

Systèmes d'exploitation et processus

Fiche de synthèse

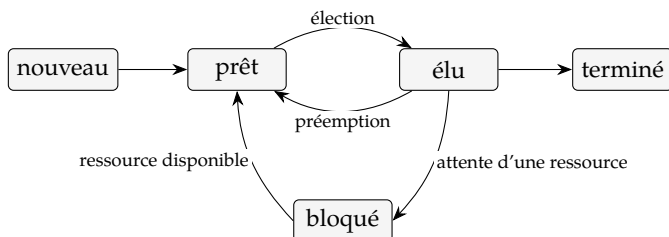
① PROGRAMME ET PROCESSUS

Le **système d'exploitation** fait le lien entre le matériel et les programmes : il gère les processus, les fichiers, la mémoire, les périphériques et la sécurité.

Définition 1 (Processus) Un **programme** est un fichier d'instructions, statique. Un **processus** est un programme en cours d'exécution, avec ses données, ses ressources et son état d'avancement. Lancer trois fois le même programme crée trois processus.

- Chaque processus a un **PID**, numéro unique, un **PPID**, le PID du processus père qui l'a créé, et un **UID**, l'identifiant de l'utilisateur qui l'a lancé.
- Son espace mémoire contient son code, ses variables globales, sa pile et son **contexte d'exécution**, copie des registres du processeur.
- Pour passer d'un processus à un autre, le système sauvegarde puis restaure les contextes en quelques microsecondes : les programmes semblent s'exécuter en même temps, c'est le **pseudo-parallélisme**.

② LES ÉTATS D'UN PROCESSUS



- Un processus **prêt** attend le processeur ; un processus **élu** s'exécute.
- Un processus élu qui attend une ressource, comme une saisie au clavier, passe à l'état **bloqué** ; quand la ressource est disponible, il redevient prêt, car un autre processus occupe le processeur.
- Un processus ne se termine que depuis l'état élu.

③ COMMANDES LINUX

Commande	Rôle
<code>ps -ef</code>	liste les processus, avec PID et PPID
<code>ps tree</code>	affiche l'arbre des processus
<code>top</code>	suit les processus en temps réel
<code>kill PID</code>	demande l'arrêt d'un processus
<code>kill -9 PID</code>	force l'arrêt

④ ORDONNANCEMENT

L'**ordonnanceur** choisit quel processus prêt obtient le processeur.

- **FIFO** : premier arrivé, premier servi ; un processus long fait attendre tous les autres.
- **SJF** : le plus court d'abord ; il minimise le temps d'attente moyen mais exige de connaître les durées à l'avance.
- **Priorités** : les processus de faible priorité risquent de ne jamais être élus, c'est la **famine**.
- **Tourniquet** : chaque processus reçoit à son tour un **quantum** de temps ; s'il n'a pas fini, il repart en fin de file.

Définition 2 (Temps de séjour et temps d'attente) Le **temps de séjour** d'un processus est son temps de terminaison moins son temps d'arrivée ; son **temps d'attente** est son temps de séjour moins sa durée.

Cinq processus de durées 3, 6, 4, 2 et 1 arrivent aux instants 0, 1, 4, 6 et 7. Avec le tourniquet et un quantum de 2 :

P1	P2	P1	P3	P2	P4	P5	P3	P2	
0	2	4	5	7	9	11	12	14	16

Les temps d'attente valent 2, 9, 6, 3 et 4 : la moyenne est 4,8 quanta, contre 3,2 avec SJF. Le tourniquet est plus équitable et évite la famine. Linux combine quantum et priorités.

⑤ ACCÈS CONCURRENT

Deux processus testent qu'il reste un pion, puis le prennent. Si le premier est interrompu entre le test et la prise, le second prend le même pion : c'est un **accès concurrent** mal protégé. On protège la **section critique** par un **verrou** : un seul processus à la fois peut l'exécuter.

```

import threading

compteur = 0
verrou = threading.Lock()

def incrementer():
    global compteur
    for _ in range(10_000):
        with verrou:
            compteur = compteur + 1

threads = [threading.Thread(target=incrementer)
            for _ in range(10)]
for t in threads:
    t.start()
  
```

```
for t in threads:
    t.join()
print(compteur)    # 100000
```

Un **thread** est un fil d'exécution à l'intérieur d'un processus ; les threads d'un même processus partagent sa mémoire, d'où le besoin de verrous. `start` lance un thread, `join` attend sa fin.

⑥ INTERBLOCAGE

Définition 3 (Interblocage) Il y a **interblocage** quand des processus sont bloqués, chacun attendant une ressource détenue par un autre : P1 détient R1 et attend R2, P2 détient R2 et attend R1.

Selon Coffman (1971), un interblocage exige quatre conditions à la fois :

1. **exclusion mutuelle** : une ressource est libre ou attribuée à un seul processus ;
2. **détention et attente** : un processus qui détient une ressource en demande une autre ;
3. **non-préemption** : seul le processus qui détient une ressource peut la libérer ;
4. **attente circulaire** : un cycle de processus s'attendent les uns les autres.

Supprimer une seule condition rend l'interblocage impossible : par exemple, numéroté les ressources et imposer de les demander dans l'ordre croissant empêche toute attente circulaire.

📌 À retenir.

- Un processus est un programme en cours d'exécution, identifié par son PID.
- États : nouveau, prêt, élu, bloqué, terminé.
- Temps de séjour : terminaison moins arrivée ; temps d'attente : séjour moins durée.
- Un verrou protège une section critique contre les accès concurrents.
- L'interblocage exige les quatre conditions de Coffman.