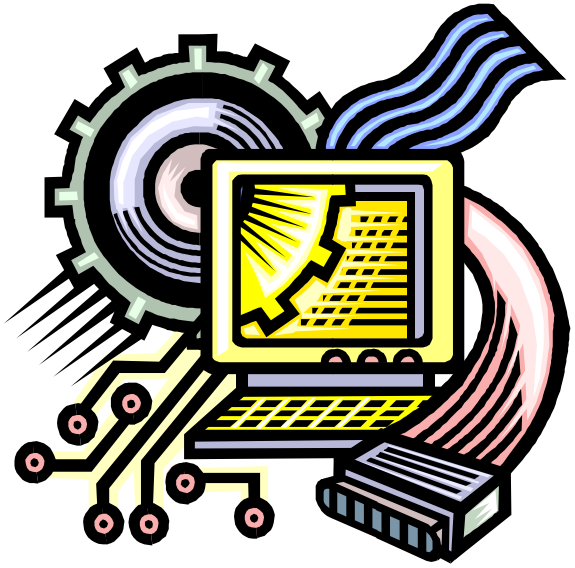
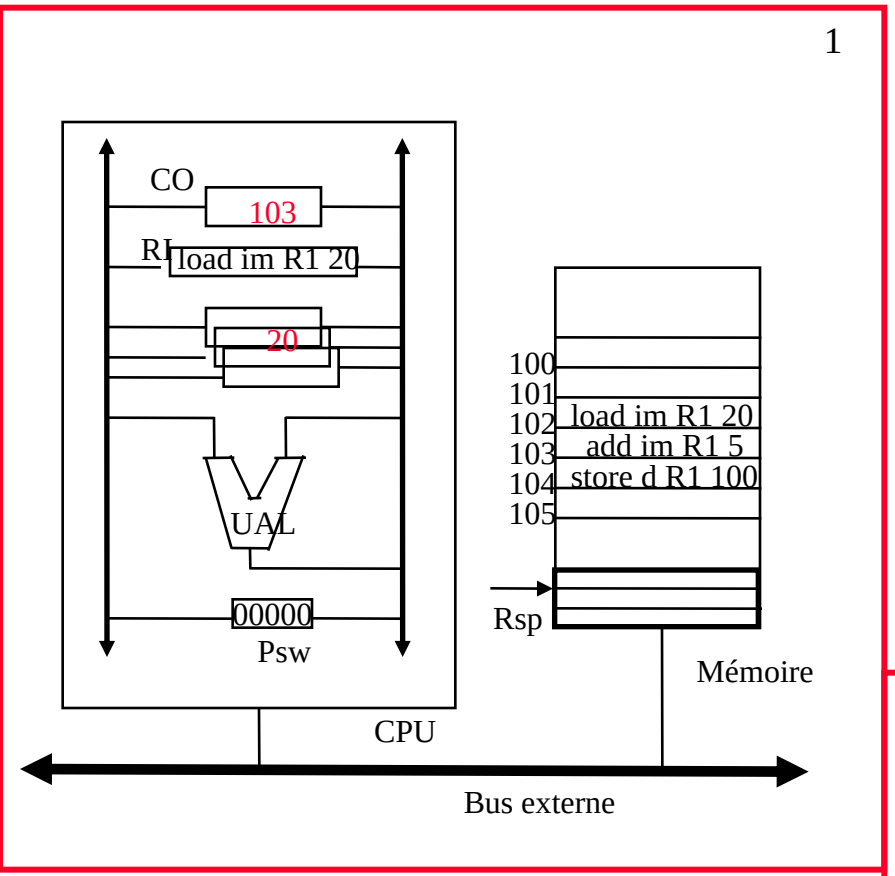
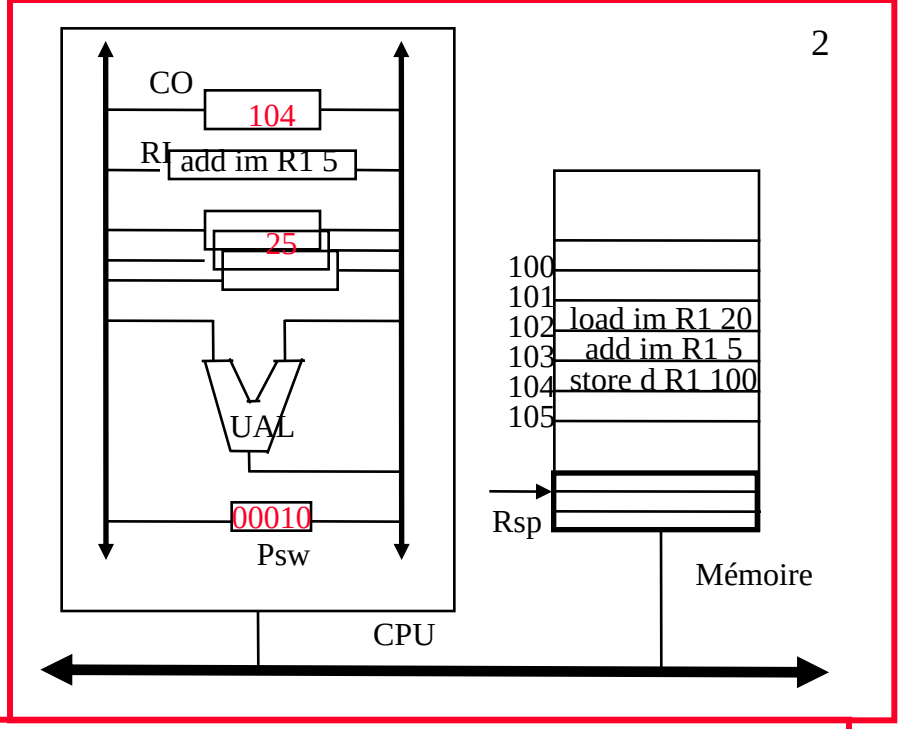
I.
Processus- Ordonnancement

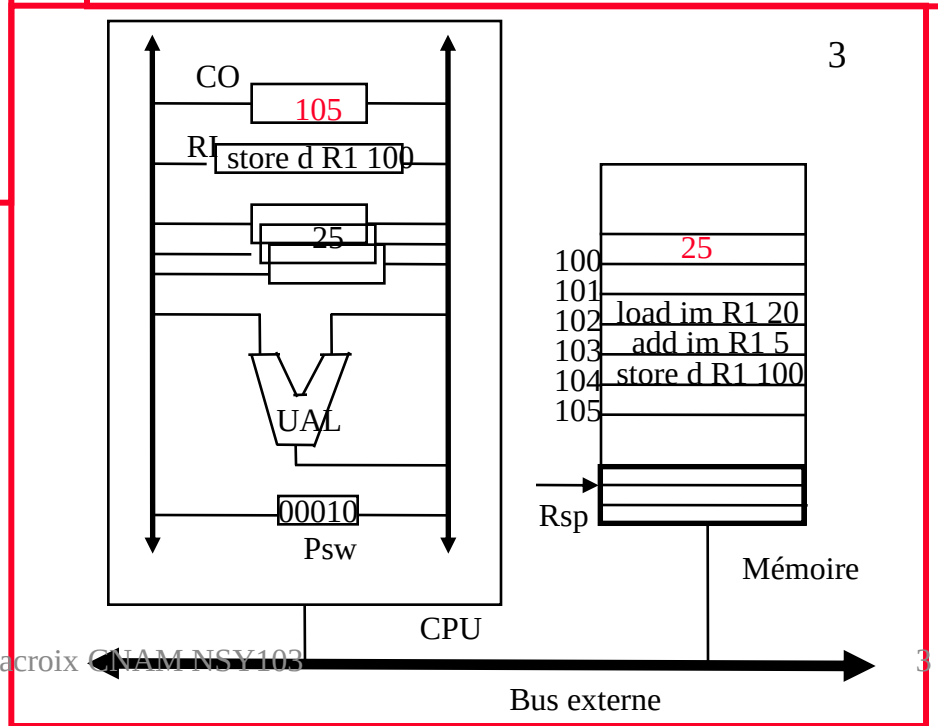


Processus

Un processus est une exécution de programme







- ✓CO : compteur ordinal
- ✓PSW : registre d'état
- ✓RI : registre instruction

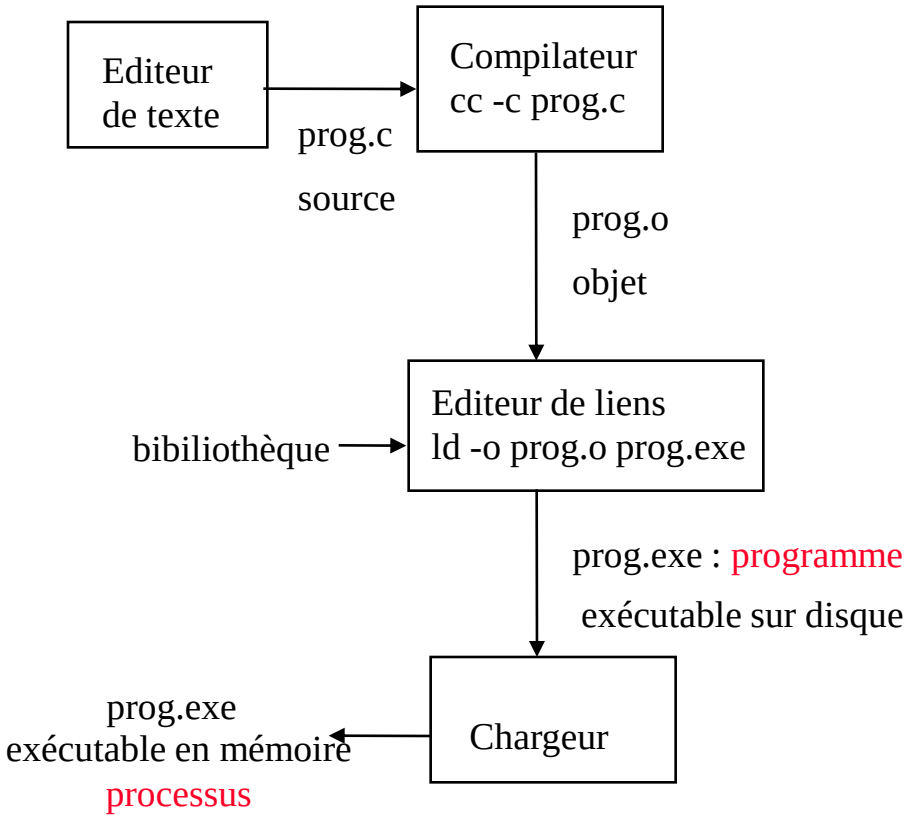
Notion de processus

- **Définition**

  - **Un processus est un programme en cours d ’exécution auquel est associé un environnement processeur (CO, PSW, registres généraux) et un environnement mémoire appelés contexte du processus.**

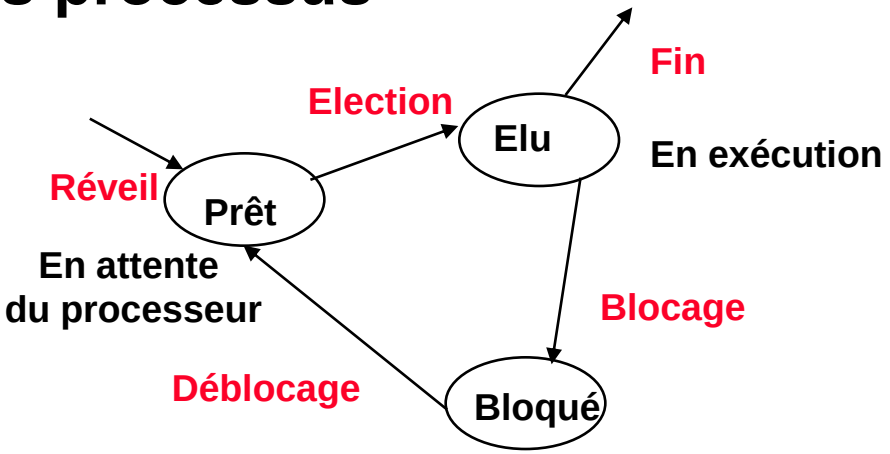
  - **Un processus est l ’instance dynamique d ’un programme et incarne le fil d ’exécution de celui-ci**

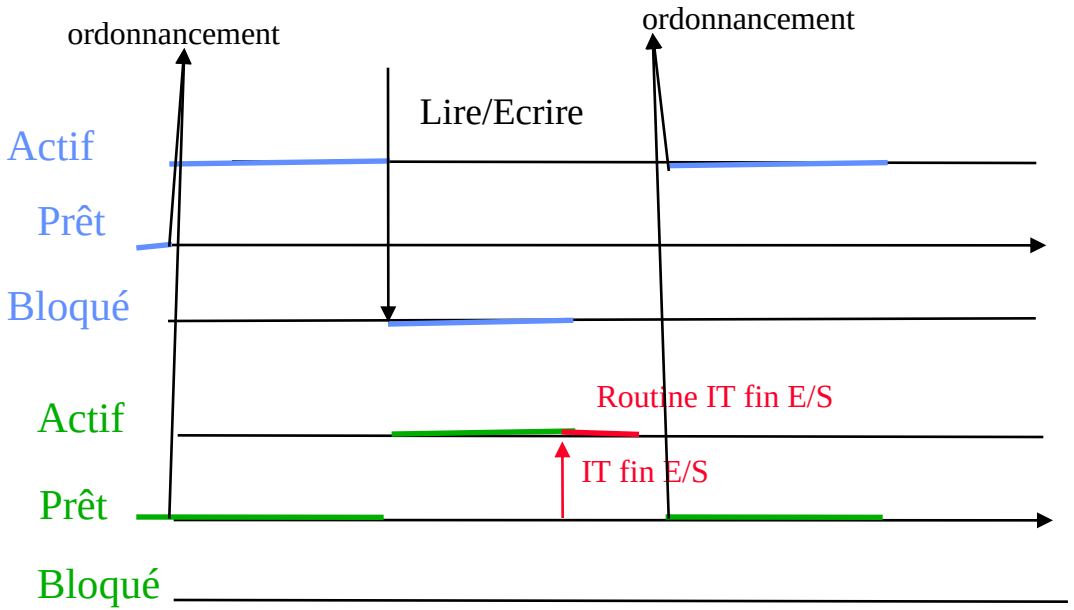
  - **Un processus évolue dans un espace d’adressage protégé**



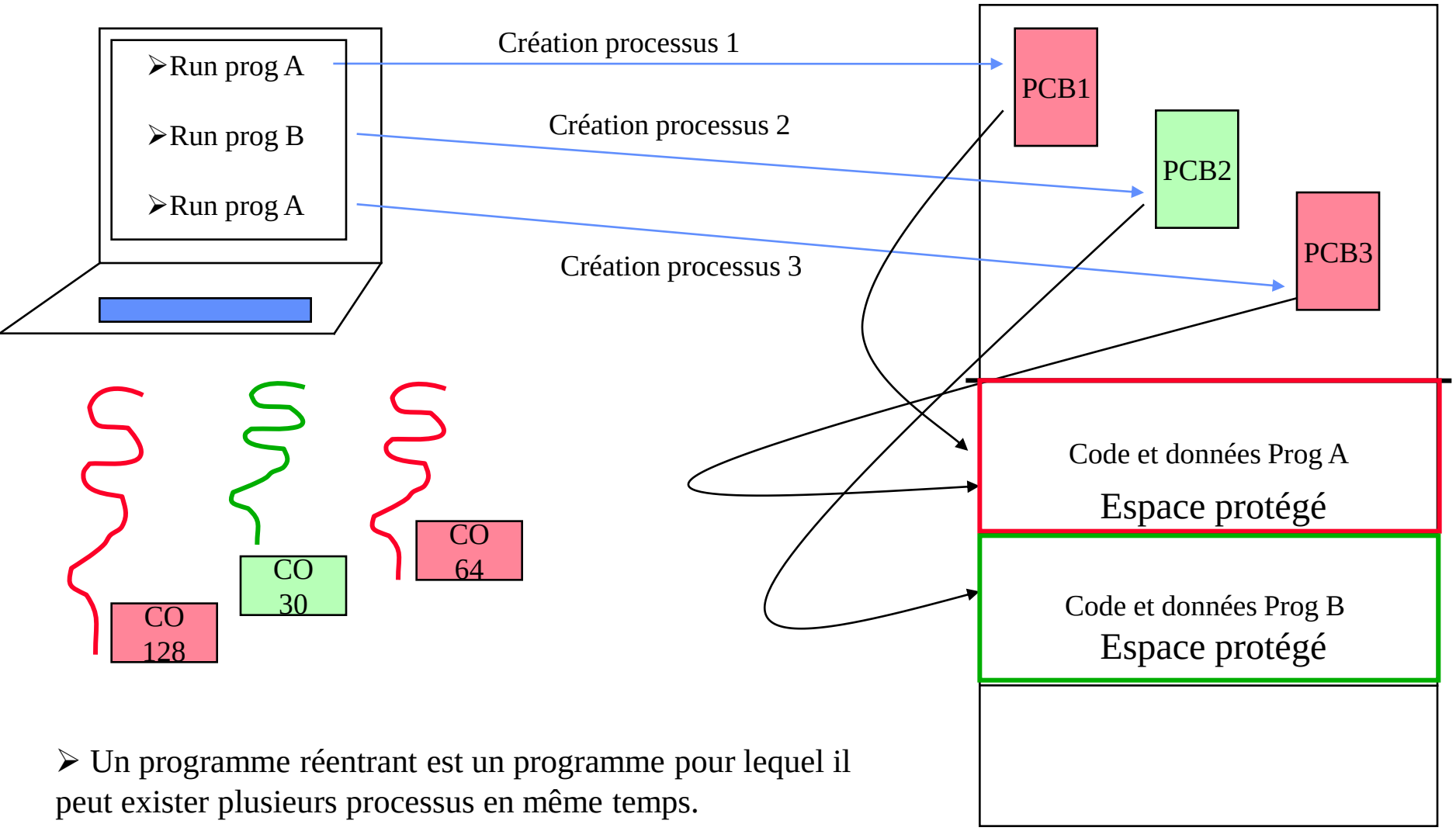
bloc de contrôle de processus PCB
Etats des processus

identificateur processus
état du processus
contexte pour reprise (registres et pointeurs, piles,..) compteur instructions
pointeurs sur file d' attente et priorité(ordonnancement)
informations mémoire (limites et tables pages/segments
informations de comptabilisation et sur les E/S, périphériques alloués, fichiers ouverts,..





Notion de processus
programme réentrant



➢ Un programme réentrant est un programme pour lequel il peut exister plusieurs processus en même temps.

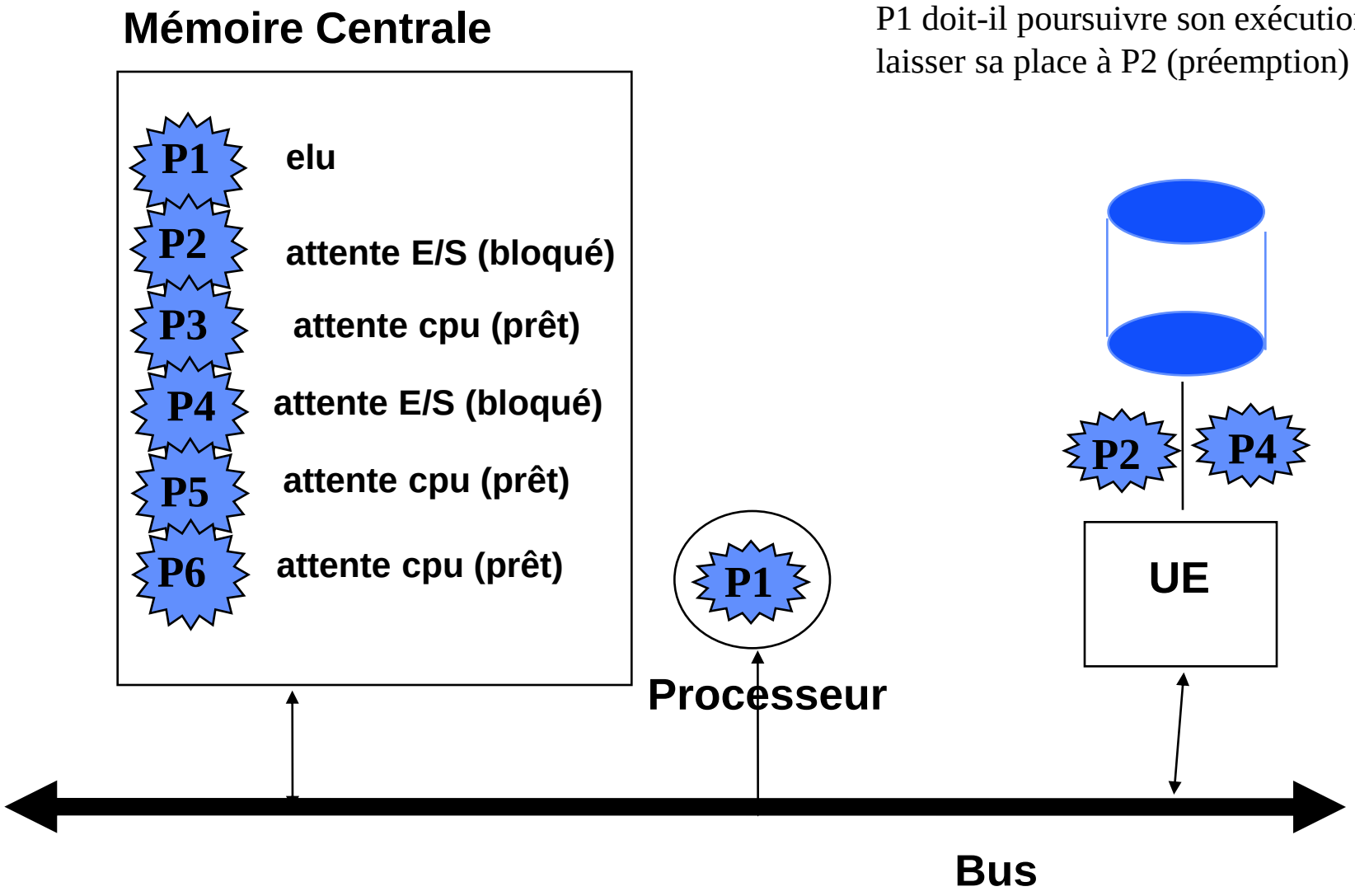
Ordonnancement dans un système multiprocessus

Cette fonction planifie l'exécution des processus

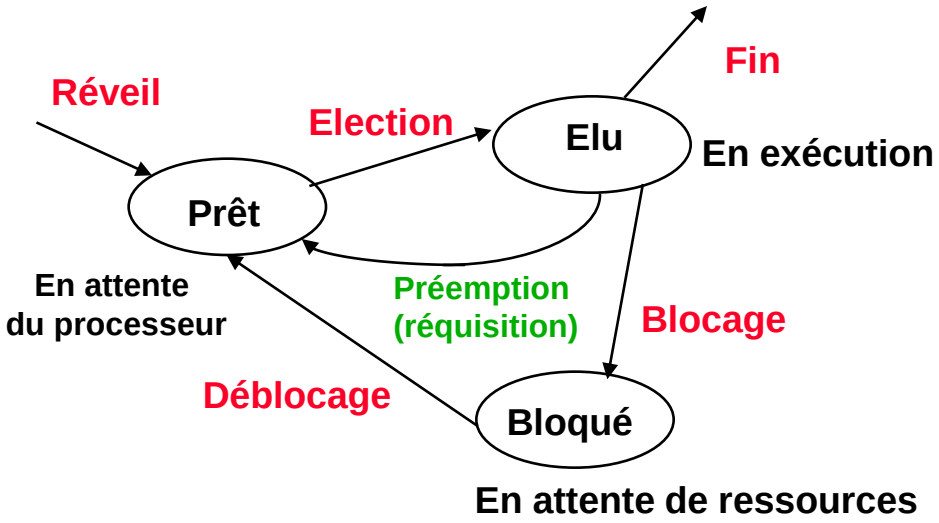
Système multiprocessus

P1 se termine : quel processus exécuter parmi les processus prêts P3, P5, P6

P2 achève son entrées-sortie et devient prêt : P1 doit-il poursuivre son exécution ou laisser sa place à P2 (préemption) ?



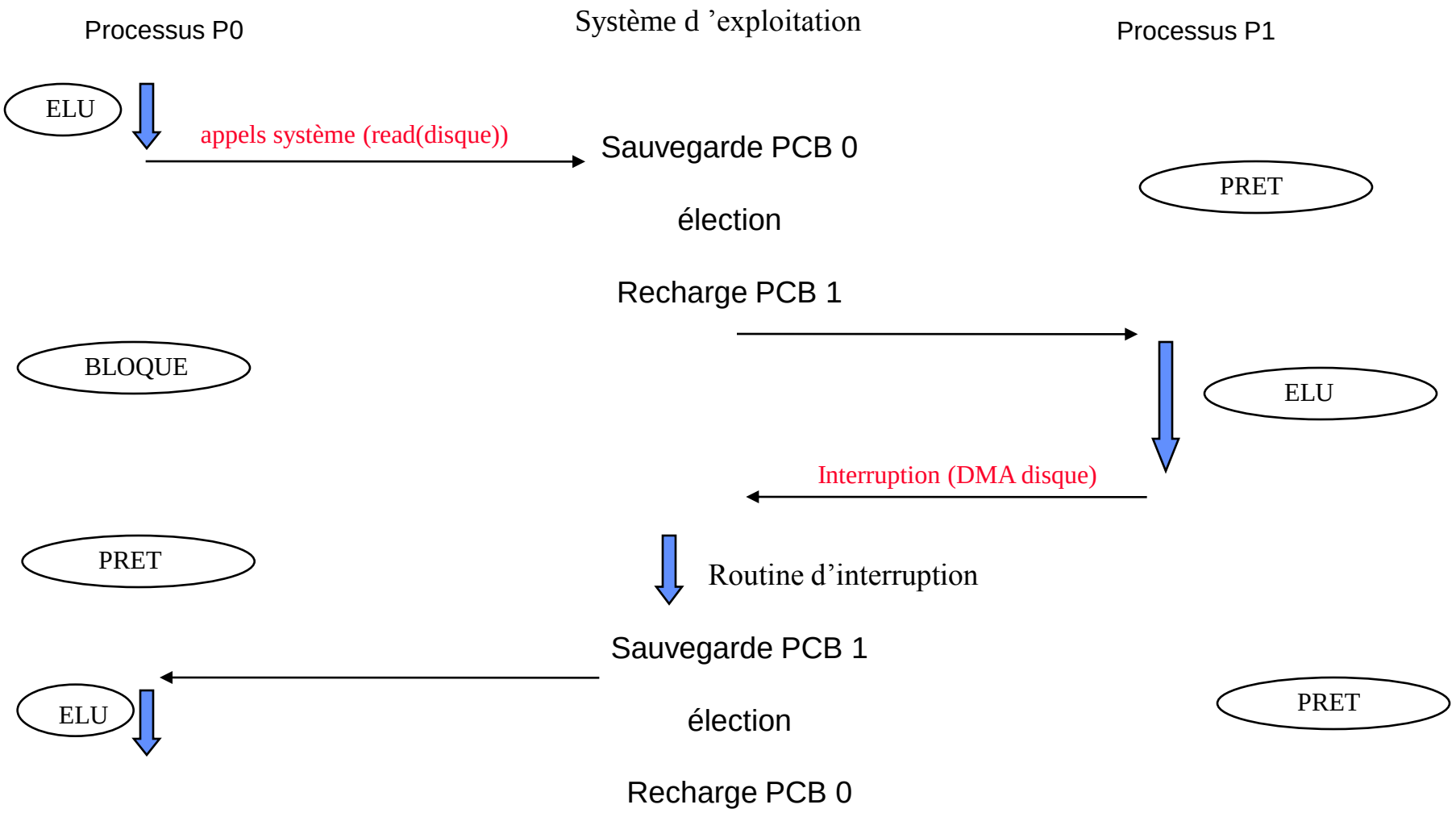
Système multiprocessus
Etats des processus
Préemption/Election



- **Election** : allocation du processeur

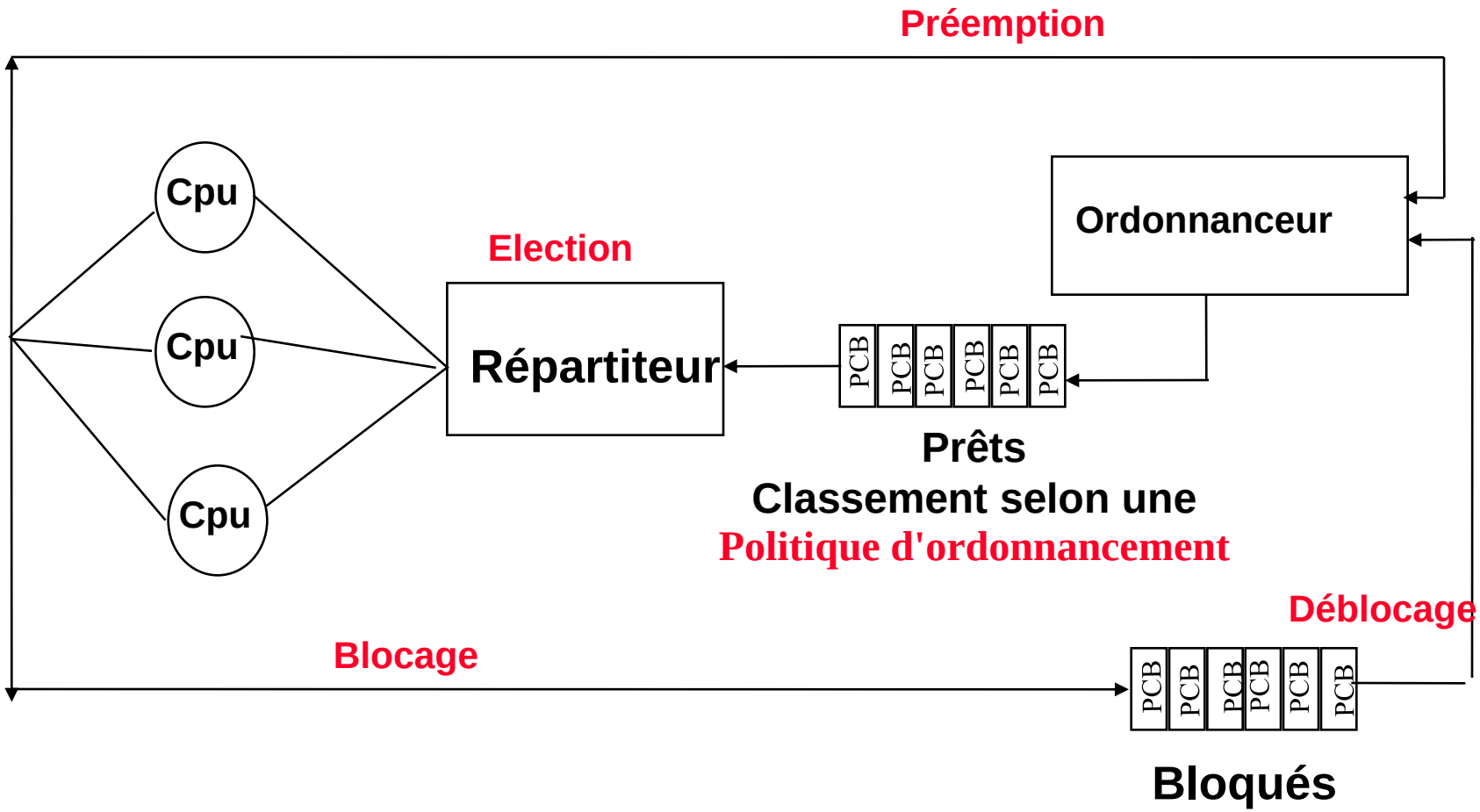
- **Préemption** : réquisition du processeur
  - ordonnancement non préemptif : un processus élu le demeure sauf s 'il se bloque de lui-même
  - ordonnancement préemptif : un processus élu peut perdre le processeur
    - s 'il se bloque de lui-même (état bloqué)
    - si le processeur est réquisitionné pour un autre processus (état prêt)

Système multiprocessus
Ordonnancement



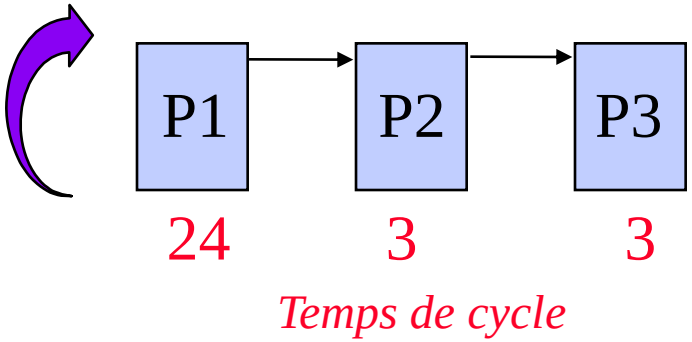
Système multiprocessus

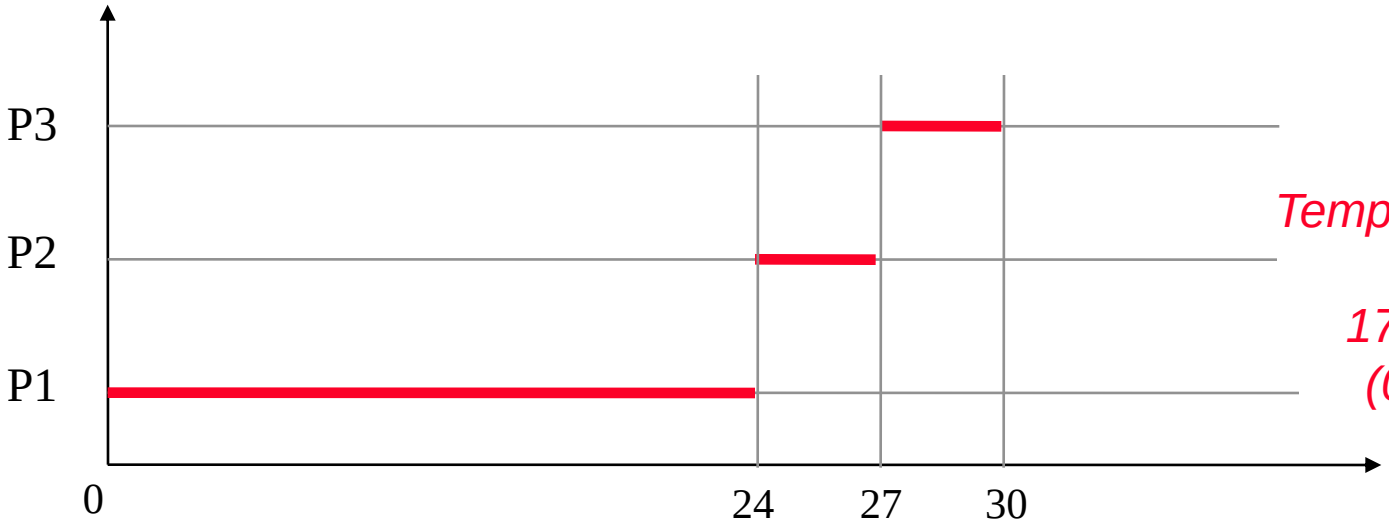
Ordonnanceur et répartiteur



Politiques d'ordonnancement

- **Premier arrivé, premier servi**
  - **FIFO, sans réquisition**
- **Par priorités constantes**
- **Par tourniquet (round robin)**

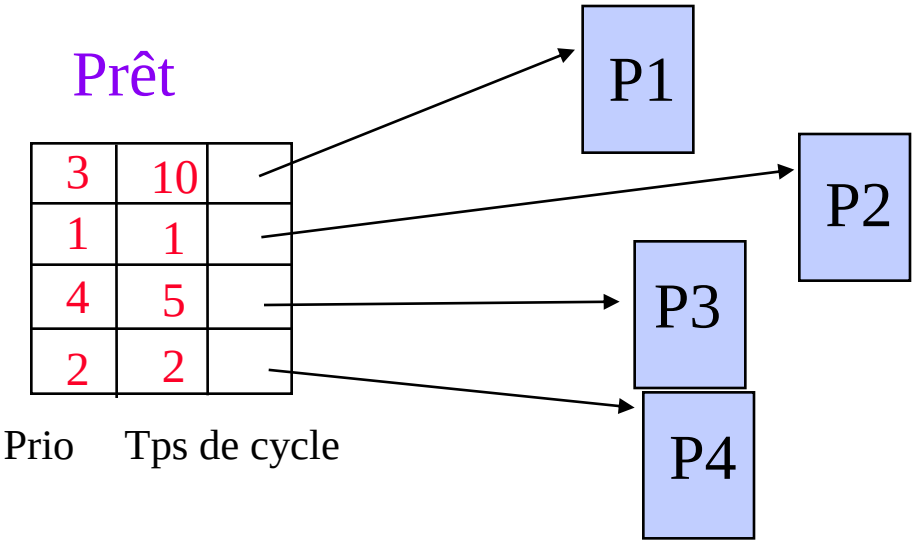




Temps moyen d'attente

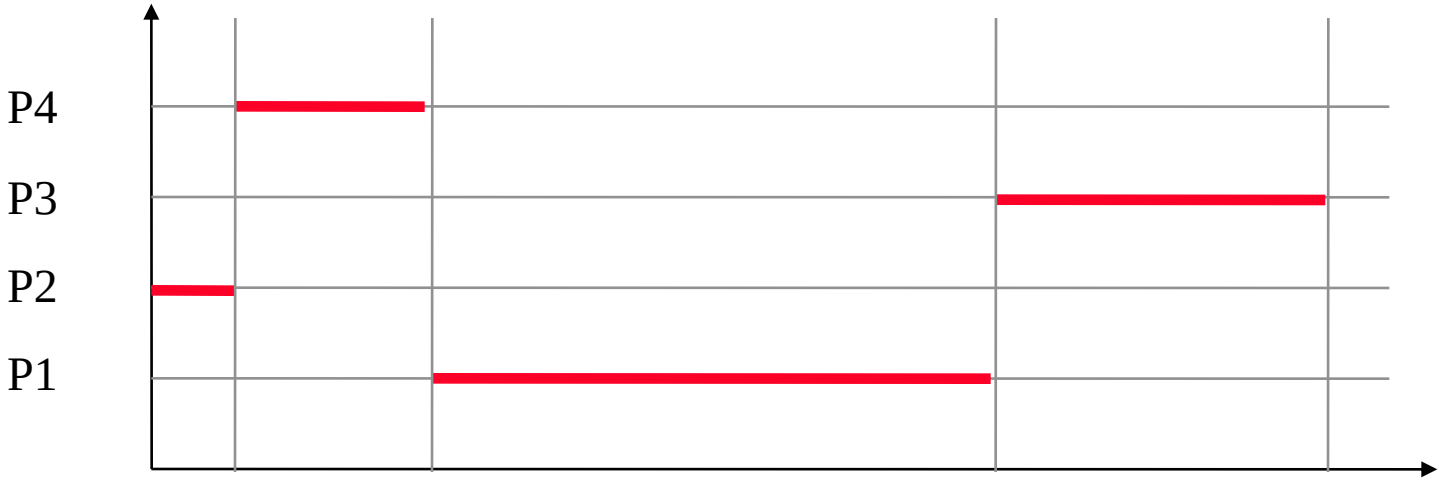
17 millisecondes
(0 + 24 + 27)/3

Politiques d'ordonnancement



- **Premier arrivé, premier servi**
- **Par priorités constantes : le processus de plus forte priorité est élu.**
- **Par tourniquet (round robin)**

↳Priorité : la plus petite valeur correspond à la plus forte priorité



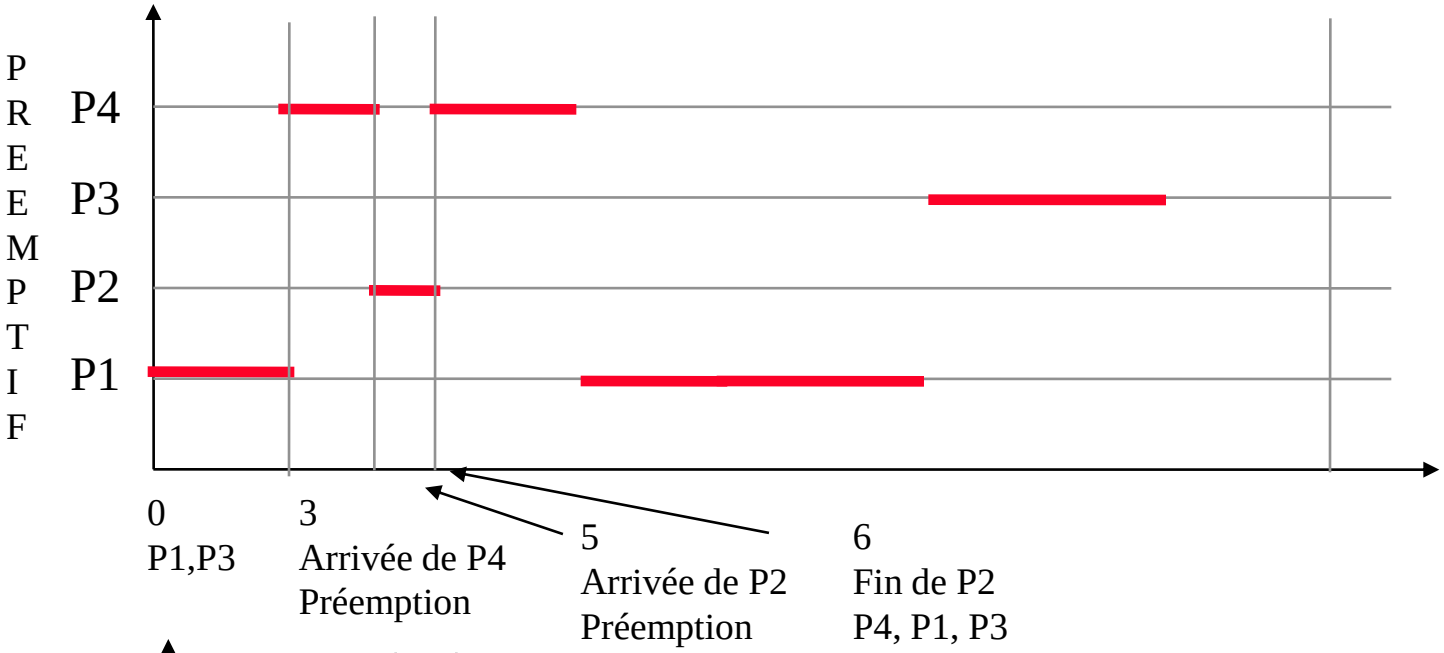
Temps moyen d'attente

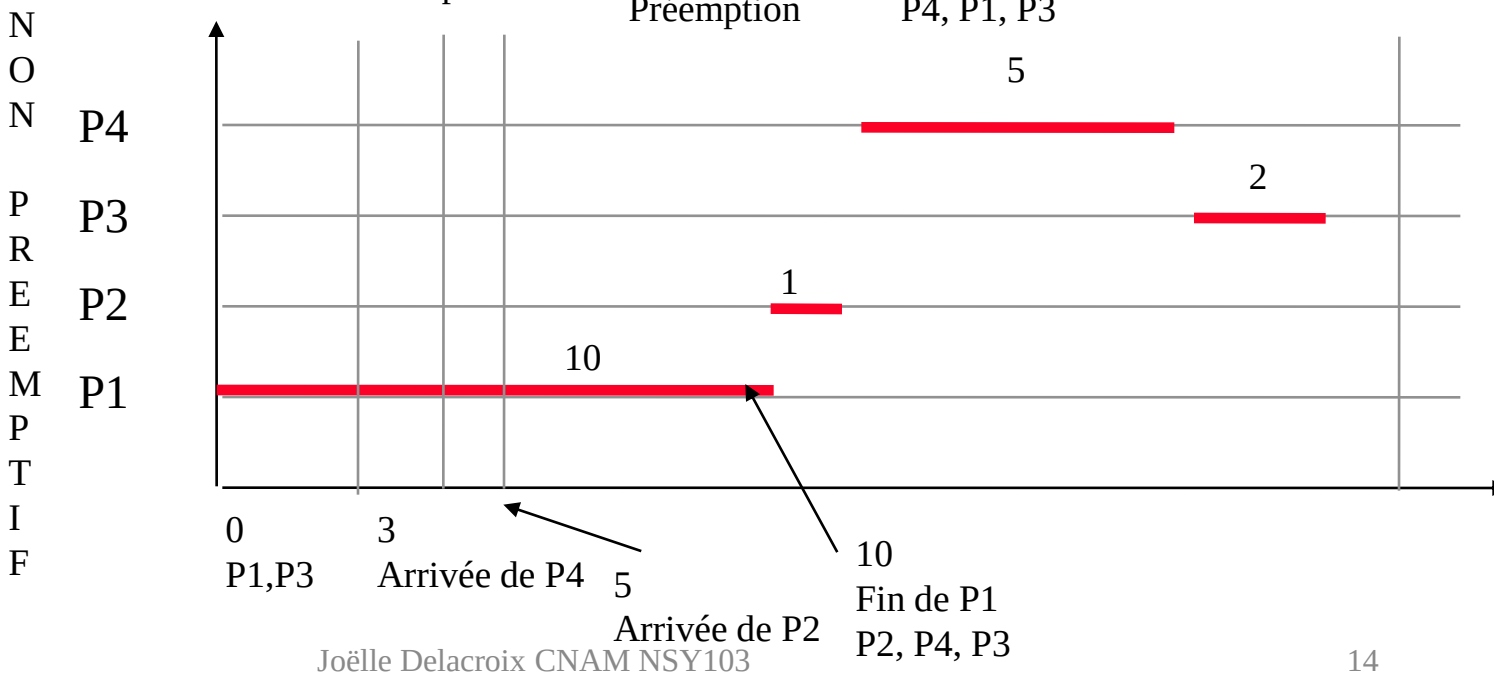
8,2 millisecondes

Politiques d'ordonnancement par priorité

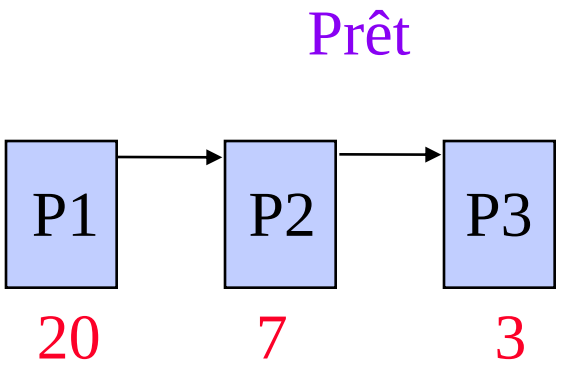
Prêt

Date de soumission	Prio	Tps de cycle	
0	3	10	P1
5	1	1	P2
0	4	2	P3
3	2	5	P4

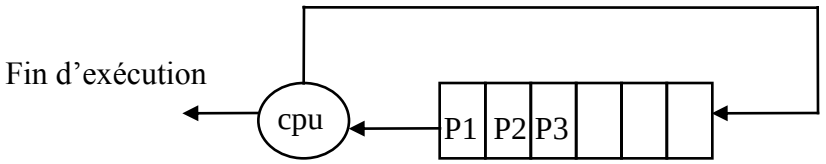


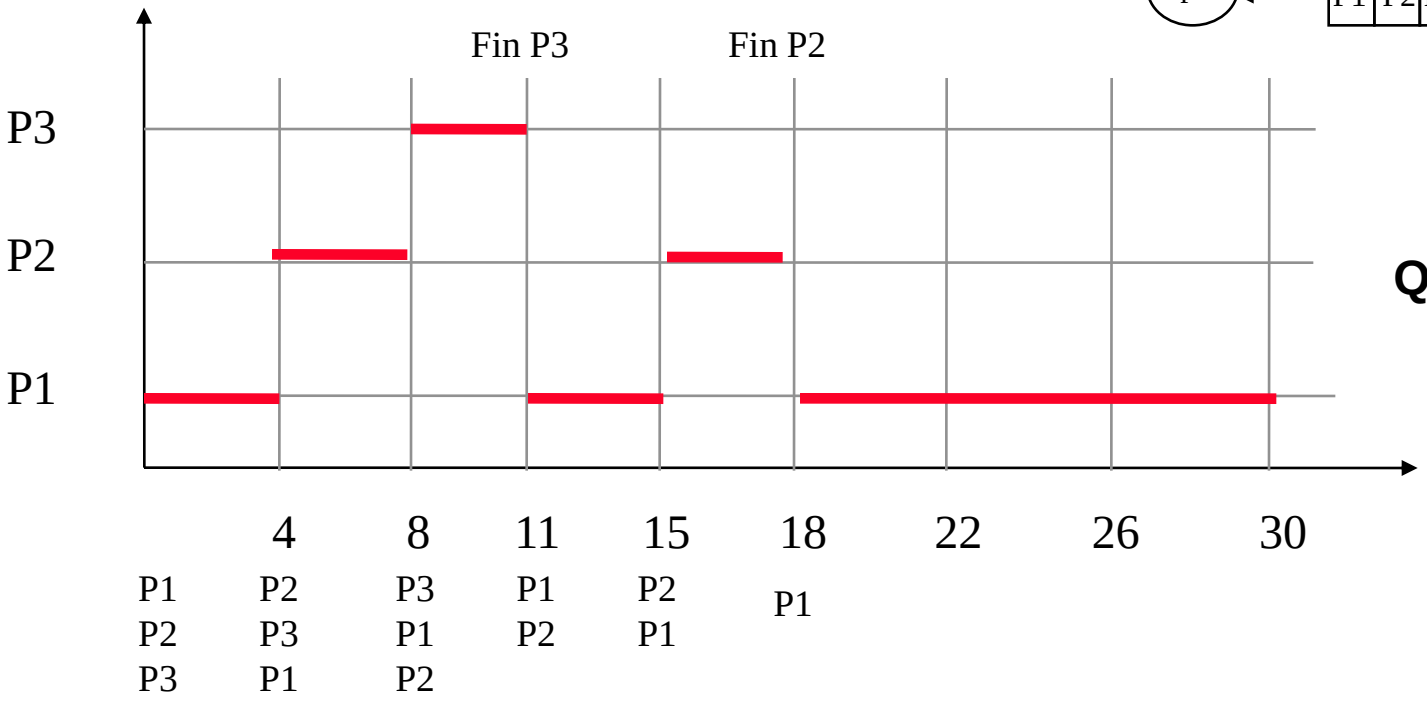


Algorithme : tourniquet



Un processus élu s'exécute au plus durant un quantum; à la fin du quantum, préemption et réinsertion en fin de file d'attente des processus prêts





Quantum = 4

Politiques d'ordonnancement

Politique d'ordonnancement	Commentaire
FIFO	Pénalise les processus de court temps d'exécution
Priorité	Risque de famine des processus de faible priorité
Tourniquet	Valeur de quantum par rapport au coût des commutations de contexte

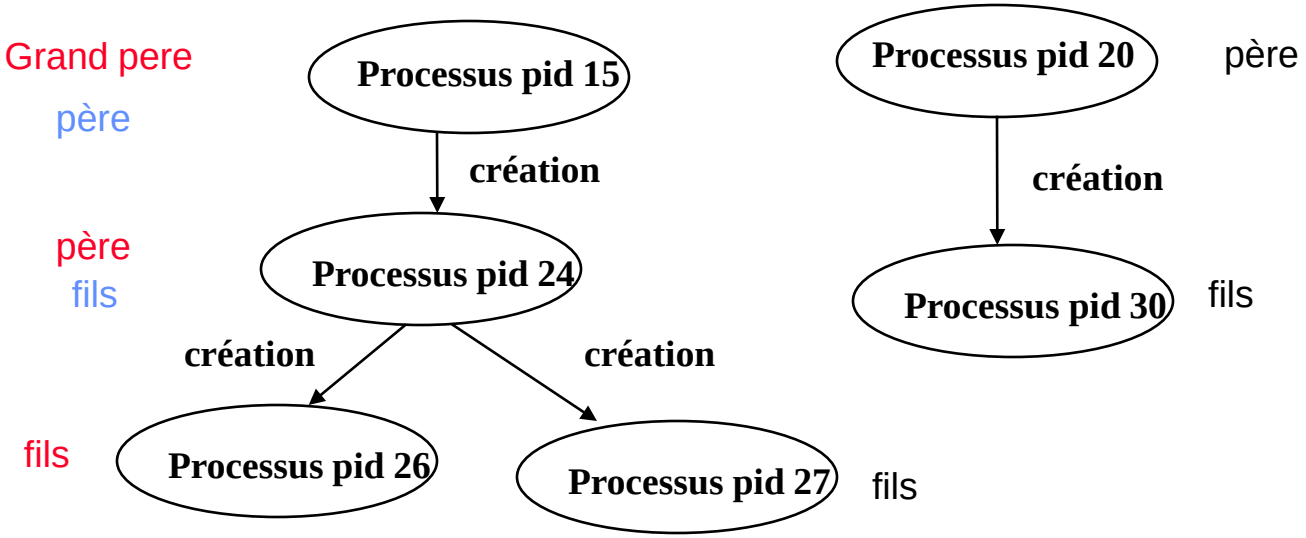
Illustration sous LINUX

Caractéristiques Générales

Un processus Unix/Linux est identifié par un numéro unique, le PID.

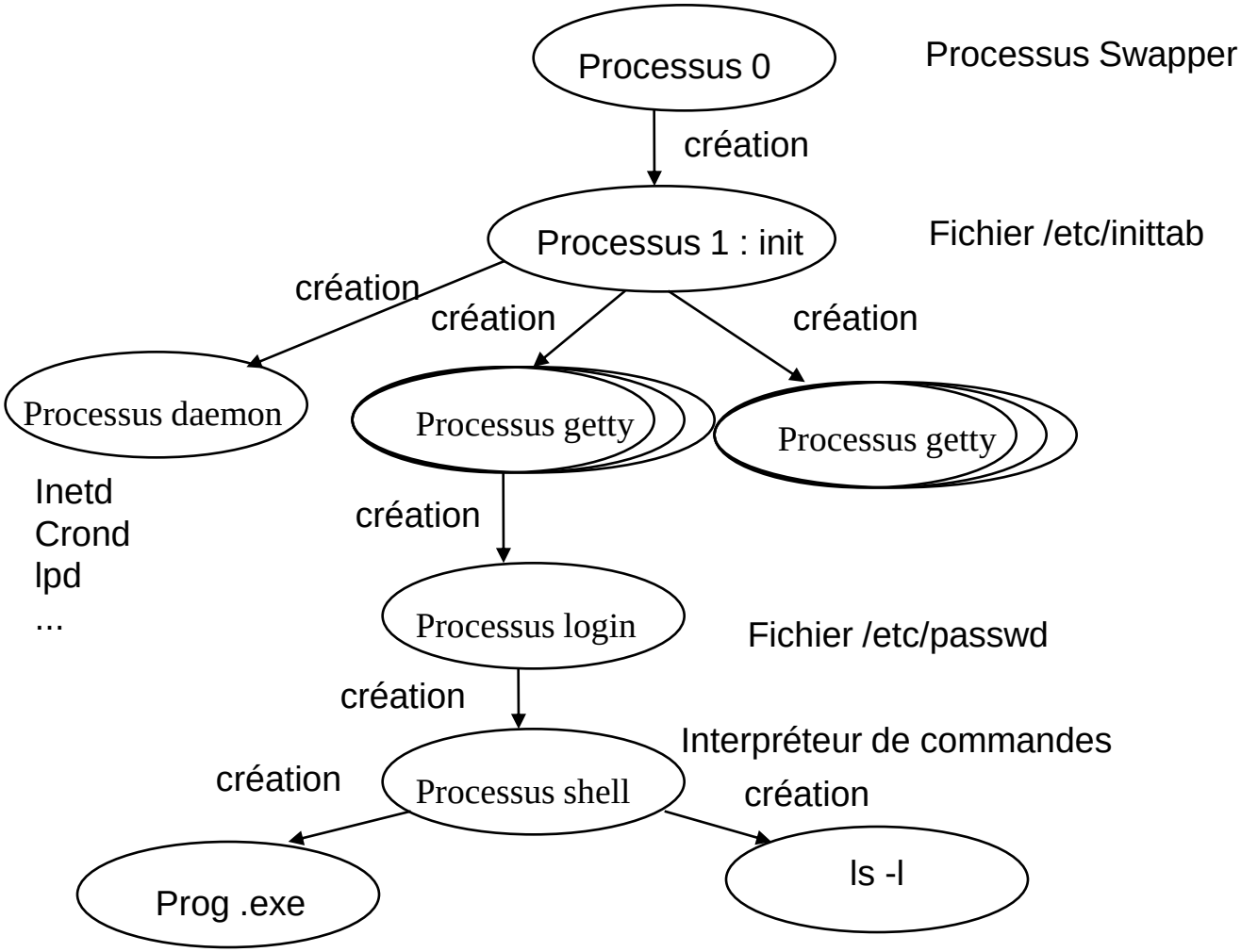
Processus Linux

- **Tout processus Linux peut créer un autre processus Linux**
  - **Arborescence de processus avec un rapport père - fils entre processus créateur et processus crée**

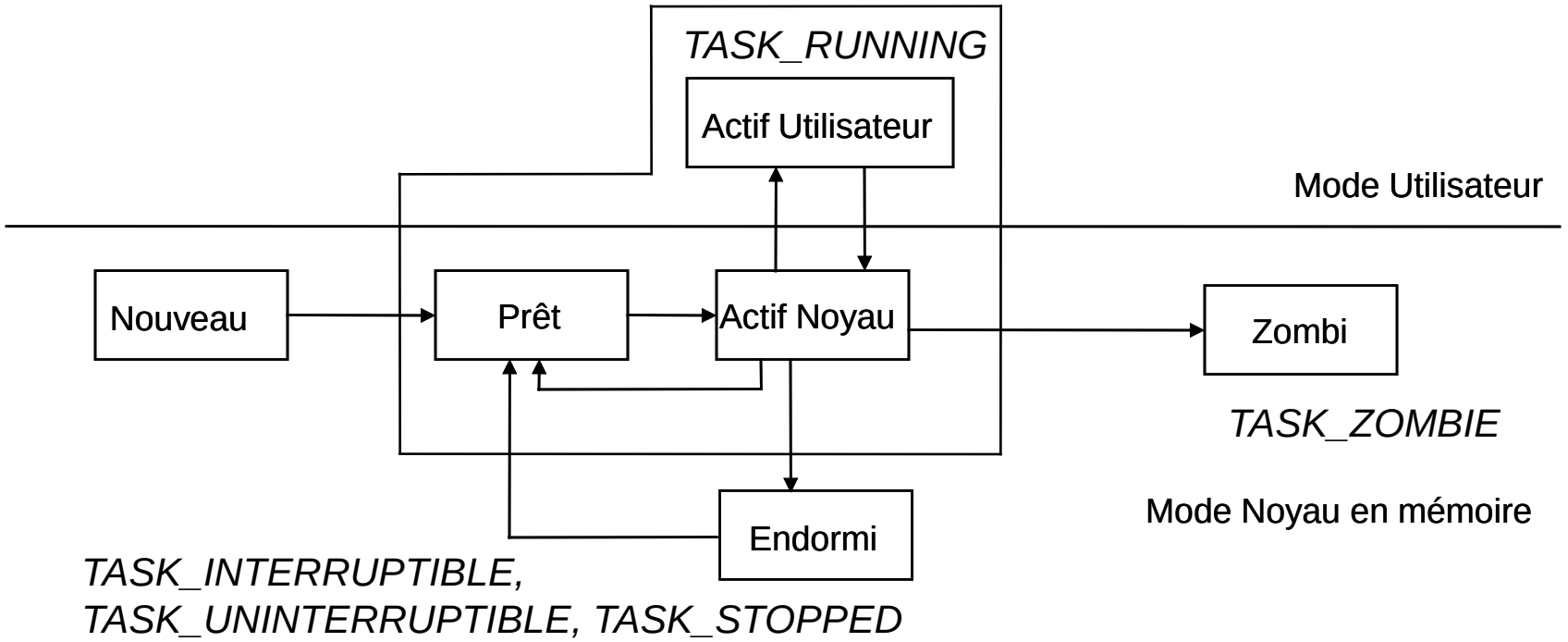


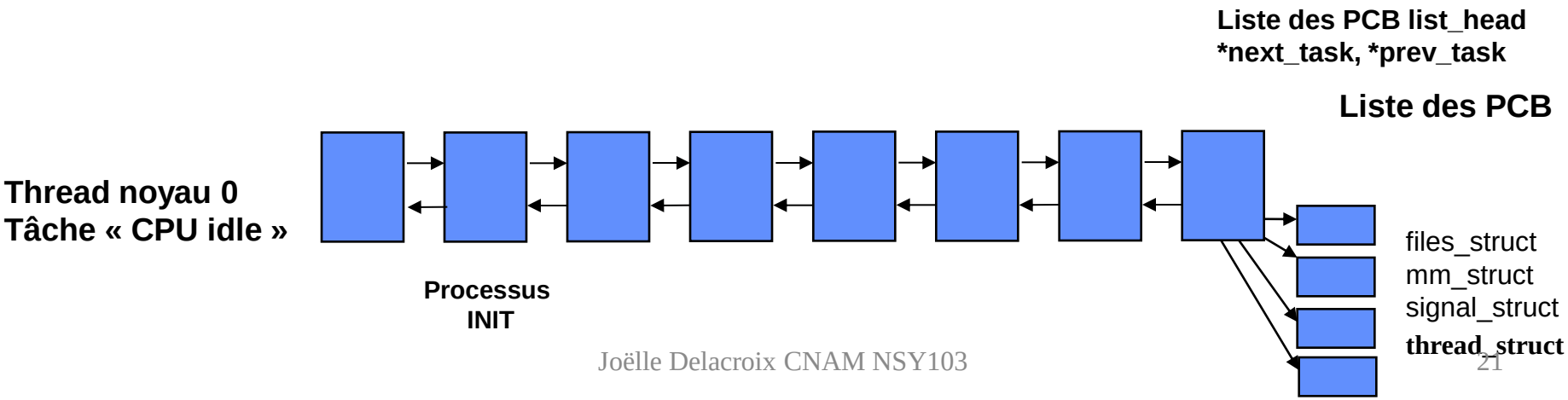
Processus Linux

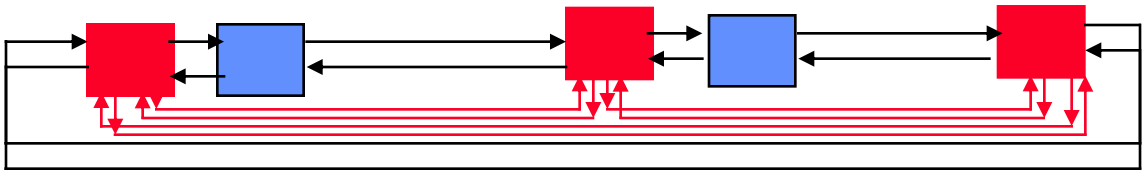
- **Tout le système Linux repose sur ce concept arborescent**



Processus Linux







Liste des PCB RUNNING (run_queue) *next_run, *prev_run;

Liste des PCB list_head *next_task, *prev_task

TASK_STRUCT

volatile long state; *- - état du processus*
long counter; *- - quantum*
long priority; *-- priorité SCHED_OTHER*
struct task_struct *next_task, *prev_task; *-- chainage PCB*
struct task_struct *next_run, *prev_run; *-- chainage PCB Prêt*
int pid; *-- pid du processus*
struct task_struct *p_opptr, *p_pptr, *p_cptr,; *-- pointeurs PCB père originel, père actuel, fils*
long need_resched; *-- ordonnancement requis ou pas*
long utime, stime, cutime, cstime;
-- temps en mode user, noyau, temps des fils en mode user, noyau
unsigned long policy; *-- politique ordonnancement SCHED_RR, SCHED-FIFO, SCHED_OTHER*
unsigned rt_priority; *-- priorité SCHED_RR et SCHED_FIFO*
struct thread_struct tss; *-- valeurs des registres du processeur*
struct mm_struct *mm; *-- contexte mémoire*
struct files_struct *files ; -- table fichiers ouverts
struct signal_struct *sig ; -- table de gestion des signaux

Système multiprocessus
Ordonnancement LINUX

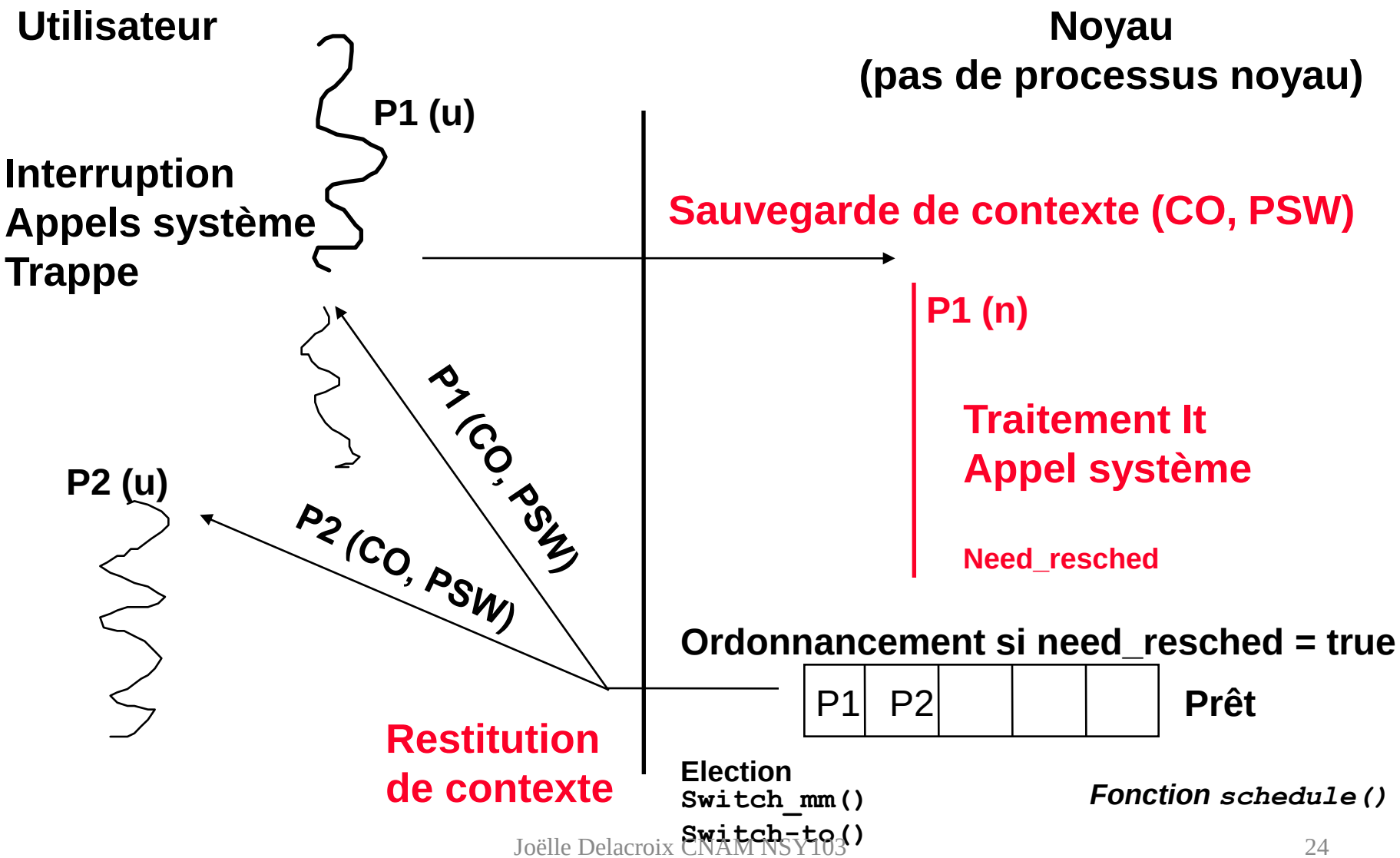
Le système Linux est un gestionnaire de processus.

Il offre des services aux processus

Il ne comporte pas à proprement parler de processus qui exécutent son code.
Ce sont les processus utilisateurs qui en passant en mode noyau exécutent le code du système

L'ordonnancement est lancé à chaque fois qu'un processus utilisateur s'apprête à repasser en mode utilisateur depuis le mode noyau et la variable noyau need_resched = true.

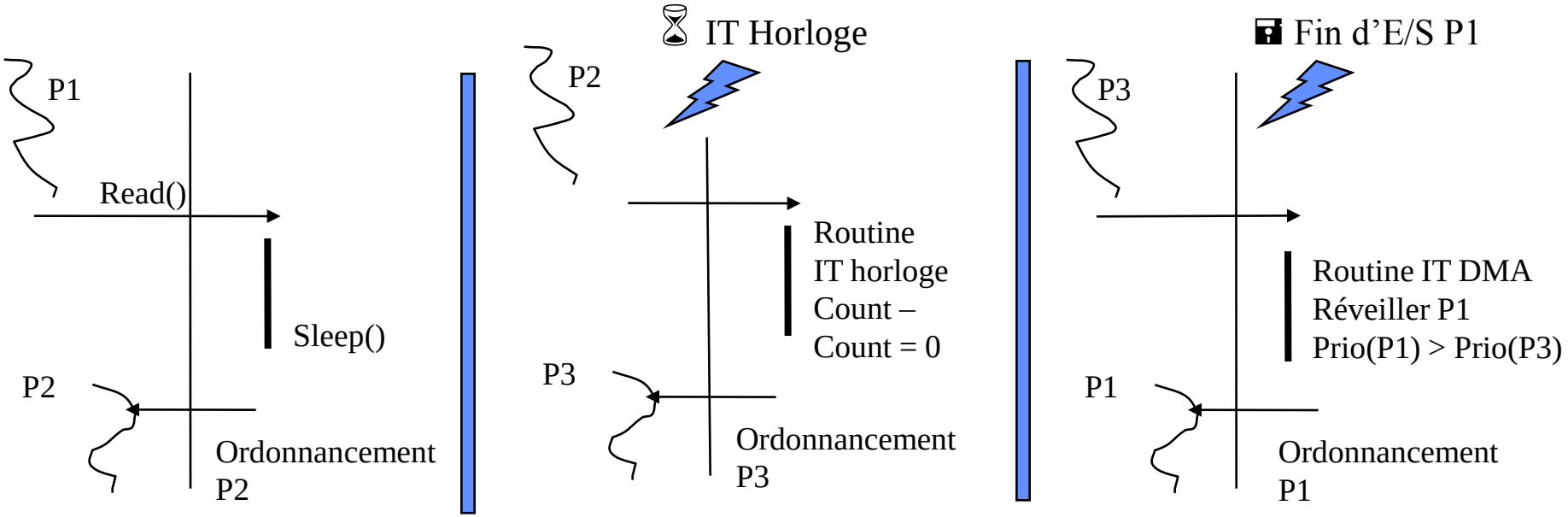
Ordonnancement dans le système LINUX



Système multiprocessus
Ordonnancement LINUX

Positionnement de need_resched :

- Un processus passe en mode bloqué ;
- Un processus a épuisé son quantum;
- Un processus temps réel plus prioritaire est réveillé.



Ordonnancement : système LINUX

- **Cinq classes d ’ordonnancement :**

Prio ++
0 à 99

- **Priorité fixe**
- **SCHED_FIFO : Fifo Priorité**
- **SCHED_RR : Tourniquet (quantum)**

Processus privilégiés

100 à 140

- **Priorité dynamique**
- **SCHED_OTHER : Temps partagé par défaut**
- **SCHED_IDLE : tâche de très faible priorité**
- **SCHED_BATCH : comme SCHED_OTHER avec pénalité**

Processus classiques

A l’instant t, le système élit (fonction GOODNESS du noyau)
-Le processus SCHED_FIFO de plus forte priorité qui s’exécute jusqu’à sa fin ou jusqu’à préemption par un processus FIFO plus prioritaire
-Le processus SCHED_RR de plus forte priorité pour un quantum
-Le processus SCHED_OTHER de plus forte priorité

Ordonnancement Linux (SCHED OTHER)

Politique utilisant à la fois les notions de quantum et de priorité.

- **Le processus a une priorité statique initiale**
- **Le processus élu est le prioritaire.**
- **Il s'exécute durant au plus un quantum de temps**

- **La priorité des processus est recalculée en fonction de leur utilisation du processeur (priorité dynamique) :**
  - **Un processus qui a eu le processeur voit sa priorité baisser**
  - **Un processus qui attend le processeur voit sa priorité augmenter.**

Ordonnancement Linux (SCHED OTHER)

Lister les processus : ps

- S : état du processus (S Stopped, R Running)
- UID : User Id, nom de l'utilisateur
- PID : Process Id, identifiant du processus
- PPID : Parent Process ID, identifiant du processus père
- PRI : priorité du processus
- TTY : terminal auquel est rattaché le processus
- TIME : durée de traitement du processus
- CMD : commande executée

```
linux-9bxb:~/jojo # ps
  PID TTY          TIME CMD
 3430 pts/1    00:00:00 bash
 3866 pts/1    00:00:00 ps
```

```
linux-9bxb:~/jojo # ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0  3430  3363  0  80   0 -  3309 wait   pts/1    00:00:00 bash
0 R     0  4015  3430  0  80   0 -  3932 -      pts/1    00:00:00 ps
```

Ordonnancement Linux (SCHED OTHER)

$> nice –gentillesse commande_longue

$> nice -5 cc monprogramme.c

- **Baisse la priorité du processus « commande_longue »**
- **Gentillesse de 0 à 20 (par défaut 10)**
- **Seul root ou un processus privilégié peut augmenter sa priorité**

```
linuxjojo@linuxjojo-VirtualBox:~$ ./exemple > fichtrace &
[1] 3099
linuxjojo@linuxjojo-VirtualBox:~$ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000  2088  2084  0  80   0 -  6758 wait   pts/14   00:00:00 bash
0 S  1000  3099  2088  0  80   0 -  1051 hrtime pts/14   00:00:00 exemple
0 R  1000  3105  2088  0  80   0 -  3562 -      pts/14   00:00:00 ps
linuxjojo@linuxjojo-VirtualBox:~$ nice ./exemple > fichtrace2 &
[2] 3106
[1]   Termine 29                 ./exemple > fichtrace
linuxjojo@linuxjojo-VirtualBox:~$ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000  2088  2084  0  80   0 -  6758 wait   pts/14   00:00:00 bash
0 S  1000  3106  2088  0  90  10 -  1051 hrtime pts/14   00:00:00 exemple
0 R  1000  3107  2088  0  80   0 -  3562 -      pts/14   00:00:00 ps
linuxjojo@linuxjojo-VirtualBox:~$
```

Adaptation multi-processeur

- **Chaque processeur dispose de sa run_queue()**
- **Un mécanisme d'équilibrage de charge permet de répartir la charge entre les processeurs (`load_balance()`)**
  - **Quand une run_queue() est vide;**
  - **Périodiquement, pour lisser les différences entre files**
- **Il est possible d'assigner des processus à des processeurs donnés et d'empêcher la migration.**