Term NSI **Gestion des processus et des ressources par un système d'exploitation**

1) Distinguer **programme** et **processus**

```
# programme test.py
n = int(input('Entrer n : '))
for i in range(n):
    print(i+1)
```

Un même **programme** peut être exécuté plusieurs fois sur une même machine : ces différentes exécutions peuvent être « simultanées » ou « concurrentes ».
Chaque ***exécution d'un programme*** correspond à un **processus**.
Chaque processus possède :

- une liste **d'instructions** (un programme traduit en assembleur)
- un **contexte** d'exécution dans lequel sont stockées les valeurs des données utilisées par le processus.

Un processus peut utiliser certaines ressources :

- le processeur : pour effectuer les calculs !
- d'autres ressources matérielles : clavier, carte réseau, imprimante, disque dur…

Question : comment se fait-il que plusieurs programmes semblent s'exécuter « en même temps » ?

2) gestion des processus

Un processus peut déclencher la création d'un second processus.
Dans ce cas, le premier processus est appelé « parent » ou « père » et le second « enfant » ou « fils ».
C'est le système d'exploitation qui est chargé de gérer l'exécution de tous les processus, depuis leur création jusqu'à leur destruction.
On peut distinguer les processus crées par les utilisateurs, des processus (et sous processus) crées par le système d'exploitation (démarrage de la machine, services Windows, antivirus…).
C'est **l'ordonnanceur du système d'exploitation** qui déterminer l'état de chacun des processus en cours d'exécution.

a) les différents états d'un processus

Chaque processus est créé, puis détruit. Entre les deux, il peut se trouver dans différents états :

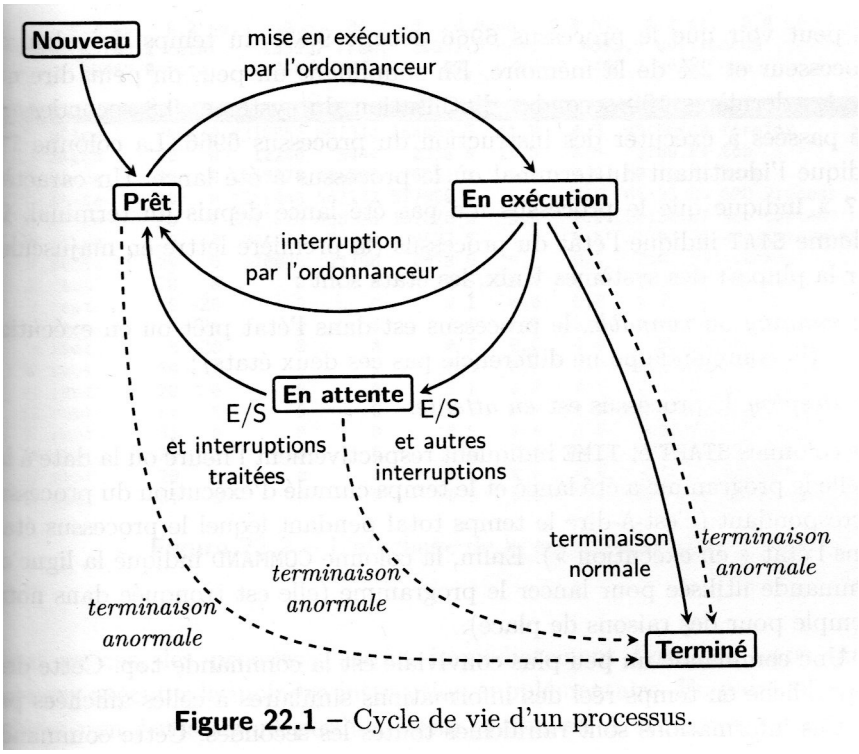

Figure 22.1 – Cycle de vie d'un processus.

- **Nouveau :** état d'un processus en cours de création. Le système d'exploitation vient de copier les instructions en mémoire et d'initialiser le contexte d'exécution.
- **En exécution** (ou élu) : il est en cours d'exécution.
- **En attente :** le processus est interrompu et en attente d'un événement externe (entrée/sortie, allocation de mémoire).
- **Prêt** : les données sont disponibles. Le processus n'attend plus que la disponibilité du processeur.
- **Terminé :** le processus vient de terminer son exécution (il est alors détruit)

b) ordonnancement des processus

C'est le système d'exploitation qui est chargé de gérer l'ordonnancement des processus. Lorsqu'une unité de calcul est libre, c'est le système d'exploitation qui va déterminer un nouveau processus à affecter à l'unité de calcul. Pour cela il existe plusieurs algorithmes d'ordonnancement :

- Le modèle **FIFO**: on affecte les processus dans l'ordre de leur apparition dans la file d'attente.
- Le modèle **SJF**: Shortest Job First, on affecte en premier le « plus court processus en premier » de la file d'attente à l'unité de calcul.

Actuellement,la plupart des systèmes d'exploitation utilise un modèle **Priorité**, reposant sur les principes suivants :

- chaque processus possède une priorité de base.
- cette priorité augmente quand le processus est inactif et diminue quand il est actif (le taux de changement dépend de la priorité de base).
- le système choisit parmi les processus de plus forte priorité.

3) Risque d'interblocage

a) Utilisation des ressources

Un processus va trouver à sa disposition un grand nombre de ressources, comme la RAM, les disques durs, les supports amovibles (clés USB,..), les fichiers, ...

Lorsqu'un processus est dans l'état actif (ou élu), dans des conditions normales de fonctionnement, il ne peut utiliser une ressource qu'en suivant la séquence de tris étapes suivante:

1. **Requête** : le processus fait une demande pour utiliser la ressource. Si cette demande ne peut pas être satisfaite immédiatement, parce que la ressource n'est pas disponible, le processus demandeur se met en état attente jusqu'à ce que la ressource devienne libre.
2. **Utilisation** : Le processus peut exploiter le ressource.
3. **Libération** : Le processus libère la ressource qui devient disponible pour les autres processus éventuellement en attente.

Lorsqu'un processus a fait une requête et que la ressource n'est pas disponible pour le moment, il va passer à l'état « En attente », le temps que la ressource se libère. Quand la ressource sera disponible, il pourra passer à l'état « Prêt ».

b) Exemple pratique

Trois élèves doivent tracer une droite parallèle à une droite donnée : la méthode qu'ils doivent appliquer utiliser une règle et une équerre.

- Alice possède une règle, mais pas d'équerre
- Bob possède une équerre, mais pas de règle
- Charlie ne possède ni règle ni équerre…

Si aucun des élèves qui possède une ressource ne renonce (temporairement) à l'utiliser… aucun des élèves ne parviendra à exécuter la construction.

Cette analogie permet de comprendre que des processus peuvent s'**inter-bloquer**.

c) Quatre conditions caractérisent l'interblocage:

1. **Exclusion mutuelle** : Les ressources ne sont pas partageables, un seul processus à la fois peut utiliser la ressource.
2. **Possession et attente** : un processus détient une ressource, et demande une autre ressource détenue par un autre processus.
3. **Non préemption** : Les ressources ne sont pas préemptibles c'est-à-dire que les libérations sont faites volontairement par les processus. On ne peut pas forcer un processus à rendre une ressource.
4. **Attente circulaire** : Il doit exister un ensemble de ***n*** processus tel que P1 attend une ressource possédée par P2, P2 attend une ressource possédée par P3, … et P***n*** attend une ressource possédée par P1 !