Gestion des Processus

Plan

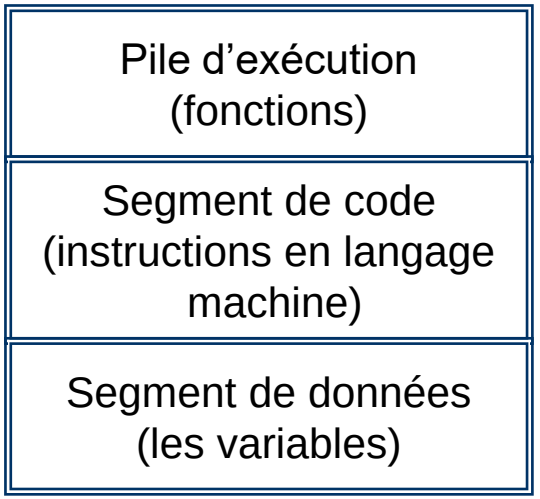
1. Définition
   a. Processus
   b. Threads
2. Le modèle des processus
   a. L'arborescence
   b. Le PCB
   c. Les états d'un processus
   d. Commutation de contexte
   e. Les opérations sur les processus
   f. Réalisations des processus
3. Ordonnancement des processus

Définition: Processus

Le processus est un concept fondamental de tout système d'exploitation. Un processus est l'unité système qui permet l'exécution d'un programme.

Un processus est un programme en exécution. Il définit un objet dynamique tandis que le programme est un objet statique.

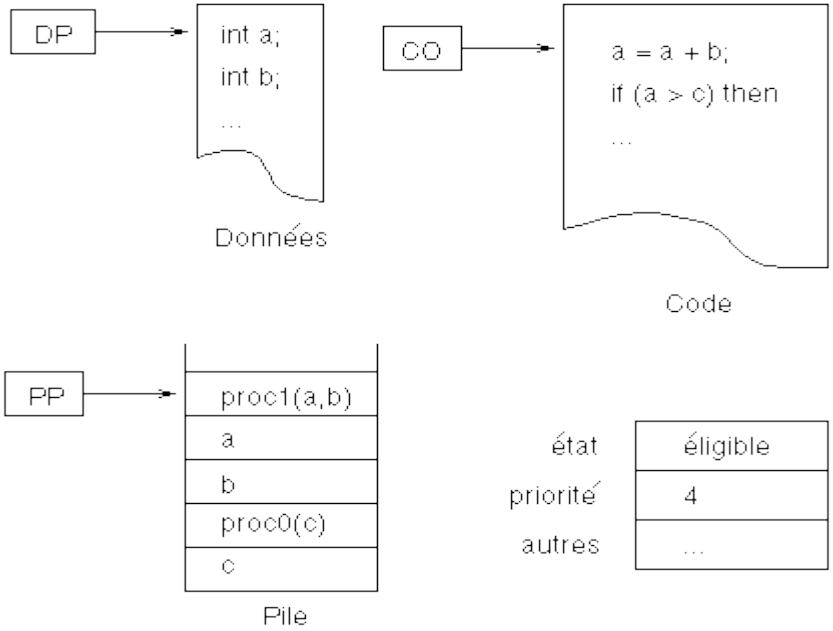
Un processus est caractérisé par son pointeur de pile, ses instructions et ses données, … Ces attributs définissent le **contexte d'un processus**.



Composantes d'un processus

Définition: Processus

- Le contexte d'un processus est l'ensemble des informations dynamiques qui représente l'état d'exécution d'un processus



On définit aussi le vecteur d'état d'un processus (PSW : Program Status Word) comme l'ensemble des bits de condition, priorité, etc. au moment de la commutation de contexte.

Le contexte → état courant d'un processus.

Définition: Thread

Un thread ou encore processus léger (lightweight process) est une unité d'exécution de code. Il est issu d'un processus mais ne contenant que la pile d'exécution.

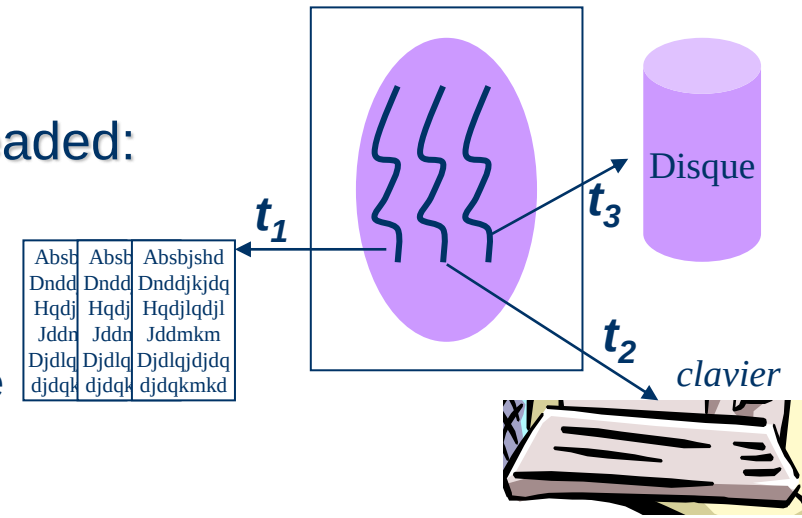
Un processus contient donc au moins un thread de contrôle unique en plus de l'espace d'adressage (segments code et données).

Exemple: traitement de texte multi-threaded:

Thread 1: remet en forme le document

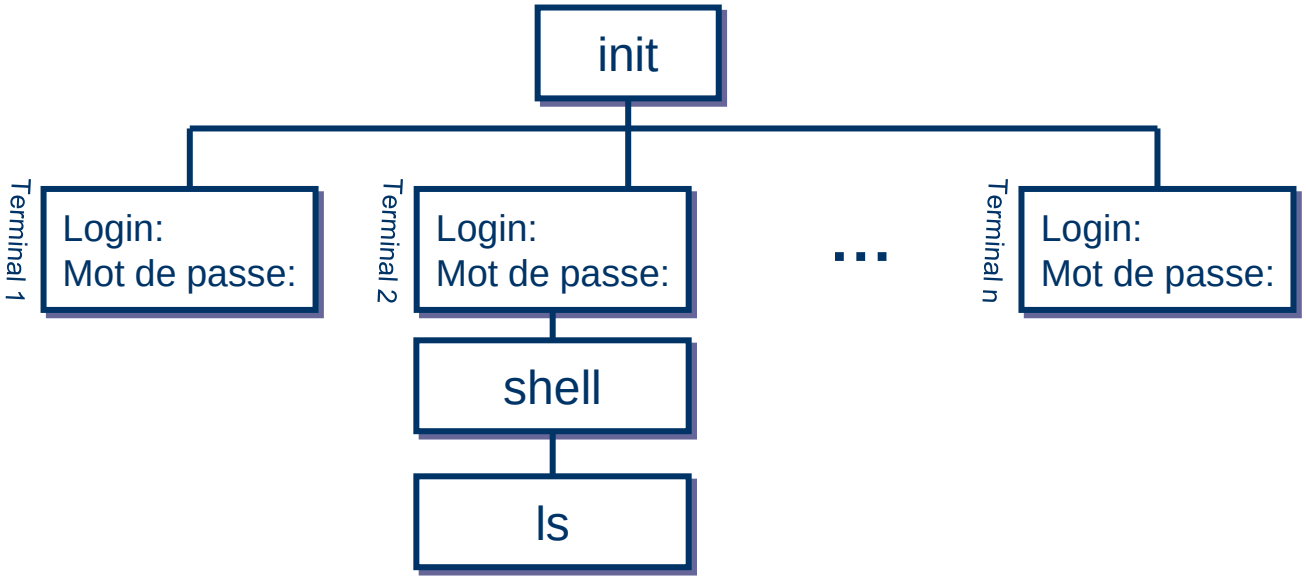
Thread 2: interaction avec l'utilisateur

Thread 3: écrit périodiquement le contenu de la RAM sur le disque



Modèle des processus: Arborescence

- Les processus sont organisés sous forme d'une arborescence ou chaque processus a un seul père et peut avoir plusieurs fils.
- Un processus est identifié par un **PID** (**P**rocess **ID**entifier) et un **PPID** (**P**arent **P**rocess **ID**entifier).
- **Exemple**: L'arborescence des processus sous Linux



Modèle des processus: Arborescence

- **Exemple**: la liste de processus sous Linux

```
$ ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 11:36 ?        00:00:00 init [2]
root         2     1  0 11:36 ?        00:00:00 [migration/0]
root         3     1  0 11:36 ?        00:00:00 [ksoftirqd/0]
root         4     1  0 11:36 ?        00:00:00 [events/0]
root         5     1  0 11:36 ?        00:00:00 [khelper]
...
daemon    2192     1  0 11:36 ?        00:00:00 /sbin/portmap
root      2440     1  0 11:36 ?        00:00:00 /sbin/syslogd
root      2446     1  0 11:36 ?        00:00:00 /sbin/klogd -x
root      2464     1  0 11:36 ?        00:00:00 /usr/sbin/hpiod
...
```

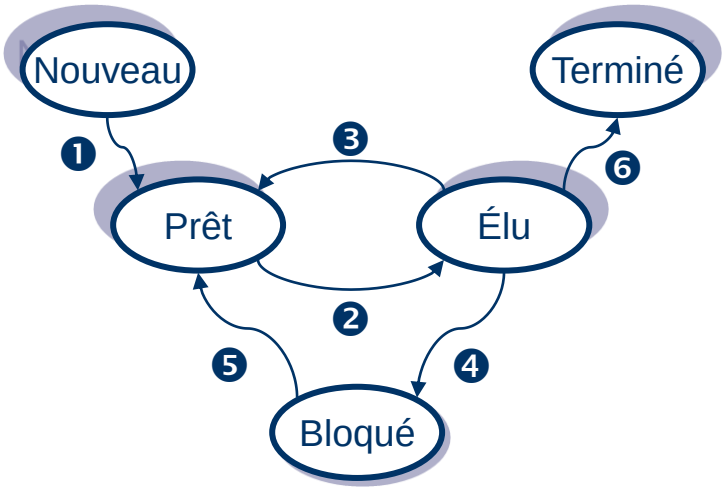
Modèle des processus: États

- Lorsqu'un processus s'exécute; il change d'état. Il peut se trouver dans l'un des trois états principaux suivants:

  • **Prêt** (Ready) : le processus attend son tour pour s'exécuter

  • **Élu** (Running) : les instructions sont encours d'exécution.

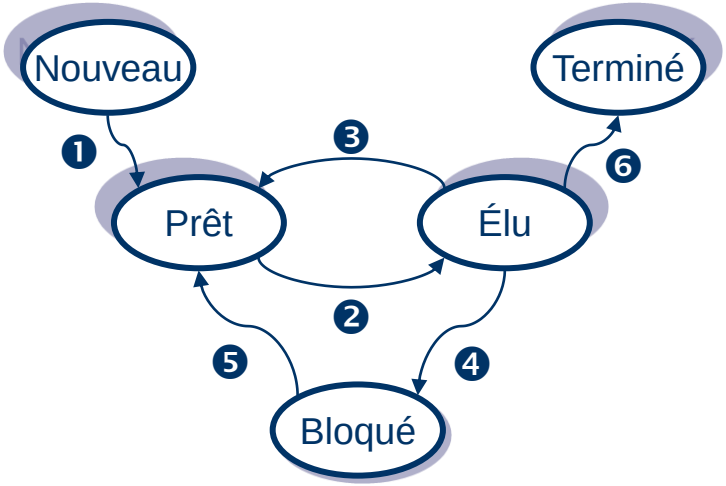
  • **Bloqué** (Sleep) : le processus bloqué en attente d'événement: signal, E/S, …



Graphe des états d'un processus

Modèle des processus: Transitions

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



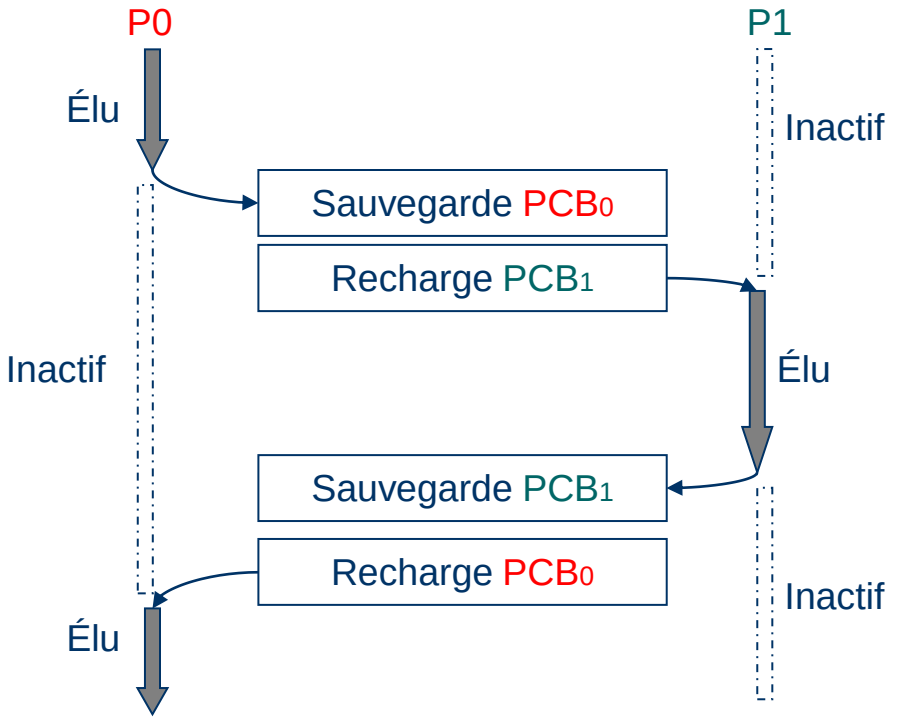
Graphe des états d'un processus

Modèle des processus: PCB

- Chaque processus est représenté dans le système d'exploitation par une structure de données contenant toute information décrivant le contexte du processus appelé bloc de contrôle (**Process Control Bloc: PCB**).
- Attributs d'un **PCB**:
  - PID et PPID,
  - État,
  - Priorité,
  - Compteur ordinal,
  - Fichiers ouverts,
  - Pointeurs: seg. code, seg. données, seg. Pile,
  - Temps d'exécution.

Modèle des processus: Commutation de contexte

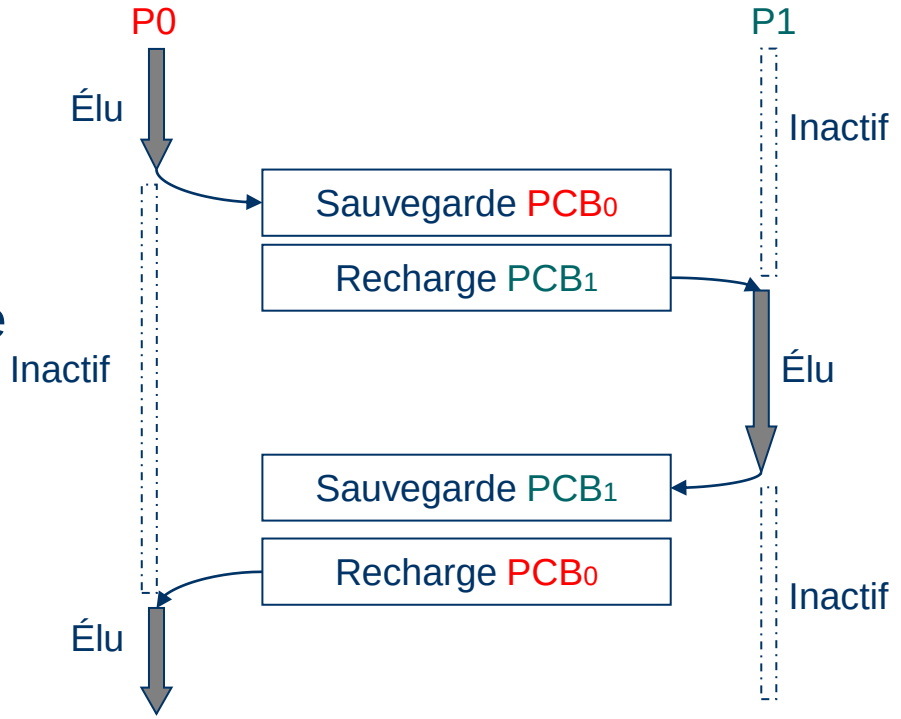
- Sur un système multiprogrammé, le SE doit redonner le contrôle du processeur d'un processus à un autre en effectuant des commutations de contexte.
- La commutation de contexte consiste à mémoriser le PCB du processus courant et charger le PCB du processus à élire.



Modèle des processus: Commutation de contexte

- Le basculement d'un processus à l'autre est géré par le noyau.

- Suspendre le processus P0
- Mettre à jour le PCB de P0
- Restaurer le PCB de P1
- Reconfigurer l'espace mémoire
- Démarrer P1



Modèle des processus: Les opérations sur les processus

Le SE offre 4 opérations:

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

Modèle des processus: Les opérations sur les 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)

Modèle des processus: Les opérations sur les processus

- **Exemple de programme:**

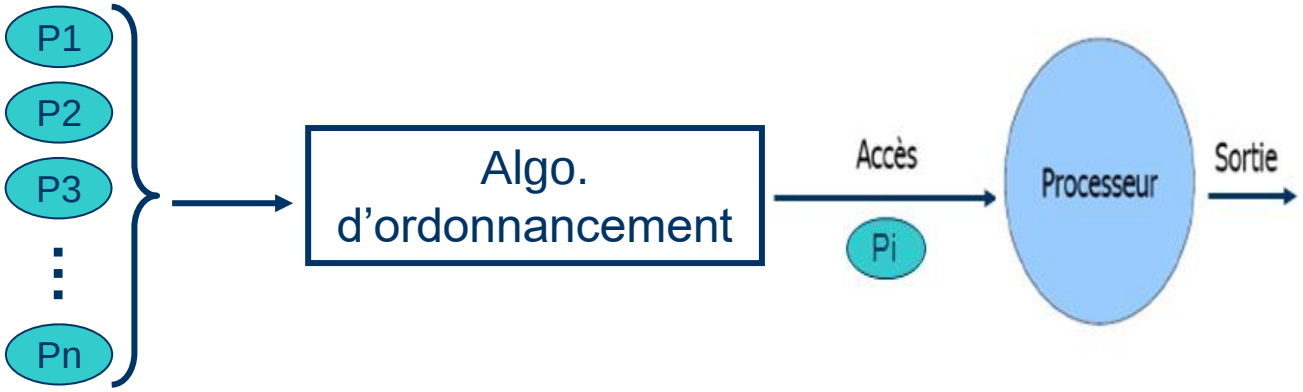
```
#include<stdio.h>
main()
{int n;
if ((n=fork())<0) {perror("fork");exit();}
if (n)
printf(" %d : Je suis le processus père \n",getpid());
else printf(" %d : Je suis le processus fils de %d \n",
getpid(),getppid());
}
```

Modèle des processus: Réalisation des processus

- Le SE utilise gère 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



Étant donnée un ensemble de processus prêts, l'Ordonnanceur (**scheduler**) du SE doit choisir quel processus élire en utilisant un algorithme d'ordonnancement.

Ordonnancement des processus

Un bon algorithme d'ordonnancement doit être capable de:

1. Chaque processus doit avoir sa part de temps CPU: **équité**.
2. Utiliser le temps processeur a 100%: **efficacité.**
3. Minimiser le **temps de réponse** en mode interactif.

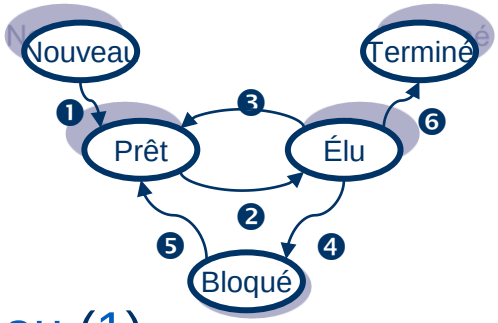
Il existe deux types d'algorithmes d'ordonnancement:

1. Ordonnancement sans réquisition (sans préemption): exécution d'un processus jusqu'à sa terminaison.
2. Ordonnancement avec réquisition (avec préemption):suspension possible d'un processus élu même s'il peut continuer son exécution.

Ordonnancement des processus

Quand invoquer l'ordonnanceur

- Chaque fois que le processus exécutant est interrompu
  - un processus exécutant devient bloqué (4)
  - un processus change d'élu à prêt (3)
  - un processus exécutant se termine (6)



- Chaque fois qu'un nouveau processus est prêt
  - un processus se présente en tant que nouveau (1)
  - un processus change de bloqué à prêt (5)

- L'ordonnanceur choisi un processus parmi les processus prêts et lui alloue le processeur

Ordonnancement des processus

Les algorithmes d'ordonnancement:

- **FCFS:** First Come First Served
- **SJF:** Shortest Job First
- **RR:** Round-Robin
- **Ordonnancement par priorité**

Algorithmes Ordonnancement

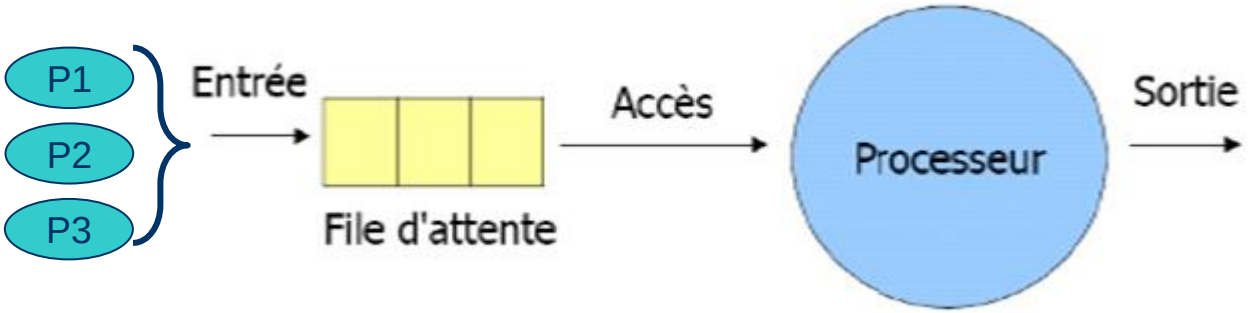
1. **Premier Arrivée Premier Servi (First Come First Served FCFS):** Exécution des processus dans leur ordre d'arrivée chronologique

   - Gestion des processus à l'aide d'un FIFO
   - Ordonnancement **non-préemptif**
   - Facile à implémenter
   - Sous-optimal

Algorithmes Ordonnancement

Exercice d'application:

Donner le diagramme de Gantt sachant que l'ordonnancement sur le processeur se fait selon une stratégie **FCFS**. Donner les différents cas possibles.



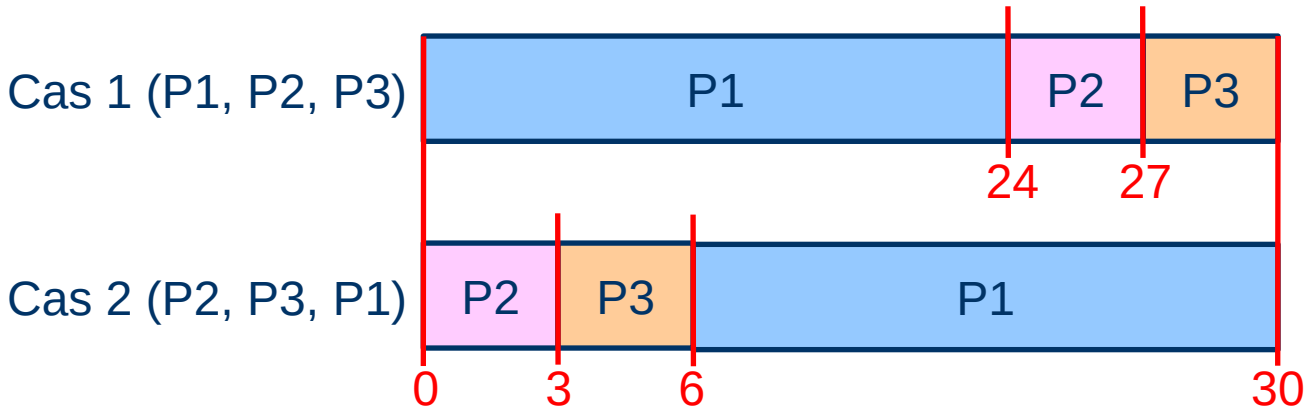
Processus	Durée
P1	24
P2	3
P3	3

Algorithmes Ordonnancement

Correction de l'exercice d'application:

- **Algorithme d'ordonnancement FCSF**

Processus	Durée
P1	24
P2	3
P3	3



Algorithmes Ordonnancement

2. Plus Court d'Abord (Short Job First SJF):

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

Algorithmes Ordonnancement

2. Plus Court d'Abord (Short Job First SJF):

a. SJF sans réquisition

Le scheduler choisi le processus prêt **ayant le plus petit temps d'exécution**. Une fois un processus est élu, il n'est jamais suspendu jusqu'à la fin de son exécution.

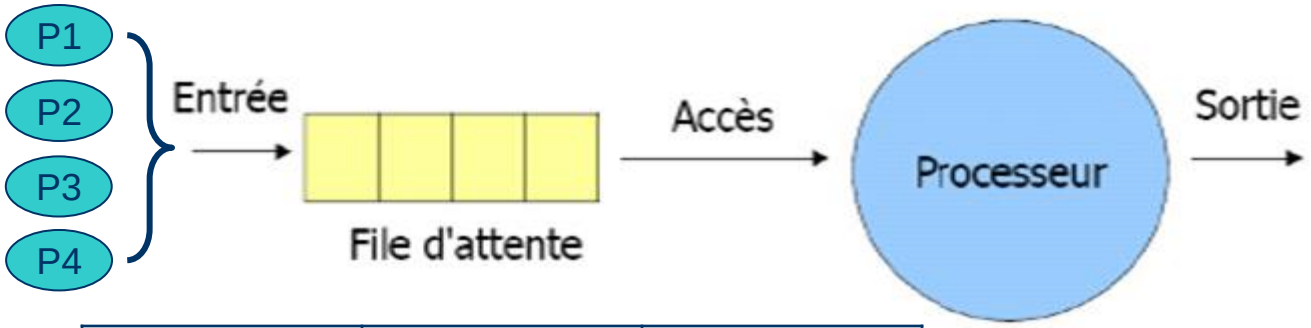
b. SJF avec réquisition (Short Next Remaining Time SNRT)

A chaque instant, le scheduler cherche parmi les processus prêts celui **ayant le plus petit temps d'exécution restant**.

Algorithmes Ordonnancement

Exercice d'application:

Donner le diagramme de Gantt sachant que l'ordonnancement sur le processeur se fait selon une stratégie **SJF (sans réquisition et avec réquisition).**



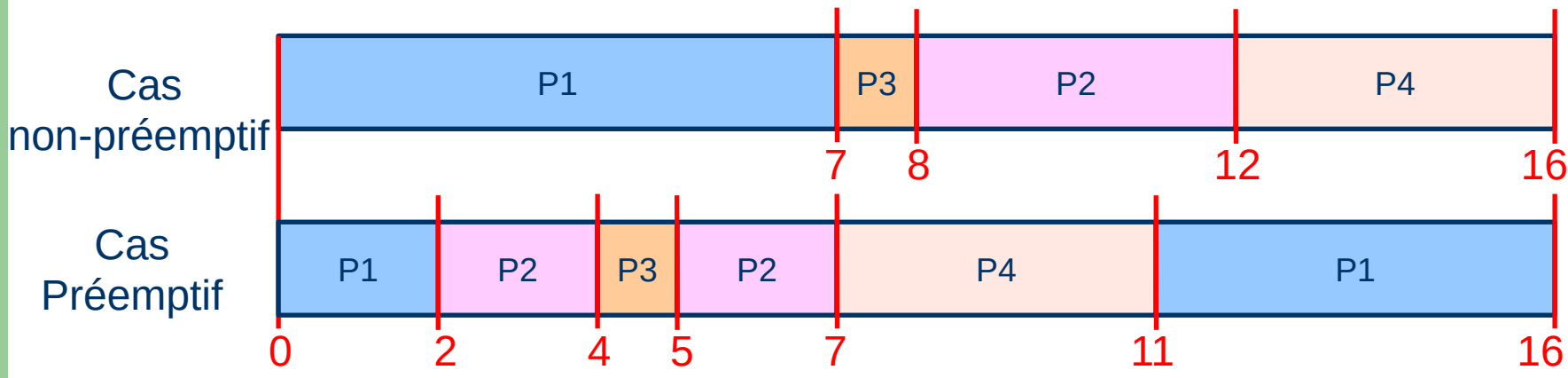
Processus	Durée	Arrivée
P1	7	0.0
P2	4	2.0
P3	1	4.0
P4	4	5.0

Algorithmes Ordonnancement

Correction de l'exercice d'application:

- **SJF (sans et avec réquisition)**

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



Algorithmes Ordonnancement

Exercice d'application:

Exercice 1 :

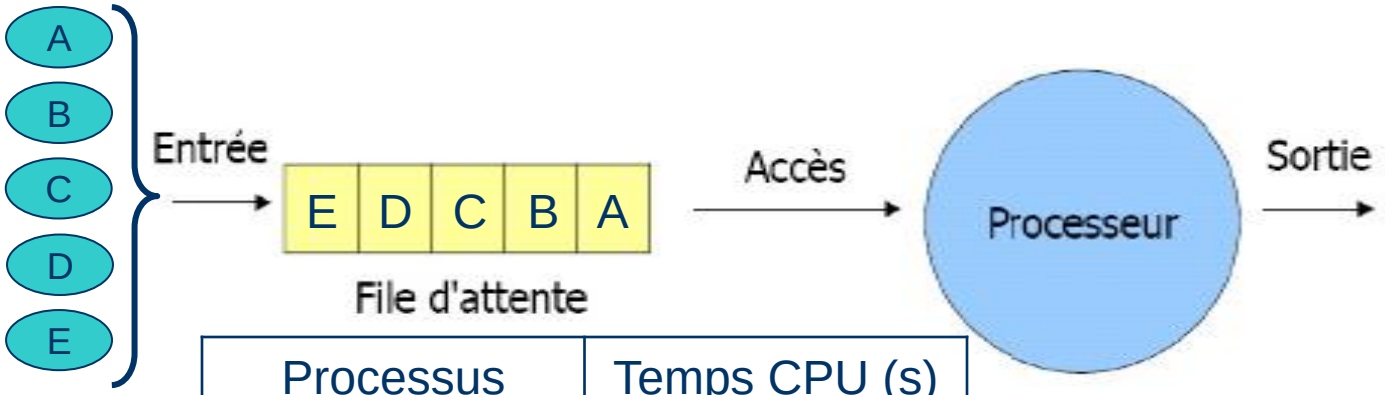
Cinq travaux A, B, C, D et E arrivent pratiquement en même temps dans un centre de calcul. Leur temps d'exécution respectif est estimé à 10, 6, 2, 4 et 8 secondes.

Tracez le digramme de Gantt et déterminez le temps moyen de traitement pour chacun des algorithmes d'ordonnancement suivants. Ne tenez pas compte du temps perdu lors de la commutation des processus.

- Premier arrivé, premier servi FCFS (exécution dans l'ordre 10, 6, 2, 4, 8) ;
- Plus court d'abord SJF ;

Algorithmes Ordonnancement

Exercice d'application:



Processus	Temps CPU (s)
A	10
B	6
C	2
D	4
E	8

Algorithmes Ordonnancement

Correction de l'exercice d'application:

- **FCSF (exécution dans l'ordre 10, 6, 2, 4, 8)**

  Temps moyen de traitement = (10 + 16 + 18 + 22 + 30) / 5 = 19.2

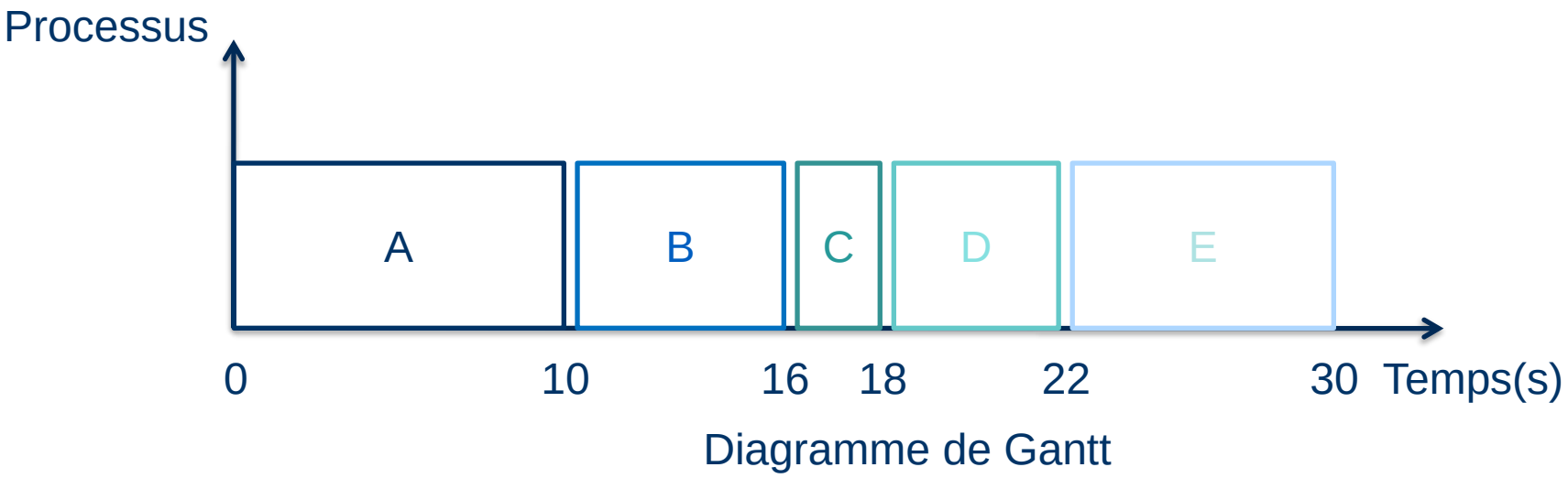


Diagramme de Gantt

Algorithmes Ordonnancement

Correction de l'exercice d'application:

- **SJF sans réquisition**

Temps moyen de traitement = (2 + 6 + 12 + 20 + 30) / 5 = 14

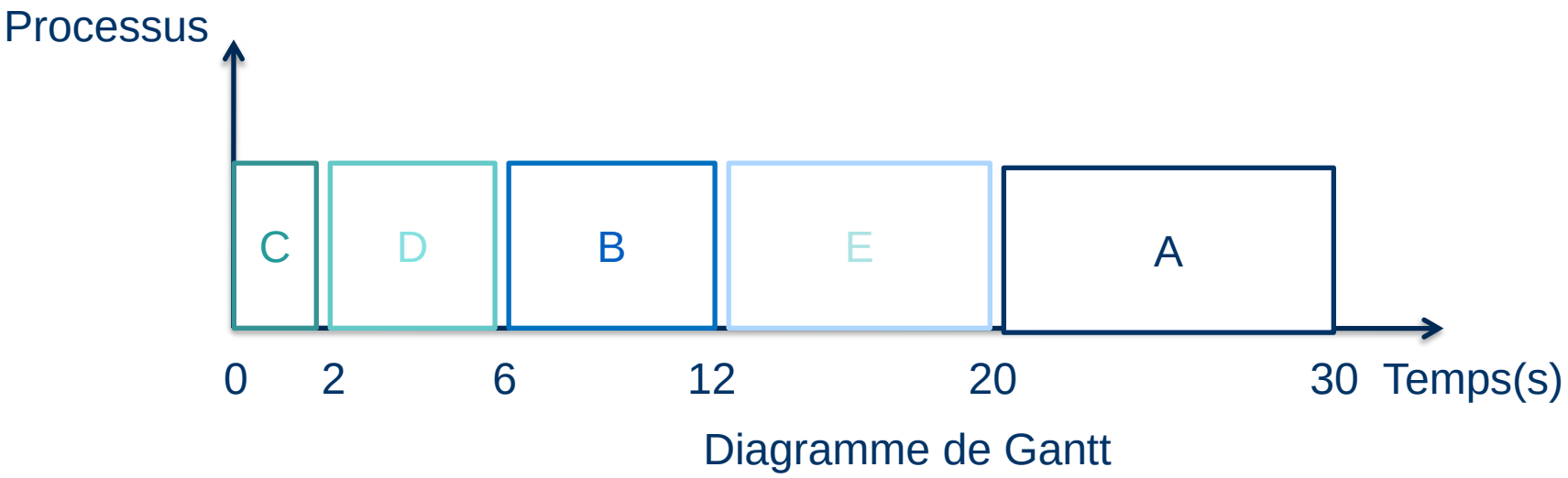


Diagramme de Gantt

Algorithmes Ordonnancement

3. **Tourniquet (Round Robin, l'algorithme circulaire)**

   Le temps processeur est diviser en intervalles de temps appelés **Quantum Q**, chaque processus s'exécutera exactement pendant son quantum.

   • Il s'agit d'un FCFS avec l'introduction d'un **Quantum Q**

   • Ordonnancement préemptif

   • Très utilisé

Algorithmes Ordonnancement

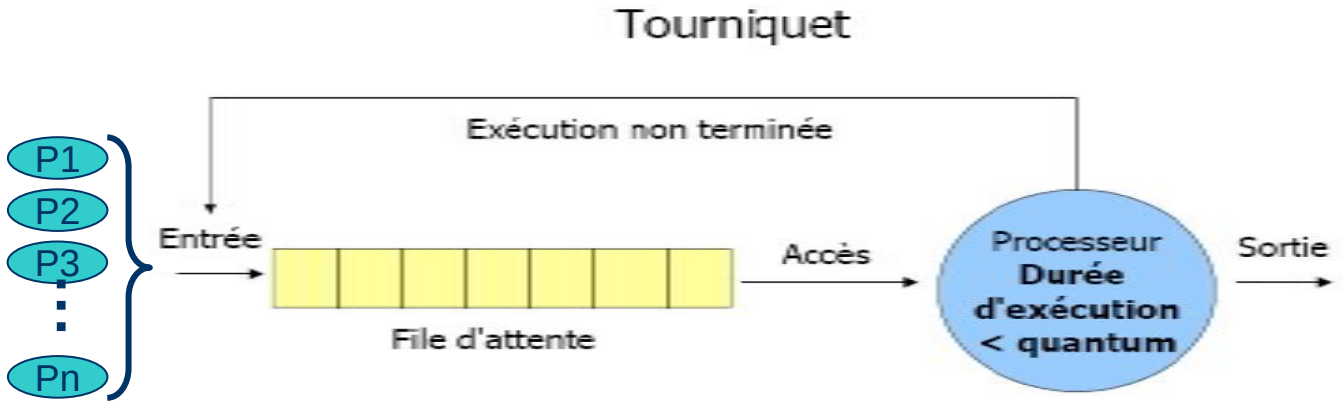
Choix du processus à exécuter :

• Il alloue le processeur au processus en tête de file, pendant un quantum.

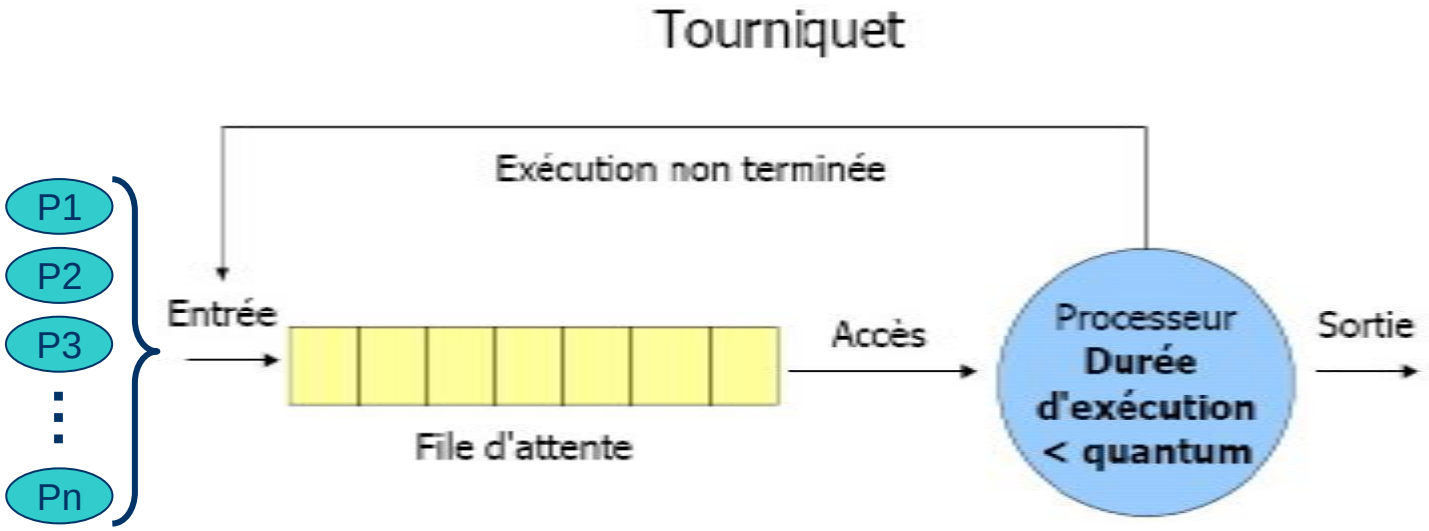
• Si le processus se bloque ou se termine avant la fin de son quantum, le processeur est immédiatement alloué à un autre processus (en tête de file).

• Si le processus ne se termine pas au bout de son quantum, l'exécution du processus est suspendue. Le processeur est alloué à un autre processus (en tête de file).

• Le processus suspendu est inséré en queue de file.



Algorithmes Ordonnancement



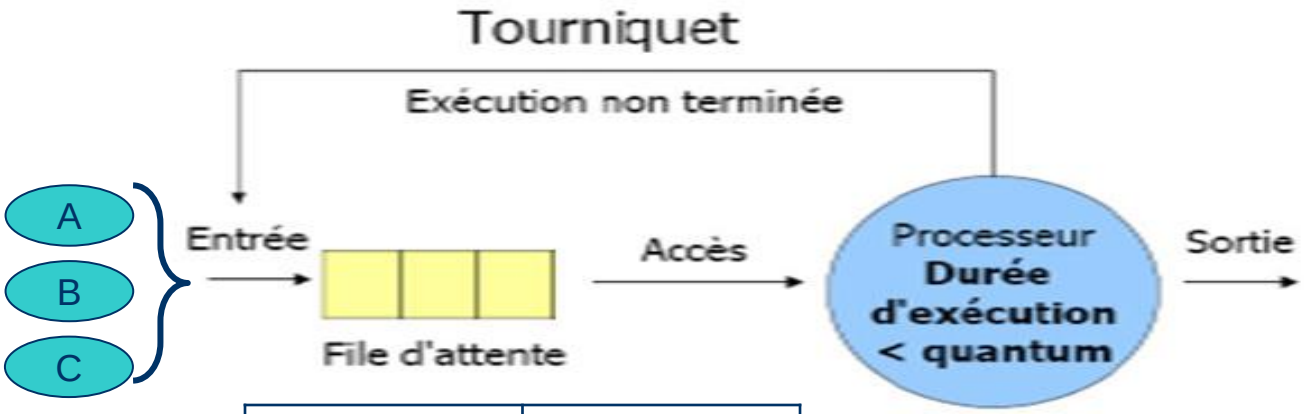
Le processeur sera réquisitionner:

a. Le processus élu a épuisé son quantum,
b. Le processus élu a fini son exécution avant la fin de son quantum,
c. Le processus élu demande une entrée/sortie.

Algorithmes Ordonnancement

Exercice d'application:

Donner le diagramme de Gantt sachant que l'ordonnancement sur le processeur se fait selon une stratégie **Round Robin (Tourniquet)**



Processus	Durée
A	24
B	3
C	5

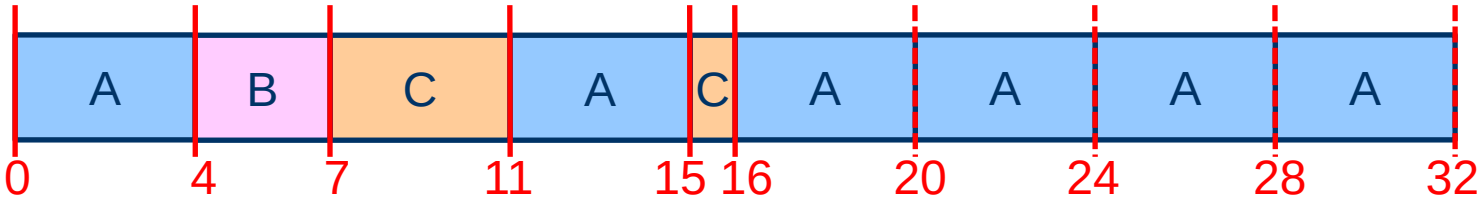
Quantum = 4 ms

Algorithmes Ordonnancement

Correction de l'exercice d'application:

- **Tourniquet (Quantum = 4 ms)**

Processus	Durée
A	24
B	3
C	5



Algorithmes Ordonnancement

4.Ordonnancement avec priorité

Le système d'exploitation ordonne les processus prêts selon l'ordre décroissant de leurs priorités et le processus à élire est celui avec la plus haute priorité.

Les priorités utilisées peuvent être **statiques** ou **dynamiques**. Les priorités statiques ne changent pas au cours de l'ordonnancement.

Les priorités dynamiques seront recalculées périodiquement après un intervalle de temps bien défini.

Algorithmes Ordonnancement

4.Ordonnancement avec priorité

• Un processus ne peut s'exécuter que si aucun processus de priorité supérieure n'est dans l'état *ready.*

• Tous les processus ont la même priorité c'est la politique FIFO qui est appliquée**.**

• **Priorité dynamique**
  – Augmentation graduelle de la priorité des processus en basse priorité,
  – Evite la famine des processus.

• **Ordonnancement**
  – **non-préemptif** ou ***préemptif***

Algorithmes Ordonnancement

Exercice d'application:

Donner le diagramme de Gantt sachant que l'ordonnancement sur le processeur se fait selon une stratégie **avec priorité statique (dans l'ordre croissant).**

Processus	Durée	Priorité
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

Algorithmes Ordonnancement

Correction de l'exercice d'application:

- **Priorité**

Processus	Durée	Priorité
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

Algorithmes Ordonnancement

FCFS:

Processus	Durée
P1	24
P2	3
P3	3

Si les processus arrivent au temps 0 dans l'ordre: P1 , P2 , P3

Le diagramme Gantt est:

- Calculer le Temps d'Attente pour P1, P2 et P3
- Calculer le Temps Attente Moyen (TAM)
- Calculer le Temps d'Traitement Moyen (TTM)

Algorithmes Ordonnancement

FCFS:

- Temps d'Attente pour P1= 0; P2= 24; P3= 27
- Temps 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

Algorithmes Ordonnancement

FCSF: Tenir compte du temps d'arrivée!

- Dans le cas où les processus arrivent à moment différents, il faut soustraire les temps d'arrivée
- Répéter les calculs si:
  - P1 arrive à temps 0 et dure 24
  - P2 arrive à temps 2 et dure 3
  - P3 arrive à temps 5 et dure 3

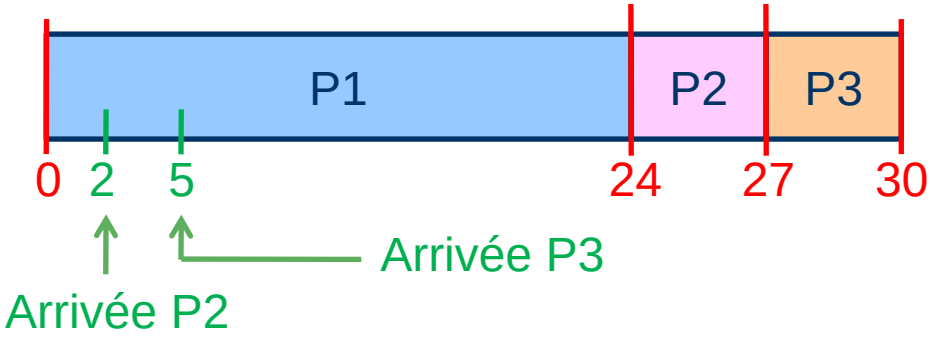
Processus	Durée	Arrivée
P1	24	0
P2	3	2
P3	3	5

Algorithmes Ordonnancement

FCSF: Tenir compte du temps d'arrivée!

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

- Donc P1 attend 0 comme avant
- Mais P2 attend 24-2, et P3 attend 27-5.



Algorithmes Ordonnancement

FCSF: Si les mêmes processus arrivent à 0 mais dans l'ordre

P2 , P3 , P1 .

Le diagramme de Gantt est:

- Calculer le Temps d'Attente pour P1, P2 et P3
- Calculer le TAM
- Calculer le TTM

Algorithmes Ordonnancement

FCSF: Si les mêmes processus arrivent à 0 mais dans l'ordre P2, P3, P1

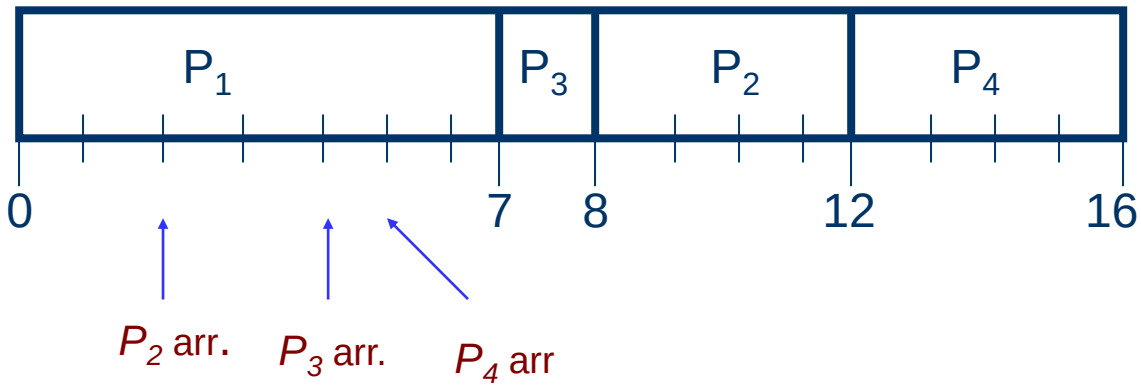
- Temps d'attente P1 = 6; P2 = 0; P3 = 3
- TAM = (6 + 0 + 3)/3 = 3
- TTM = (3+6+30)/3 = 13
- Beaucoup mieux!
- Donc pour cette technique, les temps peuvent varier grandement par rapport à l'ordre d'arrivée de différent processus

Algorithmes Ordonnancement

Exemple de SJF sans préemption

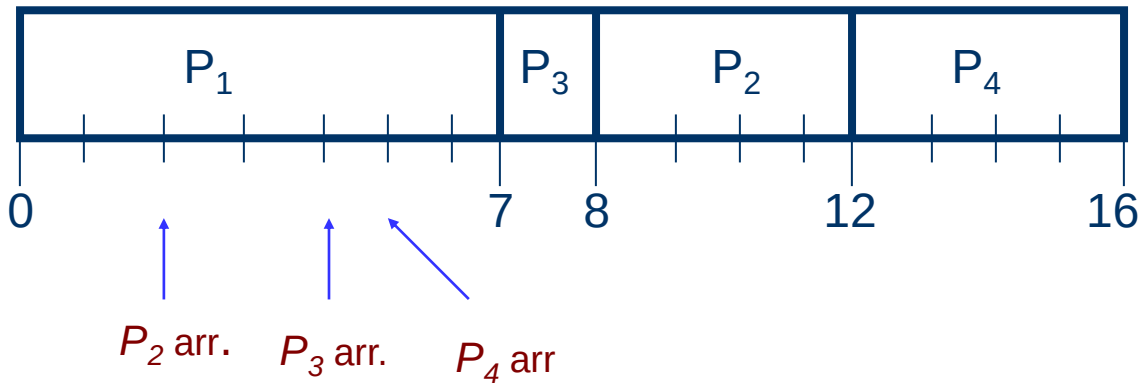
Processus	Arrivée	Durée
P1	0	7
P2	2	4
P3	4	1
P4	5	4

- Calculer le TAM
- Calculer le TTM



Algorithmes Ordonnancement

Exemple de SJF sans préemption



- **TAM =** (0+(8-2)+(7-4)+(12-5))/4
  = (0 + 6 + 3 + 7)/4 = 4

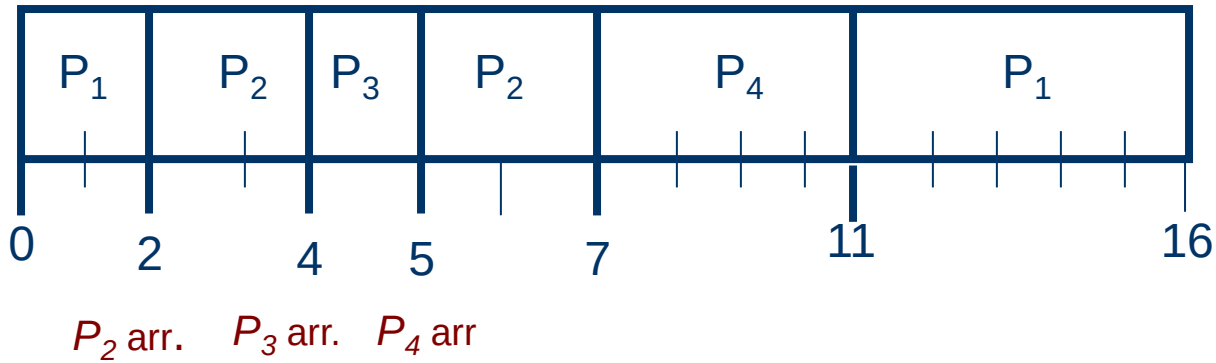
- **TTM = (**7+(12-2)+(8-4)+(16-5))/4 = 8

Algorithmes Ordonnancement

Exemple de SJF avec préemption

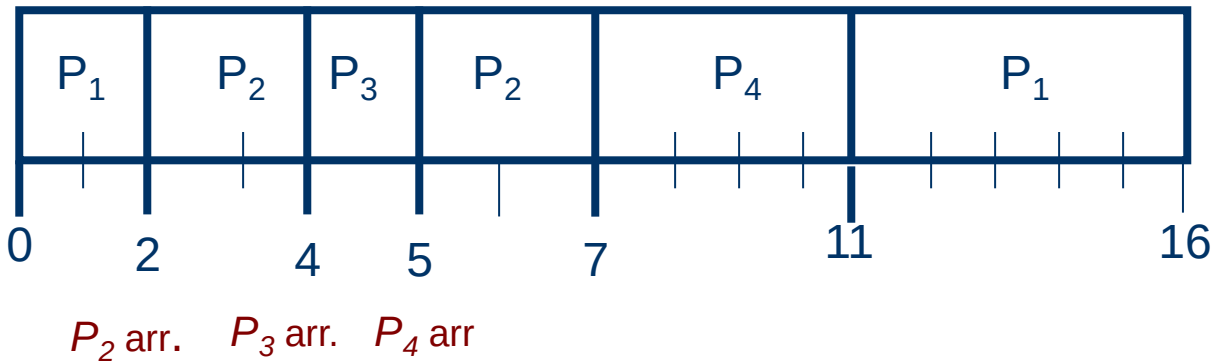
Processus	Arrivée	Durée
P1	0	7
P2	2	4
P3	4	1
P4	5	4

- Calculer le TAM
- Calculer le TTM



Algorithmes Ordonnancement

Exemple de SJF avec préemption



- **TAM =** (9 + 1 + 0 +2)/4 = 3

  P1 attend de 2 à 11, P2 de 4 à 5, P4 de 5 à 7
- **TTM =** (16+ (7-2) + (5-4) + (11-5))/4 = 7

Algorithmes Ordonnancement

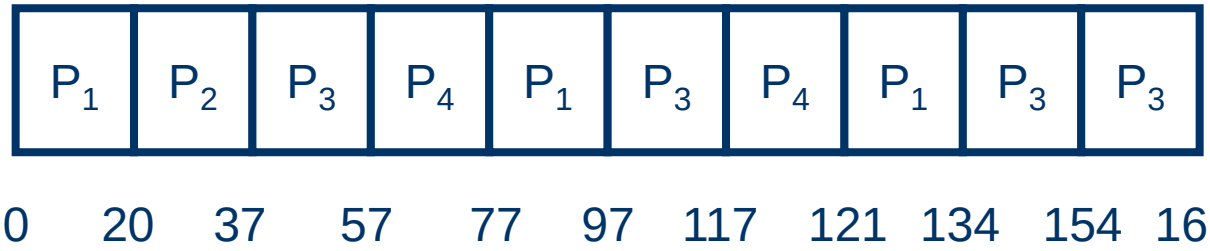
Exemple: Tourniquet quantum = 20

Processus	Durée
P1	53
P2	17
P3	68
P4	24

- Tracer le diagramme de Gantt
- Calculer le TAM et le TTM
- Comparer les résultats par rapport au SJF, qu'observez vous!

Algorithmes Ordonnancement

Exemple: Tourniquet quantum = 20



Observez

- Temps de traitement et temps d'attente moyens beaucoup plus élevés que SJF.
- Mais aucun processus n'est favorisé.

Exercice

Q=3

Process	Arrival time	1st exec	1st I/O	2nd exec	2nd I/O	3rd exec
A	0	4	4	4	4	4
B	2	8	1	8	-	-
C	3	2	1	2	-	-
D	7	1	1	1	1	1