

# Gestion des processus et des ressources

- L'ordonnanceur -

J. Boucher

Lycée Pierre-Paul RIQUET, Terminale NSI

5 avril 2025

# Plan

## 1 Problématique

## 2 L'ordonnaceur

# Plan

## 1 Problématique

## 2 L'ordonnaceur

# Problématique

**Comment le processeur peut-il exécuter « en même temps » les instructions de plusieurs programme ?**



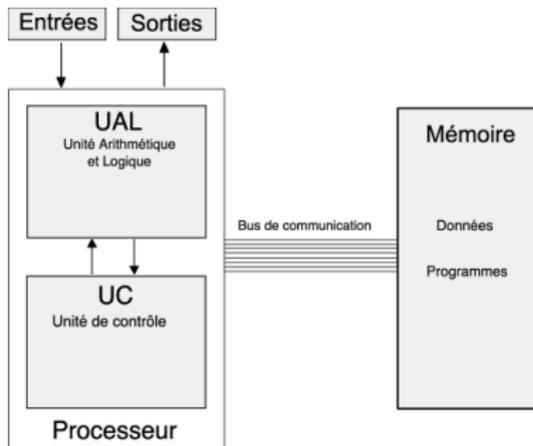
# Plan

1 Problématique

2 L'ordonnaceur

# 0. Rappel sur l'exécution d'un programme

- Un **exécutable** est un fichier contenant une suite d'instruction en langage machine.



- Lorsqu'on exécute un programme, le **système d'exploitation** effectue les actions suivantes :

- 1 l'exécutable est copié dans la mémoire RAM, à l'adresse  $a$  ;
- 2 le système d'exploitation écrit la valeur  $a$  dans le registre  $IR$ .

Au prochain cycle d'horloge du **processeur**, l'instruction dont l'adresse est stockée dans le registre  $IR$  sera exécutée. (cf 1<sup>ère</sup>, Architecture von Neumann).

# 1. Interruptions

- Une **interruption** est un **signal** envoyé au processeur lorsqu'un événement se produit.

*Exemple : le disque a fini une tâche d'écriture, la carte réseau reçoit des paquets de données...*

- Le **gestionnaire d'interruption** est appelé après l'instruction courante pour y répondre. C'est un programme spécial, dit « système », chargé très tôt au démarrage de la machine.

- Les **interruptions d'horloge** sont générées à intervalle de temps fixe par le processeur.

*Par exemple, les horloges de haute précision sont capables d'émettre des interruptions à 10 MHz, i.e. toutes les 100 ns.*

→ **Interruptions d'horloge et gestionnaire d'interruption sont à la base de l'exécution concurrente des programmes.**

## 2. Vocabulaire

- **Exécutable** : fichier binaire contenant des instructions machines directement exécutables par le processeur de la machine.
- **Processus** : « *programme en cours d'exécution* ». Un processus est le phénomène dynamique qui correspond à l'exécution d'un programme donné. Il est décrit notamment par :
  - la mémoire allouée par le système pour l'exécution du programme ;
  - les ressources utilisés par le programme (fichiers, connexion réseau, etc) ;
  - les valeurs stockées dans tous les registres du processeur.
- **Tâche** (ou *thread* en anglais) : exécution d'une suite d'instructions démarré par un processus.
- **Exécution concurrente** : deux processus ou tâches s'exécutent de manière *concurrente* si les intervalles de temps entre le début et la fin de leur exécution ont une partie commune.
- **Exécution parallèle** : deux processus ou tâches s'exécutent en *parallèle* si ils s'exécutent au même instant.

## 2. Vocabulaire

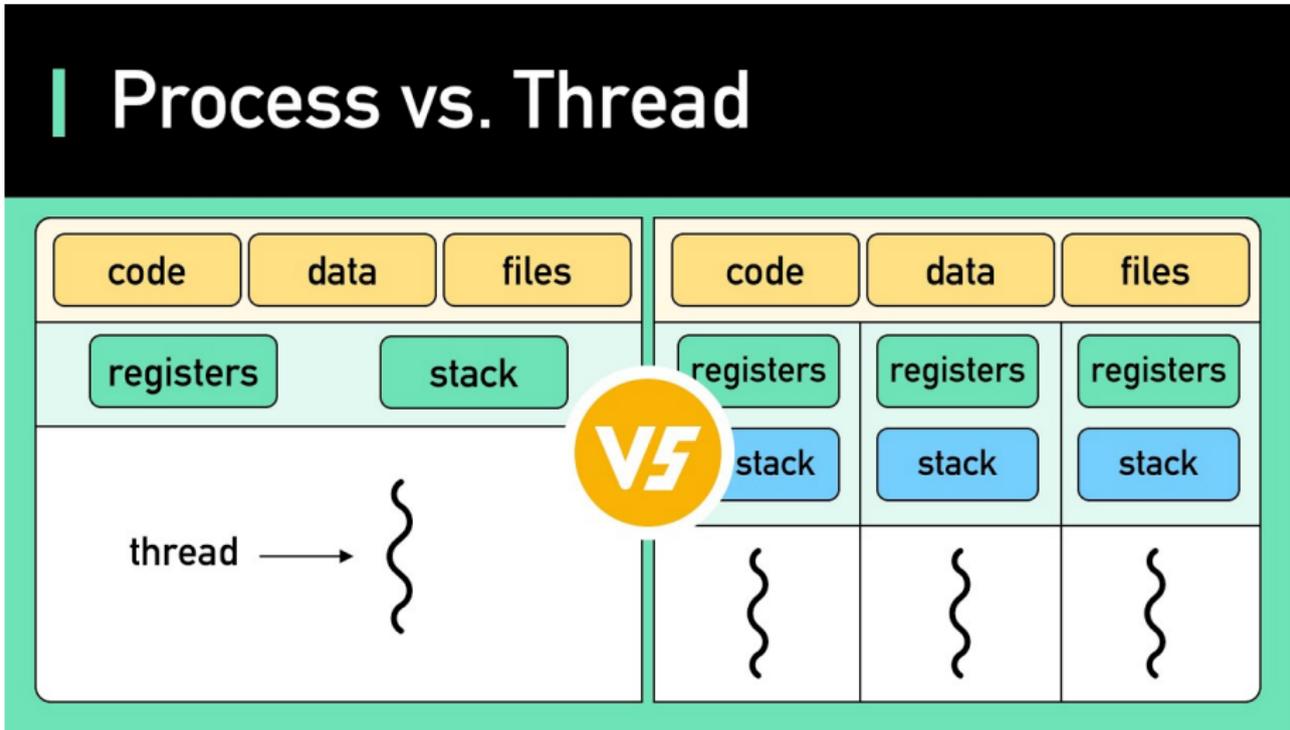
### Processus VS Tâche

Il ne faut pas confondre *Processus* et *Tâche*!

- Deux **processus** sont l'exécution de deux programmes.  
*Par exemple, un traitement de texte et un navigateur Web.*
- Deux **tâches** sont l'exécution *concurrente* de deux suites d'instructions d'un même processus.  
*Par exemple, un navigateur Web peut lancer deux tâches, l'une pour effectuer le rendu de la page Web, l'autre pour télécharger un fichier.*
- **Différence fondamentale** : deux processus ne partagent pas le même espace mémoire alors que deux tâches peuvent accéder aux variables globales du programme et occupent le même espace mémoire.

## 2. Vocabulaire

# Process vs. Thread



## Scénario : Utilisation concurrente d'un navigateur et d'un traitement de texte

- 1 Le traitement de texte est en cours d'exécution.
- 2 Une interruption d'horloge se déclenche.
- 3 Le code du gestionnaire d'interruption est appelé. Il reçoit en argument les valeurs qu'ont tous les registres avant le déclenchement de l'interruption (état interne du traitement de texte).
- 4 Le gestionnaire d'interruption sauvegarde ces registres à un endroit donné de la mémoire.
- 5 Il choisit dans la liste des processus un autre processus, par exemple celui correspondant au navigateur Web.
- 6 Il restaure les valeurs de tous les registres du processeur qu'il a suavegardé lors de la précédente interruption, notamment le contenu du registre *IR* (adresse de la prochaine instruction à exécuter).
- 7 Le gestionnaire d'interruption rend la main. La prochaine instruction à exécuter est celle du processus navigateur Web, qui reprend son exécution jusqu'à la prochaine interruption d'horloge.

## 3. L'ordonnanceur

### Ordonnanceur du système d'exploitation

- L'**ordonnanceur** permet de déterminer quel processus s'exécute à un instant donné et quel temps de processeur lui allouer.
- L'interruption d'un processus et la sauvegarde de son état par l'ordonnanceur s'appelle une **commutation de contexte**.
- Le système d'exploitation conserve pour chaque processus une structure de données nommée **PCB** (pour *Process Control Bloc*) ou **bloc de contrôle du processus** qui stocke dans une zone mémoire :

Nom	Description
PID	<i>Process ID</i> ou identifiant du processus
État	état dans lequel se trouve le processus
Registres	la valeur des registres lors de sa dernière interruption
Mémoire	zone mémoire (plage d'adresses) allouée par le processus
Ressources	liste des fichiers ouverts, connexions réseaux en cours, etc.

# À toi de jouer !

On considère le programme suivant :

```
1 texte = input("Saisir une phrase : ")
2 print("Votre phrase en majuscules : ", texte.upper() )
```



**Travail à faire** Que va-t-il se passer après l'exécution de la première ligne de code par le système ?

# À toi de jouer !

On considère le programme suivant :

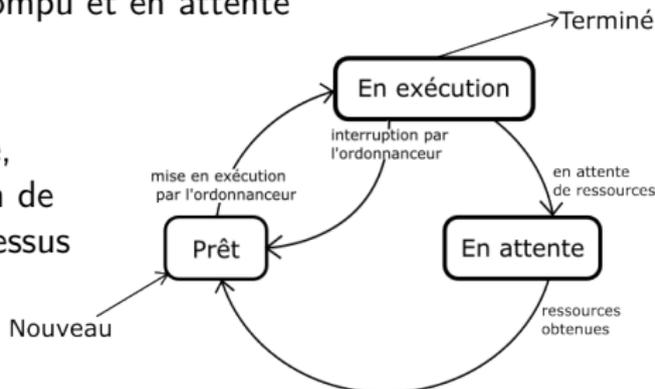
```
1 texte = input("Saisir une phrase : ")
2 print("Votre phrase en majuscules : ", texte.upper() )
```

- 1 Tant que l'utilisateur ne saisit rien, le programme ne peut pas avancer à l'instruction suivante.
- 2 Le système d'exploitation peut suspendre le processus et le mettre en attente.
- 3 Une interruption déclenchée par l'interaction de l'utilisateur avec le clavier signale qu'un événement est survenu.
- 4 Le système peut alors « réveiller » le processus du programme qui attendait cet événement.

→ **Les processus peuvent être dans différents états.**

## 4. États des processus

- **Nouveau** : état d'un processus en cours de création. Le système d'exploitation vient de copier l'exécutable en mémoire et d'initialiser le PCB.
- **Prêt** : le processus peut être le prochain à s'exécuter. Il est dans la file des processus qui « attendent » leur tour et peuvent être choisi par l'ordonnanceur.
- **En exécution** : le processus est en train d'être exécuter.
- **En attente** : le processus est interrompu et en attente d'un événement externe (E/S, allocation mémoire, etc.).
- **Terminé** : le processus s'est terminé, le système d'exploitation est en train de désallouer les ressources que le processus utilisait.



# Mini-TP : « Observer les processus »

Il est très facile de voir les différents processus s'exécutant sur une machine.

- Sous *GNU/Linux* ou *macOS*, on peut utiliser la commande `ps` pour afficher les informations sur les processus.
- Sous *Windows*, on peut utiliser le gestionnaire de tâches CTRL+MAJ+ECHAP.

```

pichentte@MAGI:~$ ps -aef
  UID          PID     PPID  C   STIME TTY          TIME CMD
  root           1         0   0  16:35 ?        00:00:01 /sbin/init splash
  root           2         0   0  16:35 ?        00:00:00 [kthreadd]
  root           3         2   0  16:35 ?        00:00:00 [pool_workqueue_release]
  root           4         2   0  16:35 ?        00:00:00 [kworker/R-rcu_gp]
  root           5         2   0  16:35 ?        00:00:00 [kworker/R-sync_wq]
  root           6         2   0  16:35 ?        00:00:00 [kworker/R-slub_flushwq]
  root           7         2   0  16:35 ?        00:00:00 [kworker/R-netns]
  root          4922         2   0  16:49 ?        00:00:00 [kworker/0:0-events]
  pichent+    4954       3566   0  16:49 ?        00:00:00 /snap/firefox/5987/usr/lib/firefox/firefox -con
  root          4980         2   0  16:50 ?        00:00:00 [kworker/8:3-events]
  root          5027         2   0  16:51 ?        00:00:00 [kworker/16:0-events]
  root          5041         2   0  16:51 ?        00:00:00 [kworker/19:0-mm_percpu_wq]
  root          5053         2   0  16:51 ?        00:00:00 [kworker/10:2-events]
  pichent+    5095       3566   0  16:53 ?        00:00:00 /snap/firefox/5987/usr/lib/firefox/firefox -con
  pichent+    5166       4834   0  16:55 pts/1    00:00:00 ps -aef
  
```

En plus du **PID**, chaque processus possède un **PPID** (pour *Parent Process Identifier*), il s'agit du PID du processus parent, c'est-à-dire celui qui a déclenché la création du processus.

→ **Un processus peut créer lui même un ou plusieurs autres processus, appelés processus fils.**

# Mini-TP : « Observer les processus »

## Commande ps

Lance un terminal et exécutez la commande `ps -aef`.

- 1 À quoi correspond le dernier processus lancé ? Quel est son processus parent et pourquoi ?
- 2 Lance *Nautilus* avec la commande `nautilus &`
- 3 Affiche la liste des processus puis cherche celui ou ceux correspondant à l'exécution du programme Nautilus.
- 4 Cherche le processus parent du processus correspondant à l'exécution du programme Nautilus.
- 5 Ferme Nautilus puis exécute à nouveau la commande et vérifiez qu'il n'y a plus de processus correspondant à l'exécution de Nautilus.

# Mini-TP : « Observer les processus »

- 6 Ouvre le navigateur Firefox avec la commande `firefox &` puis exécutez à nouveau la commande `ps -aef`.
- 7 Cherche dans la liste des processus le premier correspondant à l'exécution du navigateur
- 8 Cherche ensuite les processus fils de ce processus.
- 9 Ouvre un nouvel onglet dans le navigateur et rends-toi sur une page Web de ton choix. Exécute à nouveau la commande, tu devrais constater qu'au moins un nouveau processus lié à l'exécution du navigateur a été créé.

# Mini-TP : « Observer les processus »

## Commande pstree

Sous *GNU/Linux* il est possible de voir l'arborescence des processus avec la commande `pstree`.

Teste la commande `pstree` et cherche dans l'arborescence les processus correspondant à l'exécution du terminal, de firefox, de nautilus...

**Remarque :** Tu peux aussi ajouter l'option `-p` pour afficher tous les processus avec leurs PID : `pstree -p`.