Processus :

- Unité système qui permet l'exécution d'un programme.
- Programme en exécution → Dynamique
- Composants :
    - Pile d'exécution **(Fonctions)**
    - Segments de code **(Instruction en langage machine)**
    - Segments de données **(Variables)**

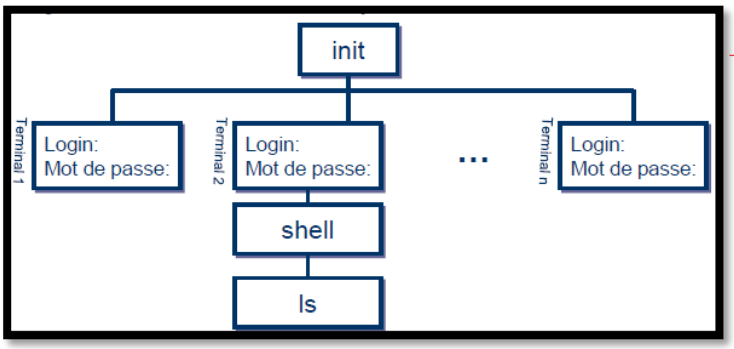
Thread :

- Unité d'exécution de code issu d'un processus mais qui contient que la pile d'exécution.

Modèle des processus :

1. **Arborescence**

   Un processus est identifié par un **PID** (**P**rocess **Id**entifier) et un **PPID** (**P**arent **P**rocess **Id**entifier).



→ Example Sous Linux

(**ps -ef** : Nous donne la liste des processus sous Linux)

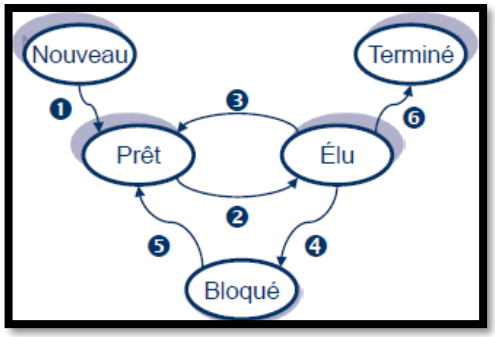
2. **États :**

   Lorsqu'un processus s'exécute, il change d'état :

   Prêt (Ready) | **Élu** (Running) | **Bloqué** (Sleep)

   - **Prêt** → En attente de son tour
   - **Élu** → En cours d'exécution
   - **Bloqué** → En attente d'événement

3. Transitions :



1. Création du processus
2. Allocation du processus
3. Fin du temps alloué sur le processus, l'exécution du processus n'est pas terminée
4. Opération E/S
5. Fin opération E/S
6. Exécution terminée

4. PCB :

Chaque processus est représenté par une structure de données contenant toute information décrivant le contexte du processus → Bloc de contrôle.

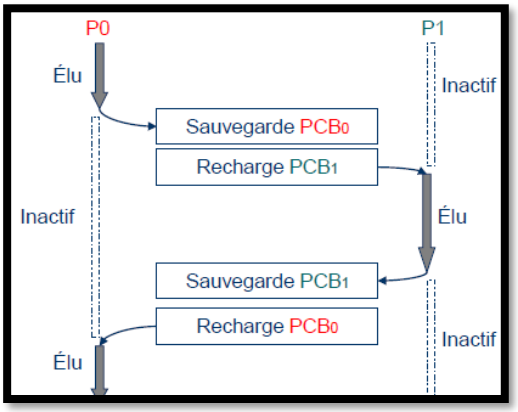
PCB

Attributs d'un **PCB** :

- PID et PPID
- État
- Priorité
- Compteur original
- Fichiers ouverts
- Pointeurs : segments code, segments pile, segments donnés
- Temps d'exécution

5. Commutation de contexte :

Le SE donne le contrôle du processeur d'un processus à un autre en effectuant des commutations de contexte, qui consiste à mémoriser le **PCB** du processus courant et charger le **PCB** du processus à élire.
Ce basculement est géré par le noyau.



6. Opérations sur les processus :

Le SE offre 4 opérations :

- **Création** du processus
- **Destruction** du processus
- **Suspension** d'exécution de processus
- **Reprise** de processus

Les primitives de gestion de processus :

- **Création** d'un processus

  int fork()
- Identification des **processus**

  int pid= getpid() ; int ppid= getppid()
- Identification de **propriétaires**

  int pid= getuid() ; int ppid= getgid()
- **Mise en sommeil** d'un processus

  int sleep(int n)
- **Terminaison** d'un processus

  void exit(int statut)

```
#include<stdio.h>
#include<stdlib.h>
int main ()
{
printf("Bonjour Je suis le processus père: %d \n",getpid());
fork();
printf(" %d : Je suis le processus fils de %d \n",
getpid(),getppid());
}
```

7. Réalisation des processus :

Le SE gére une table de processus

Tableau processus					
Processus	Etat	Compteur ordinal	Allocation mémoire	État des fichiers	Autres...
P1	Prêt	F8 B22 C	F800	Clients : ouvert	...
P2	Bloqué	A5 F4 6	E458	-	...
...	...				

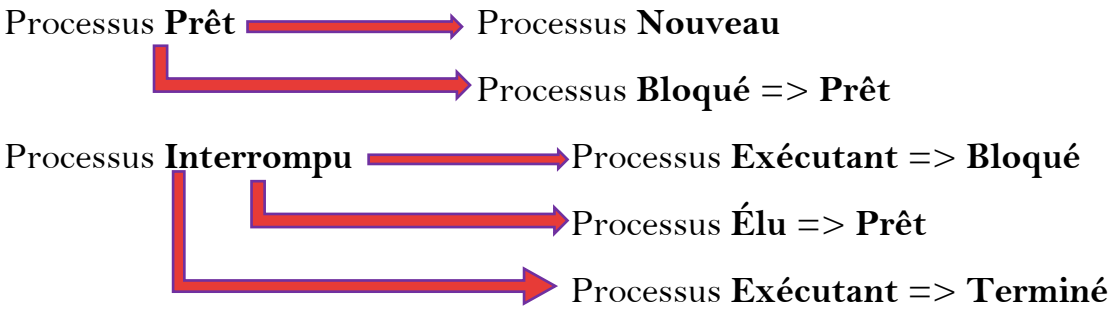
Ordonnancement des processus

L'ordonnanceur (Scheduler) du SE choisit quel processus élire en utilisant un algorithme d'ordonnance qui se doit de :

- Donner à chaque processus sa part de temps CPU **(équité)**
- Utiliser le temps processeur à 100% **(Efficacité)**
- Minimiser le temps de réponse en mode interactif

Il existe par la suite 2 types d'algorithme :

- Ordonnancement **sans** réquisition : Exécution d'un processus jusqu'à sa terminaison
- Ordonnancement **avec** réquisition : Suspension du processus même s'il peut continuer



Critères d'ordonnancement utilisés fréquemment :

Utilisation de l'UCT :
Pourcentage de temps durant lequel l'UC exécute un processus.

Débit :
Nbr. de processus pouvant être exécutés par le système sur une période de temps X.

Priorité :
Attribue un traitement préférentiel aux processus dont le nv. de priorité est supérieur

Temps de rotation :
Durée moyenne pour qu'un processus s'exécute.

Temps d'attente :
Durée moyenne que le processus passe à attendre.
Plus précis que le temps de rotation.

Temps de réponse :

Temps moyen pour que le système répond aux entrées de l'utilisateur

À maximiser	À minimiser
Utilisation UCT Débit	Temps de réponse Temps de rotation Temps d'attente

Temps de rotation=Temps fin d'exécution - Temps d'arrivée

Temps d'attente=Temps de rotation – Durée d'exécution

$$\text{Temps moyen d'attente} = \frac{\sum \textit{Temps attente}}{\textit{nbre de processus}}$$

$$\text{Rendement} = \frac{\sum \textit{Temps d'exécution}}{\textit{nbre de processus}}$$

Les Algorithmes d'ordonnancement :

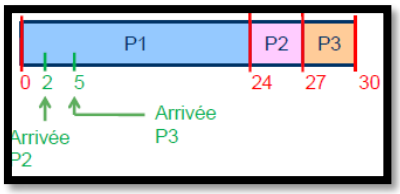
1. FCFS (First Come First Served) :

Processus	Durée
P1	24
P2	3
P3	3

Temps d'attente : P1 = 0 || P2 = 24 || P3 = 27
Temps d'attente moyen = (0+24+27) / 3 = 17
Temps de traitement moyen = (24+27+30) / 3 = 27
Utilisation UCT = 100%
Débit = 3 /30 = 0.1
=> 3 Processus complétés en 30 unités de temps

Temps d'attente : P1 = 6 || P2 = 0 || P3 = 3
Temps d'attente moyen = (6+0+3) / 3 = 3
Temps de traitement moyen = (3+6+30) / 3 = 13
➡ Le temps varie selon l'ordre d'arrivé

Processus	Durée	Arrivée
P1	24	0
P2	3	2
P3	3	5



Temps d'attente : P1 = 0 || P2 = 24-2 =22 || P3 = 27-5 = 22

Inconvénients :

- Les processus à longue durée précédent toujours ceux à faible temps d'exécution
- Temps d'attente non proportionnel au temps d'utilisation :
  - Pas équitable
  - Temps moyen de traitement élevé

2. SJF (Shortest Job First) :

- Sélection du processus avec **le moins** de temps d'exécution
- Ordonnancement **non-préemptif** ou **préemptif**
- Difficile à implémenter
- Optimal

Sans réquisition :

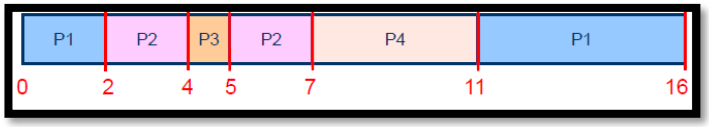
Le scheduler choisi le processus prêt ayant **le plus petit temps d'exécution**, et ne le suspend jamais jusqu'à son achèvement.

Avec réquisition :

Le scheduler cherche parmi les processus prêts celui ayant **le plus petit temps d'exécution restant**.

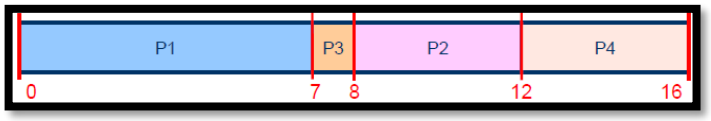
Processus	Durée	Arrivée
P1	7	0.0
P2	4	2.0
P3	1	4.0
P4	4	5.0

Cas préemptif :



$$T_{A\text{-}Moy} = (9+1+0+2)/4 = 3$$

Cas non-préemptif :



$$T_{A\text{-}Moy} = (0+6+3+7)/4 = 4$$

3. RR (Round-Robin) :

- Le temps de processus est divisé en intervalle de temps **Quantum Q**, chaque processus s'exécute durant son quantum.
- FCFS + Quantum
- Ordonnancement préemptif
- Très utilisé

Fonctionnement :

- ❖ Il alloue le processeur au processus en tête pendant son Q
- ❖ S'il se bloque ou se termine avant la fin de Q, il est automatiquement alloué à un autre en tête de file
- ❖ S'il ne se termine pas au bout de Q, son exécution sera suspendue et alloue à un autre en tête de file
- ❖ Le processus suspendu est inséré en queue de file

Réquisitionnement :

- ❖ Épuisement du Q
- ❖ Fin d'exécution avant Q
- ❖ Demande d'entrée/sortie

4. Ordonnancement par priorité :

Selon l'ordre décroissant de leurs priorités. Le processus à élit est celui avec la plus haute priorité.

2 types de priorité :

1. **Statiques :** Ne change pas durant l'ordonnancement
2. **Dynamiques :** Recalculées après un intervalle de temps X
    - ❖ Évite la famine des processus
    - ❖ Augmentation graduelle de la priorité des processus en basse priorité

Un processus ne peut démarrer que si aucun autre à priorité élevée n'est dans l'état **Ready.**

Si tous les processus ont la même priorité, on utilise la politique **FIFO.**