Processus

1 Processus vs Programme

Lorsque vous écrivez un **programme Python** avec un éditeur de texte, une fois enregistré ce fichier se trouve en **mémoire morte**

Ensuite il va être **interprété** et exécuté et c'est au moment où il sera chargé en **mémoire vive** c'est là qu'il devient un **processus** parmi de nombreux autres processus en concurrence pour l'accès au processeur. Ce problème de concurrence nous l'avons déjà rencontré avec le service de gestion d'une base de données.

D'une certaine manière tout est processus, ainsi le processus qui a interprété le programme Python et même le processus de tous les processus celui qui gère tous les processus, **le système d'exploitation** mais avant d'arriver là il faut une initialisation

Au démarrage de l'ordinateur grâce au BIOS (Basic Input Output System) **le système d'exploitation est chargé en premier dans la mémoire vive pour ensuite gérer l'accès de la mémoire vive, du processeur et des entrées/sorties pour les différents processus**

2 Histoire

Pour comprendre l'intérêt des systèmes d'exploitation il faut faire un peu d'Histoire :

1. **1945-1957 : Première Informatique : l'ère des ordinateurs**

   (a) Technologie principale : Les lampes à vide.

   (b) Les ordinateurs sont peu nombreux coûtent très cher et consomment beaucoup d'énergie

(c) Ils sont principalement utilisés par les universités et l'Armée pour des calculs scientifiques

(d) Les ordinateurs ont à leur service de nombreux chercheurs, ingénieurs et techniciens pour leur fonctionnement

(e) Progressivement apparaissent des modifications pour améliorer leur **ergonomie** (adaptation d'un environnement de travail (outils, matériel, organisation...) aux besoins de l'utilisateur), ainsi en 1953 avec l'IBM 701 l'entrée des données se fait avec des bandes magnétiques pour gagner du temps

(f) Avec l'IBM 704 en 1955 (image ci-dessus) **pour optimiser le temps de calcul** d'une machine qui a coûté plusieurs millions de dollars, deux des principaux clients ayant acheté cet ordinateur, General Motors et North American Aviation ont créé **le premier système d'exploitation**, nommé GM-NAA I/O pour General Motors and North American Aviation Input Output System.

Pourquoi et comment ?

A l'époque le temps de calcul d'une machine comme l'IBM 704 était organisé suivant un **traitement par lots** :

Un **lot** est constitué d'une entrée des données et du programme (par bandes magnétiques) puis de l'exécution du programme et enfin l'impression des résultats du programme sur un télétype (une sorte d'imprimante)

Les lots devaient ensuite se succéder en cadence pour profiter au mieux de l'ordinateur

Mais la phase d'impression, purement mécanique, ralentissait la cadence aussi les ingénieurs ont eu l'idée de détourner la phase d'impression vers une phase de mémorisation sur une mémoire auxiliaire électromagnétique et **surtout que la gestion de toutes ces phases pour tous les lots allait se faire par un super-programme**, une sorte d'arbitre des programmes, placé en premier dans la mémoire de l'ordinateur

A l'époque on a appelé ce programme, un moniteur

Pour optimiser encore plus le temps, le moniteur pouvait faire chevaucher un temps de calcul pour un programme P_1 et un temps d'entrées-sorties pour un autre programme P_2 et ainsi de suite.

A partir de maintenant les programmes clients du processeur sont appelés des processus gérés par le système d'exploitation

(g) **En gestation :**

Le transistor apparaît en 1947 il remplacera progressivement les lampes à vide

2. **1957-1970 : Deuxième Informatique : l'ère des mini-ordinateurs**

Les transistors ont remplacé les tubes à vide, la taille des ordinateurs a diminué on parle maintenant de mini-ordinateurs (la taille d'un réfrigérateur pour le PDP-8 de Digital Equipment (image ci-dessus))

Cependant un transistor occupe encore une certaine place, l'époque des circuits intégrés n'est pas encore arrivée

De plus la mémoire des ordinateurs est constituée de tores de ferrites ce qui occupe aussi de la place

Pour optimiser le rendement du processeur les systèmes d'exploitation évoluent aussi :

1961 : CTSS (Compatible Time-Sharing System), est le **premier système d'exploitation de temps partagé** développé au MIT

L'échelle de temps du processeur (la micro seconde à l'époque, la nano seconde actuellement en 2020) n'est pas celle de l'Humain (la seconde) Bien que **le concept de machine de Von Neumann implique que le processeur exécute chaque instruction l'une après l'autre** on peut encore gagner du temps en entremêlant les instructions des processus, le système d'exploitation étant une sorte d'agent de la circulation faisant circuler tel processus P_1 puis tel processus P_2 etc...

L'Humain a ainsi l'illusion que plusieurs processus s'exécutent en même temps ce qui est un progrès par rapport au traitement par lots

Avec le système CTSS apparaît un langage interprété **le shell** qui permet le lancement des programmes et le "dialogue" entre l'utilisateur et le système (on verra quelques exemples plus loin)

1965 : Multics (Multiplexed Information and Computing Service), successeur de CTSS, améliore la gestion et la sécurité de la mémoire :

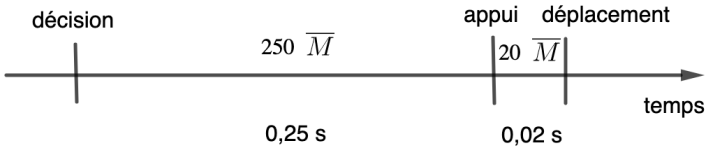
(a) Mémoire segmentée

(b) Mémoire virtuelle paginée

(c) Système de gestion de fichiers (que l'on regardera un peu en détail plus loin)

1969 : Unix, successeur de Multics et ancêtre des systèmes d'exploitation,Linux, Mac Os des ordinateurs Mac de Apple, iOs et Android

3 Temps du processeur vs Temps de L'Humain



Voici un exemple tiré du livre de Laurent Bloch *Histoire des systèmes d'exploitation*

Imaginons que l'on travaille avec un traitement de texte pour passer du début d'une ligne à la fin de la ligne on peut faire CTRL + Right.

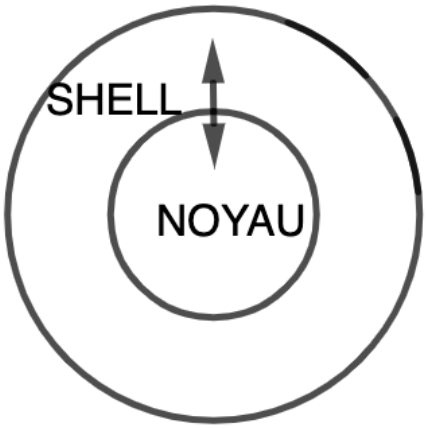
L'appui sur les touches du clavier nous prend un peu près 0,25 seconde. Le déplacement nous paraît instantané et a lieu après 0,02 s après la fin de l'appui sur les touches du clavier

En faisant l'hypothèse qu'une instruction du processeur prend une nanoseconde donc en une seconde le processeur peut exécuter environ 1 milliards d'instructions donc en 0,25 s 250 millions d'instructions et en 0,02 seconde 20 millions d'instructions

Laurent Bloch fait l'hypothèse que la capture de l'évènement CTRL + Right coûte 20 000 instructions et le déplacement du curseur 10 000. Il reste donc du temps pour 19 990 000 instructions et le processeur peut étaler les 10 000 instructions ou les entrelacer avec d'autres instructions et l'utilisateur pourra ainsi écouter des informations à la radio tout en faisant du traitement de texte et aura ainsi l'impression de la simultanéité

4 Modèle en couche d'un système d'exploitation

De la même manière qu'on utilise un modèle en couches pour faire l'abstraction de la communication des ordinateurs dans un réseau, on utilise aussi un modèle en couches pour faire l'abstraction d'un système d'exploitation

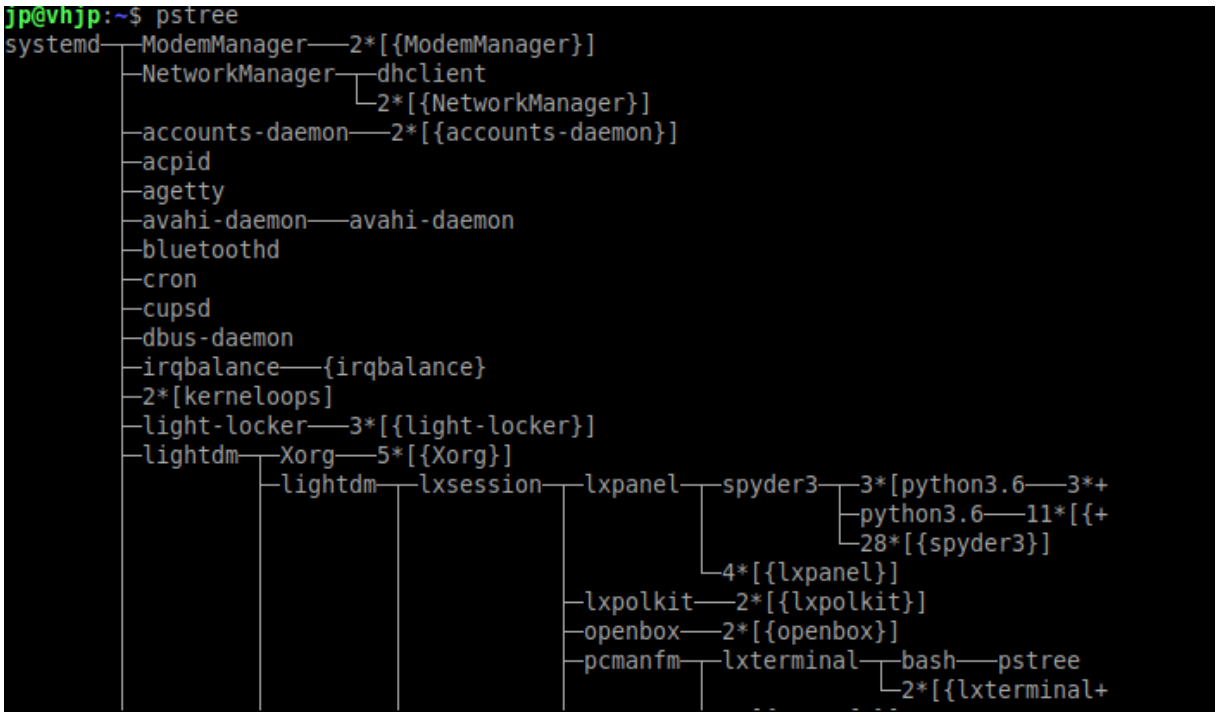


1. Au centre il y a **le noyau ou kernel** composé des éléments fondamentaux du système d'exploitation gérant le partage de la mémoire, du processeur et des entrées sorties pour les autre processus
2. Autour **le shell (coquille) un langage (interprété) de commandes** permettant de faire le lien entre le système d'exploitation et l'utilisateur via un terminal

Nous allons utiliser le shell de Ubuntu (Linux) (bash pour Bourne again shell) pour découvrir les processus par la pratique

5 Gestion de la concurrence par le système d'exploitation

Dans un terminal on entre la commande `pstree` pour visualiser sous forme d'arbre la liste des principaux processus en cours



L'arbre permet de visualiser que **chaque processus est créé par un processus père**

Ainsi le `bash` est un processus **créé** par son père le processus `lxterminal`

`pstree` est un processus fils du `bash`

Si on exécute cette fois-ci la commande `top`, on peut voir les informations concernant les principaux processus rafraîchies toutes les secondes :

On a lancé l'exécution d'un tri par sélection pour une durée de 20 secondes, on a ouvert deux onglets dans Firefox et on a ouvert un terminal dans lequel on a entré la commande `top`

Chaque processus a un `PID` pour process ID qui lui est attribué par le système d'exploitation

Le `PID` 1 est attribué au premier processus, au démarrage de l'ordinateur, le processus `init`

(pour voir le processus `init` exécutez la commande `ps -aux`)

PID	UTIL.	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TEMPS+	COM.
1694	**jp**	**20**	**0**	**264248**	**70476**	**23912**	**R**	**100,3**	**7,5**	**0:45.65**	**python3.6**
1816	jp	20	0	278900	62108	53504	S	26,7	6,6	0:00.81	lximage-qt
812	jp	20	0	127928	18028	13124	S	15,8	1,9	0:31.30	lxpanel
573	**root**	**20**	**0**	**184428**	**26244**	**15292**	**R**	**5,0**	**2,8**	**0:25.32**	**Xorg**
1265	jp	20	0	774356	172604	59252	S	5,0	18,3	4:25.58	Web Content
1040	jp	20	0	71156	2784	2460	S	3,3	0,3	0:06.51	indicator-+
744	jp	20	0	7160	2084	1580	S	1,7	0,2	0:03.54	dbus-daemon
834	jp	20	0	129104	13372	10684	S	1,7	1,4	0:09.47	nm-applet
1087	jp	20	0	117828	11768	9124	S	1,7	1,2	0:06.66	lxterminal
472	message+	20	0	7384	2708	1980	S	1,3	0,3	0:02.19	dbus-daemon
520	root	20	0	73340	7228	5748	S	1,3	0,8	0:03.56	NetworkMan+
1359	jp	20	0	407272	34644	26912	S	1,0	3,7	0:05.49	WebExtensi+
1639	jp	20	0	717712	130280	48668	S	1,0	13,8	0:26.49	spyder3
1779	**jp**	**20**	**0**	**9468**	**3404**	**2868**	**R**	**1,0**	**0,4**	**0:01.68**	**top**
1178	jp	20	0	1048424	125208	48316	S	0,7	13,3	4:37.88	firefox

1. Linux partage le temps d'accès au processeur (il peut y avoir plusieurs processeurs ou coeurs mais un processeur ne peut s'occuper que d'un seul processus) entre les différents processus mais reprend régulièrement la main grâce à un mécanisme d'**interruptions**. On dit que Linux est un système **préemptif**
2. Un processus **actif ou élu (Running)** qui est interrompu devient un processus **prêt (Ready)** et va dans une **file de priorité** attendre son tour pour être en exécution à nouveau.

   Un processus peut être interrompu de son propre fait, par exemple le processus suivant "attend" une entrée du clavier.

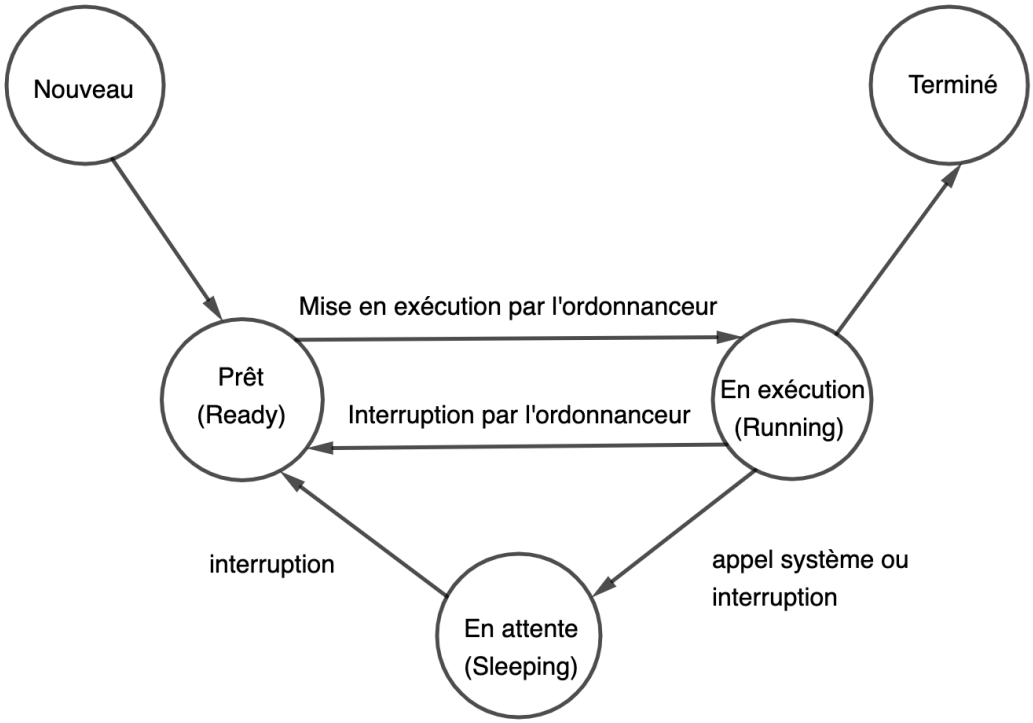
   ```
   nom = input("Entrez votre nom")
   print("Bonjour "+ nom)
   ```

   Concrètement, le processus fait un **appel système** au système d'exploitation et se met en **attente (Sleeping)**

   Le système d'exploitation transmet l'information au contrôleur d'entrée-sortie

   Une fois l'entrée clavier réalisée, le gestionnaire transmet l'information au système d'exploitation qui "réveille" le processus en attente et le met dans la file de priorité

   Mais un processus "long" peut être interrompu par le système d'exploitation, par exemple un tri par sélection qui dure une vingtaine de secondes, va être régulièrement interrompu, afin que les autres processus puissent utiliser le processeur



3. Pour que le processus puisse reprendre les calculs où il était, au moment de l'interruption l'état des registres et du compteur (**le contexte du processus**) sont sauvegardés pour pouvoir être rechargés au moment où le processus reprend la main. On dit qu'il y a **commutation du contexte** (context switch)
4. Il arrive qu'un processus ne se comporte pas comme prévu (par exemple on a une boucle infinie dans un programme) et on aimerait pouvoir l'arrêter

   On envoie alors un **signal** au processus pour cela

   La commande `kill - [nom du signal | numéro du signal] PID` permet d'arrêter un processus

   Voici les principaux signaux

   (a) 15 : SIGTERM -> arrêter un processus avec fermeture normale de ses fichiers. C'est le signal par défaut de la commande `kill`

   (b) 9 : SIGKILL -> arrêter sans sauvegarde

   (c) 19 : SIGSTOP -> stopper ou arrêter provisoirement un processus.

   (d) 18 : SIGCONT -> remettre en exécution un processus stoppé

   Par exemple si on veut arrêter un processus Firefox de PID égal à 1178 on peut :

   (a) `kill -9 1178`

   (b) `kill -15 1178` ou `kill 1178`
5. Comme pour les accès concurrents à une même base de données par plusieurs utilisateurs, il peut y avoir, un **interblocage** :

   Un processus P1 a accès à une ressource R1 (entrée/sortie, mémoire etc...) et un autre processus P2 a accès à une ressource R2, puis P1 a besoin de R2 donc P1 attend P2 dans l'ordonnancement, puis P2 a besoin de R1 dans ce cas P2 attend P1 d'où l'interblocage

6 Exercices

Ex 1

Lire la vidéo suivante sur Youtube (en anglais) `https://www.youtube.com/watch?v=DKaVvv15Heo` concernant la machine IBM 704 et répondre aux questions suivantes :

1. Quels sont les langages de programmation utilisés avec l 'IBM 704 ?
2. Chercher sur wikipedia le volume occupé par l'IBM 704

Ex 2

Lire la vidéo suivante sur Youtube (en anglais) `https://www.youtube.com/watch?v=XvDZLjaCJuw` à propos de Unix

On peut y voir Ken Thompson qui avait travaillé sur le système Multics au sein des laboratoires Bell, et Dennis Ritchie qui a inventé le langage de programmation C

Ils ont écrit Unix en C et ont privilégié **la portabilité de leur système** (c'est à dire le fait que ce système d'exploitation puisse fonctionner sur des machines de type différents en ne modifiant que le compilateur écrit aussi en C

1. Lire de 2 :37 jusqu'à 5 :15 et répondre aux questions
   (a) Quels sont les deux types de programme cités ?
   (b) Quelles sont les trois parties d' Unix ?
2. Lire de 11 :12 jusqu'à 12 :57 une description du système de gestion des fichiers UNIX

Ex 3

Vous venez de cliquer sur l'icône de votre navigateur Web. Puis dans le terminal vous exécutez la commande top, quel est l'état du processus associé, R ou S ?

Ex 4

1. Vous venez de cliquer sur l'icône de IDLE. Puis dans le terminal vous exécutez la commande top, quel est l'état du processus R ou S ?
2. Vous exécutez le programme Python suivant, quel est l'état du processus R ou S, avant d'entrer votre nom au clavier et après avoir entré votre nom ?

```
nom = input("Entrez votre nom")
print("Bonjour "+ nom)
```

3. Vous exécutez un tri par sélection sur un tableau de 10^4 nombres. Le temps de traitement est a peu près 20 s, puis dans le terminal vous exécutez la commande top, quel est l'état du processus R ou S ?

Ex 5

Exécuter la commande `ps -aux` puis répondre aux questions suivantes

1. Quel est l'état du processus `init` de PID égal à 1 ?Quel est le pourcentage CPU ?
2. Chercher le processus `usr/sbin/cron`, quel est l'utilisateur ?A quoi sert le processus `cron` ?
3. Qui sont les utilisateurs `kernoops` et `whoopsie` ?
4. Que sont les états I ? Sl ? Rl ? Ss ? Ssl ?
   (indication : dans le terminal faire `man ps`)

Ex 6

Ouvrir Firefox, chercher le PID du processus associé à Firefox, puis arrêter Firefox en lui envoyant un signal

Ex 7

Lancer un tri par sélection pour une durée d'une vingtaine de secondes

Le mettre sur pause en lui envoyant un signal puis le relancer

Ex 8

Ex 2 - Bac 2021 - Métropole Candidat libre 2