ellipses, Irif.fr, Wikipedia, Courstechinfo.be

4. Les systèmes d'exploitation

4.1 Définition

Un système d'exploitation est un ensemble de programmes qui va permettre d'utiliser les éléments physiques d'un ordinateur pour exécuter les applications nécessaires à l'utilisateur.
L'élément fondamental du système d'exploitation est le noyau (kernel), c'est lui qui permet et gère l'accès aux ressources matérielles.

4.2 L'architecture d'un système d'exploitation



4.3 Le fonctionnement du système d'exploitation

Au niveau de l'utilisateur, nous trouvons les **applications** qui sont exécutées via l'**interface graphique** ou depuis la **console de commandes**. Certaines applications peuvent utiliser des **bibliothèques de fonctions**.

Les applications s'appuient sur le **noyau**, élément central du système d'exploitation, qui génère des appels systèmes pour accéder à une ressource.

Selon la nature de la ressource nécessaire, des gestionnaires spécifiques sont sollicités :

- Le **gestionnaire de processus** pour l'exécution d'un programme par le microprocesseur,
- Le **gestionnaire de mémoire** pour l'accès à une donnée en mémoire,
- Le **système de fichiers** pour la gestion des périphériques de stockage de masse (disque dur, DVD…),
- Les **protocoles réseaux** pour les outils de gestion des différents réseaux disponibles.

Chaque ressource est gérée par un **pilote**, seule entité logicielle capable de dialoguer avec les différents périphériques.

D'après : Pixees, NSI-Ellipses, Irif.fr, Wikipedia, Courstechinfo.be

5. Les processus

5.1 Notion de processus

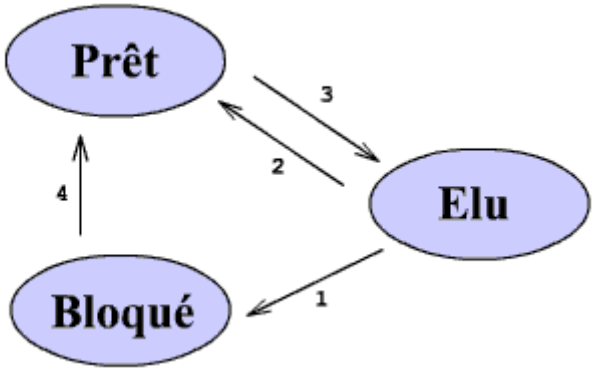
Un programme écrit à l'aide d'un langage de haut de niveau (on parle de "code source") est transformé en langage machine afin de pouvoir être exécuté par un ordinateur.
On appelle processus un programme en cours d'exécution. Attention, il ne faut pas confondre le code source du programme et le processus, qui lui correspond à l'exécution de ce programme par un ordinateur

5.2 Etats d'un processus

Tous les systèmes d'exploitation "modernes" (Linux, Windows, macOS, Android, iOS...) sont capables de gérer l'exécution de plusieurs processus en même temps.

Mais pour être précis, cela n'est pas exactement "en même temps", mais plutôt un "chacun son tour". Pour gérer ce "chacun son tour", les systèmes d'exploitation attribuent des "états" au processus.

Les processus, puisqu'ils sont concurrents et doivent se partager le processeur, ne peuvent être continuellement actifs. Ils ont donc, si on ne considère pour commencer que l'ordonnancement à court terme, trois niveaux fondamentaux (**Elu**, **Prêt** et **Bloqué**), et quelques transitions possibles.



Elu signifie en cours d'exécution. L'exécution n'est interrompue que par les conditions suivantes :

Transition 1 : Le processus se bloque, faute de données pour l'alimenter ou en attendant une opération d'entrée/sortie.

Transition 2 : Le processus est interrompu soit parce que la tranche de temps qui lui est impartie est achevée soit parce qu'un processus de plus haute priorité réquisitionne le processeur.

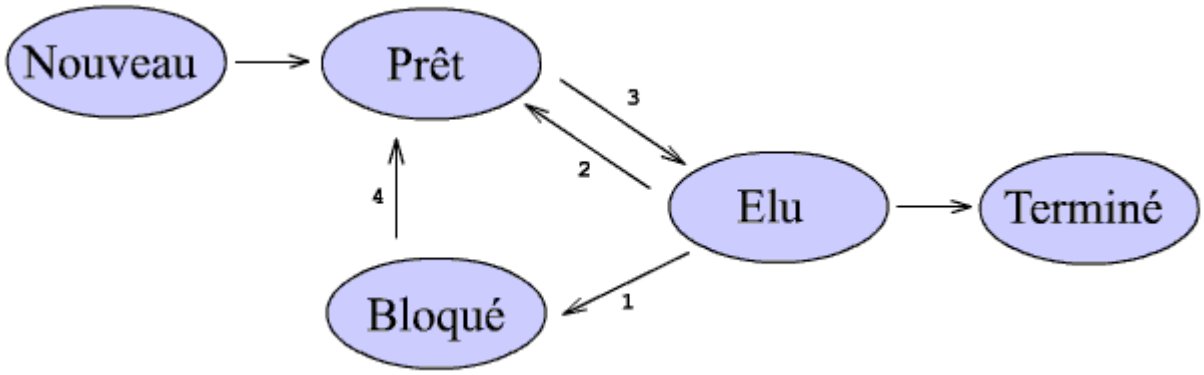
Prêt est un état provisoire pour permettre aux autres processus de s'exécuter quasi simultanément.

Bloqué est un état d'attente d'un événement extérieur, tel qu'une entrée/sortie, nécessaire à la poursuite de l'exécution du processus.

D'après : Pixees, NSI-Ellipses, Irif.fr, Wikipedia, Courstechinfo.be

Ajoutons deux états qui correspondent à l'ordonnancement à long terme :

" Nouveau " et " Terminé ".



Nouveau est un processus qui vient d'être créé mais qui n'existe encore qu'à l'état de requête de processus.
Un processus est toujours créé dans l'état "**prêt**". Pour se terminer, un processus doit obligatoirement se trouver dans l'état "**élu**".

Terminé est un processus désormais inactif car il a achevé sa tâche.

Le passage de l'état "**prêt**" vers l'état "**élu**" constitue l'opération "**d'élection**".

Le passage de l'état "**élu**" vers l'état "**bloqu**é" est l'opération de "**blocage**".

Il est vraiment important de bien comprendre que le "chef d'orchestre" qui attribue aux processus leur état "élu", "bloqué" ou "prêt" est le système d'exploitation. On dit que le système gère l'**ordonnancement des processus** (tel processus sera prioritaire sur tel autre…)

5.3 L'ordonnancement des processus (Scheduling)

Les processus concurrents doivent se partager le processeur, la mémoire et les entrées/sorties.

- Dans les systèmes anciens, les systèmes de traitement par lots mais aussi les systèmes d'exploitation pas vraiment multitâches, l'ordonnancement était de type coopératif. L'ordonnanceur n'intervenait que lorsque le processus en cours se terminait ou se bloquait. Ce système assez sommaire convenait aux traitements par lots quand le temps de réponse n'avait que peu d'importance.

- Actuellement, sur les systèmes interactifs multitâches, parfois même multi utilisateurs et multi processeurs, l'ordonnancement doit être préemptif. L'ordonnanceur ne peut laisser un processus monopoliser les ressources du système et réquisitionne régulièrement le processeur pour en répartir la disponibilité entre les processus qui simultanément sont prêts à être exécutés. La politique suivie pour déterminer la manière d'ordonnancer les processus est fonction de nombreux critères parfois contradictoires. Le fait de favoriser certaines catégories de tâches peut en léser d'autres.

D'après : Pixees, NSI-Ellipses, Irif.fr, Wikipedia, Courstechinfo.be

L'ordonnanceur doit permettre :

- De minimiser le temps de traitement du processus d'un utilisateur
- De garantir l'équité entre les différents utilisateurs
- D'optimiser l'utilisation de la ressource
- D'éviter les blocages

5.4 Les algorithmes d'ordonnancement

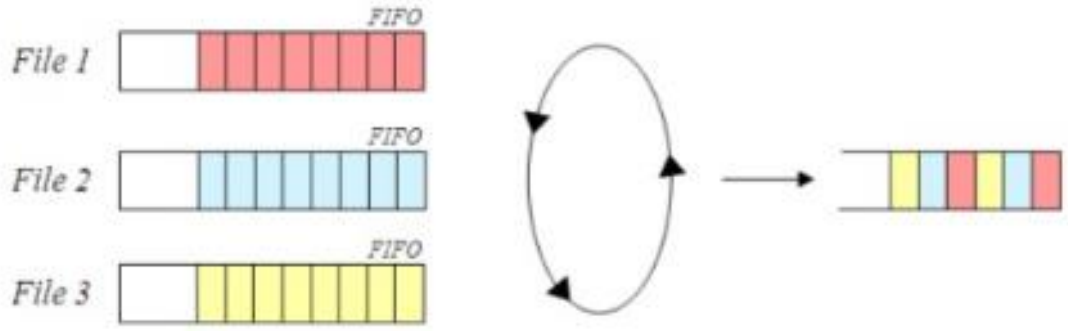
Plusieurs algorithmes d'ordonnancement sont possibles, parmi les plus répandus, nous pouvons citer :

FIFO : First In, First Out = Premier entré, premier sorti
Les jobs attendent dans une file. Le premier arrivé est admis immédiatement et s'exécute tant qu'il n'est pas bloqué ou terminé. Lorsqu'il se bloque, le processus suivant commence à s'exécuter et le processus bloqué va se mettre au bout de la file d'attente.

SJF : Shorted Job First = le travail le plus court d'abord
Sera élu, le processus dont on suppose que le traitement sera le plus court. (C'est ce qui se passe quand à la caisse d'une grande surface les clients laissent passer devant quelqu'un qui n'a qu'un article)
Inconvénient : les jobs les plus courts sont favorisés. Si des processus courts arrivent sans cesse, les processus plus longs n'auront jamais le temps de s'exécuter

L'ordonnancement avec priorité
Une valeur de priorité est assignée à chaque processus. La priorité peut être fonction d'un ordre de préséance entre utilisateurs ou fonction des caractéristiques des processus. Cette méthode est très équitable à condition que la définition du niveau de priorité d'une tâche soit objective.

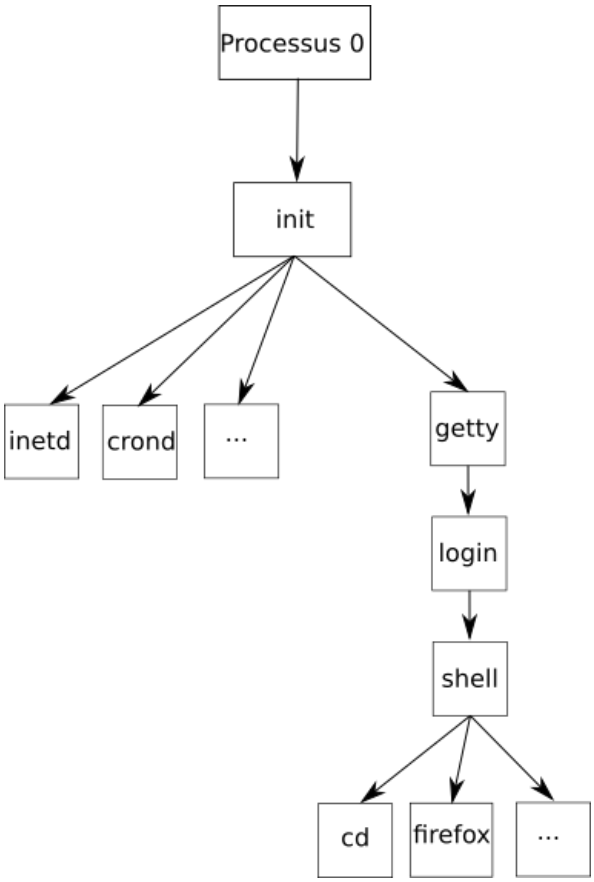
RR - Round Robin = L'algorithme du Tourniquet
Chaque processus reçoit tour à tour un intervalle de temps appelé **quantum**. Au terme de ce quantum ou, si le processus s'achève ou se bloque avant cet instant, l'ordonnanceur attribue directement le processeur au processus suivant. L'algorithme est simple et équitable. C'est généralement cet ordonnancement circulaire qui est utilisé dans les systèmes à temps partagé. La valeur du quantum est couramment fixée aux alentours de 20 à 50 ms.



D'après : Pixees, NSI-Ellipses, Irif.fr, Wikipedia, Courstechinfo.be

5.5 Création d'un processus

Au moment du démarrage de l'ordinateur un tout premier processus (appelé processus 0) est créé à partir de "rien" (il n'est le fils d'aucun processus). Ensuite, ce processus 0 crée un processus souvent appelé "init" ("init" est donc le fils du processus 0). À partir de "init", les processus nécessaires au bon fonctionnement du système sont créés (par exemple les processus "crond", "inetd", "getty",...). Puis d'autres processus sont créés à partir des fils de "init"...



Un processus peut créer un ou plusieurs processus à l'aide d'une commande système. Imaginons un processus A qui crée un processus B. On dira que A est le père de B et que B est le fils de A. B peut, à son tour créer un processus C (B sera le père de C et C le fils de B). On peut modéliser ces relations père/fils par une structure arborescente.

On peut résumer tout cela avec le schéma ci-contre.

5.6 PID et PPID

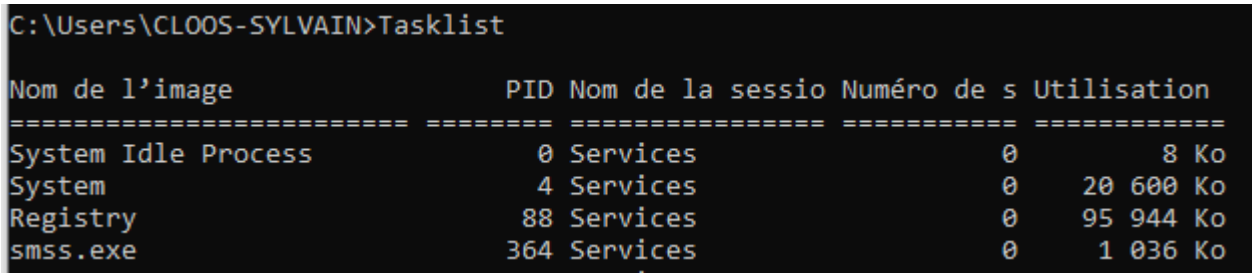
Chaque processus possède un identifiant appelé **PID** (**P**rocess **ID**entification), ce PID est un nombre. Le premier processus créé au démarrage du système à pour PID « 0 », le second « 1 », le troisième « 2 » ... Le système d'exploitation utilise un compteur qui est incrémenté de 1 à chaque création de processus, le système utilise ce compteur pour attribuer les PID aux processus.

Chaque processus possède aussi un **PPID** (**P**arent **P**rocess **ID**entification). Ce PPID permet de connaitre le processus parent d'un processus (par exemple le processus "init" vu ci-dessus à un PID de 1 et un PPID de 0). À noter que le processus 0 ne possède pas de PPID (c'est le seul dans cette situation).

En vous basant sur le schéma ci-dessus, donnez le PID et le PPID du processus "getty".

5.7 Commandes pour visualiser et supprimer les processus

Ouvrez la console Windows et saisissez la commande « Tasklist »



```
C:\Users\CLOOS-SYLVAIN>Tasklist

Nom de l'image                 PID Nom de la sessio Numéro de s Utilisation
========================= ======== ================ =========== ============
System Idle Process              0 Services                   0          8 Ko
System                           4 Services                   0     20 600 Ko
Registry                        88 Services                   0     95 944 Ko
smss.exe                       364 Services                   0      1 036 Ko
```

Vous obtenez alors la liste des tâches en cours d'exécution.

Sous Linux la commande permettant de visualiser les processus est : **ps -aef**

5.8 Interblocage

Soit 2 processus P1 et P2, soit 2 ressources R1 et R2.

Initialement, les 2 ressources sont "libres". Le processus P1 qui se trouve dans l'état Elu commence son exécution, il demande la ressource R1. Il obtient satisfaction puisque R1 est libre. Son temps d'accès est terminé, et il passe donc dans l'état Bloqué

Pendant ce temps, le système a passé P2 à l'état Elu : P2 commence son exécution et demande la ressource R2. Il obtient immédiatement R2 puisque cette ressource était libre. P2 poursuit donc son exécution et demande la ressource R1, il se retrouve dans un état Bloqué puisque la ressource R1 a été attribuée à P1.

P2 étant Bloqué (en attente de R1), le système passe P1 dans l'état Elu et avant de libérer R1, il demande à utiliser R2. Problème : R2 n'a pas encore été libéré par P2, R2 n'est donc pas disponible, P1 se retrouve donc aussi Bloqué.

Résumons la situation à cet instant : P1 possède la ressource R1 et se trouve dans l'état Bloqué (attente de R2), P2 possède la ressource R2 et se trouve dans l'état Bloqué (attente de R1)

Pour que P1 puisse poursuivre son exécution, il faut que P2 libère la ressource R2, mais P2 ne peut pas poursuivre son exécution (et donc libérer R2) puisqu'il est bloqué dans l'attente de R1. Pour que P2 puisse poursuivre son exécution, il faut que P1 libère la ressource R1, mais P1 ne peut pas poursuivre son exécution (et donc libérer R1) puisqu'il est bloqué dans l'attente de R2. Bref, la situation est totalement bloquée !

Cette situation est qualifiée d'**interblocage** (**deadlock**).

Mettez au point une petite saynète de théâtre permettant d'expliquer l'interblocage de 2 processus.

D'après : Pixees, NSI-Ellipses, Irif.fr, Wikipedia, Courstechinfo.be

6. EXERCICES

6.1 Application de l'algorithme SJF : Shorted Job First

Les 3 processus suivants doivent être exécutés simultanément sur un ordinateur à un seul processeur.

Processus-1
Instruction-1
Instruction-2
Instruction-3
Instruction-4
Instruction-5
Instruction-6
Instruction-7

Processus-2
Instruction-1
Instruction-2
Instruction-3

Processus-3
Instruction-1
Instruction-2
Instruction-3
Instruction-4
Instruction-5

Schématiser l'ordre de traitement des instructions des 3 processus en utilisant l'algorithme « du travail le plus court d'abord ».

6.2 Application de l'algorithme du Tourniquet

Schématiser l'ordre de traitement des instructions des 3 processus en utilisant l'algorithme « du Tourniquet ».

6.3 Ecrire en python un programme correspondant à l'ordonnancement effectué dans l'exercice 6.2.

D'après : Pixees, NSI-Ellipses, Irif.fr, Wikipedia, Courstechinfo.be