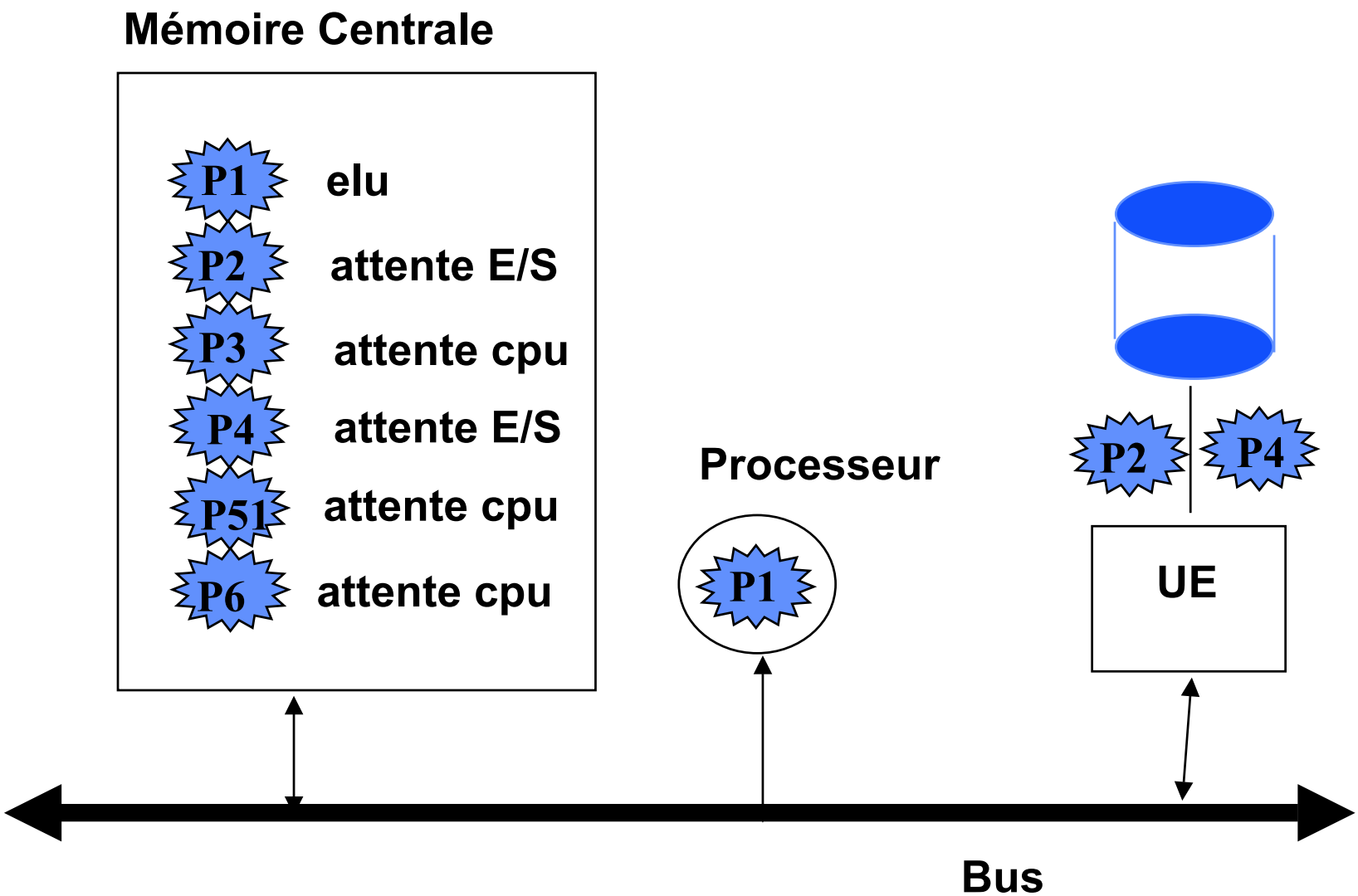
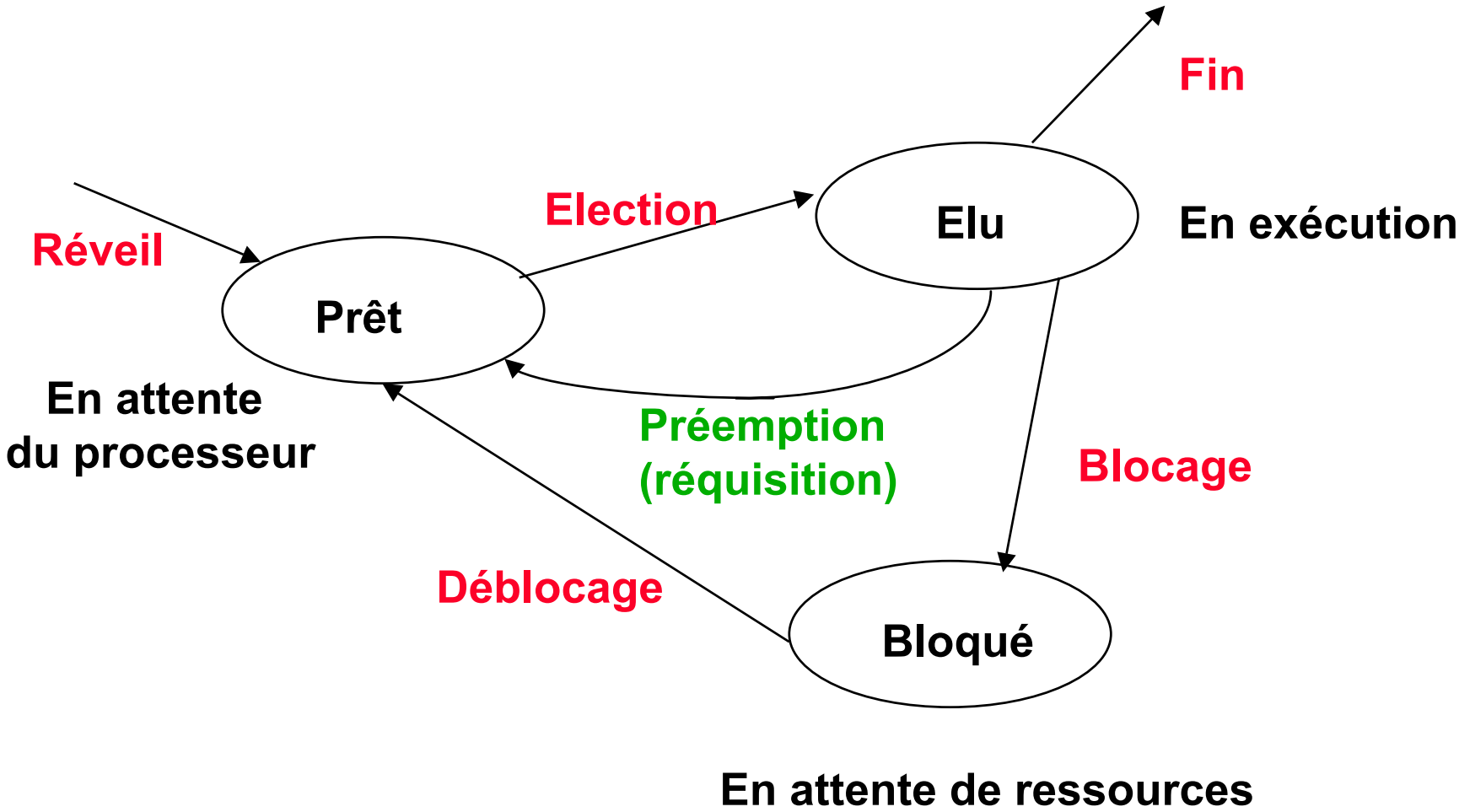
Ordonnancement

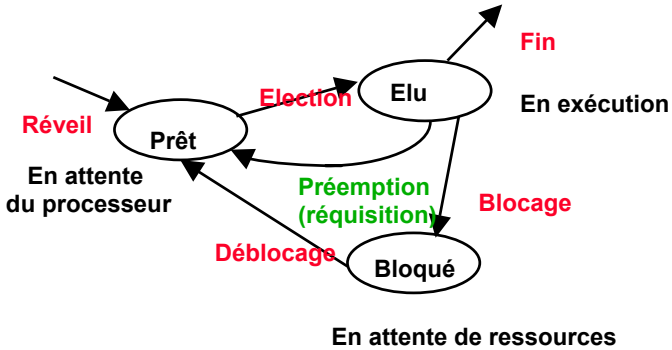
Ordonnancement dans un système multiprocessus

Système multiprocessus



Système multiprocessus
Etats des processus



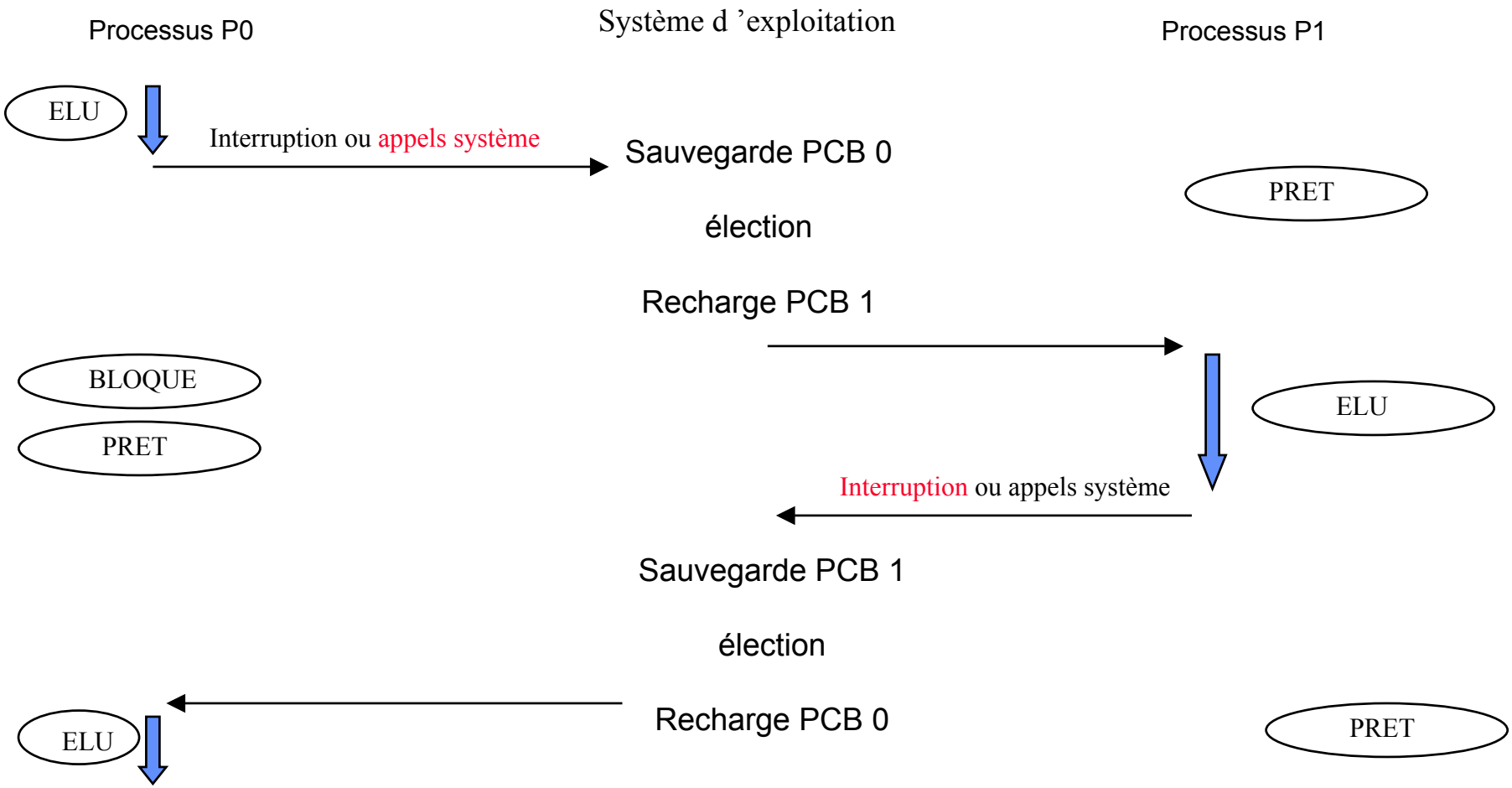


Système multiprocessus
Etats des processus

- **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
Ordonnancement

Le système Unix ou Linux est un gestionnaire de processus.

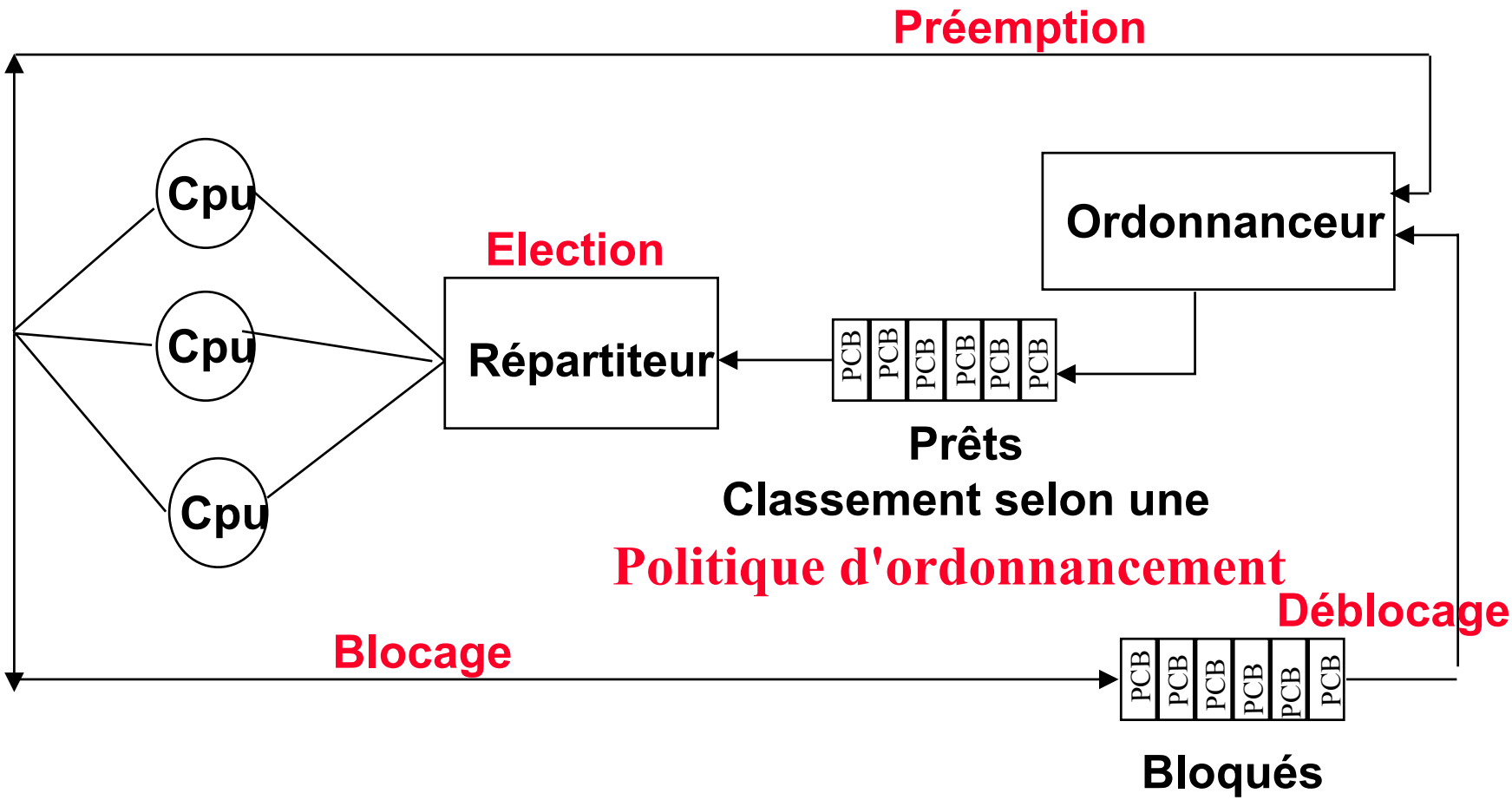
Il offre des services aux processus

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.

Système multiprocessus

Ordonnanceur et répartiteur



Politiques d'ordonnancement
Objectifs

Temps partagé (interactifs)

- **Maximiser le taux d'occupation du processeur**
- **Minimiser le temps de réponse des processus**

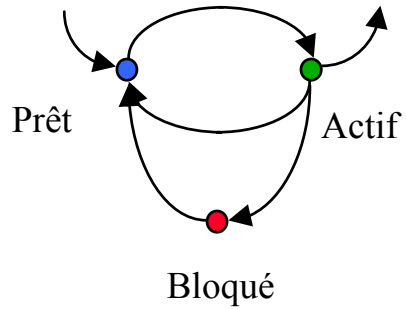
- **Temps réel**
  - **Respecter les contraintes temporelles des processus**

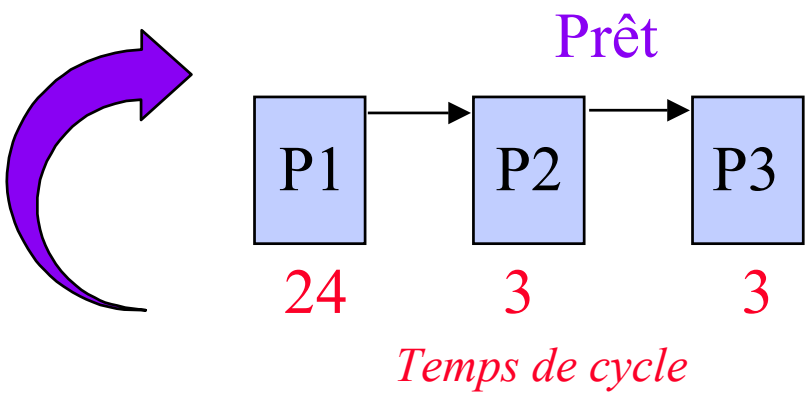
Politiques d'ordonnancement

- **Premier arrivé, premier servi**
  - **FIFO, sans réquisition**
- **Par priorités constantes**
- **Par tourniquet (round robin)**
- **Par files de priorités de priorités constantes multiniveaux avec ou sans extinction de priorité**

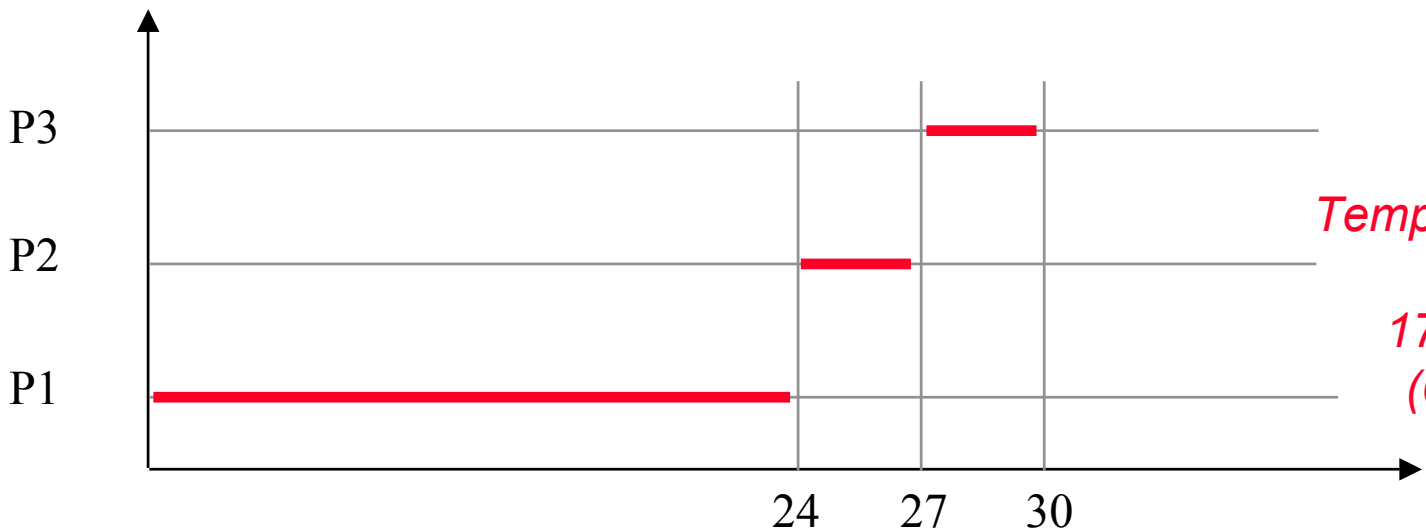
Algorithme : Premier Arrivé Premier Servi

- **FIFO, sans réquisition**





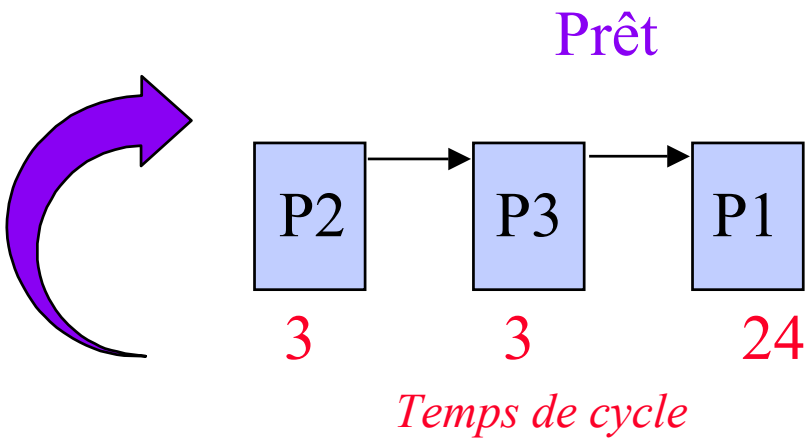
Temps de cycle

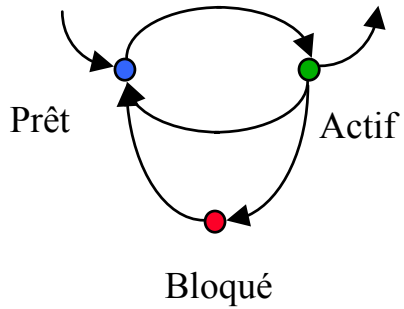


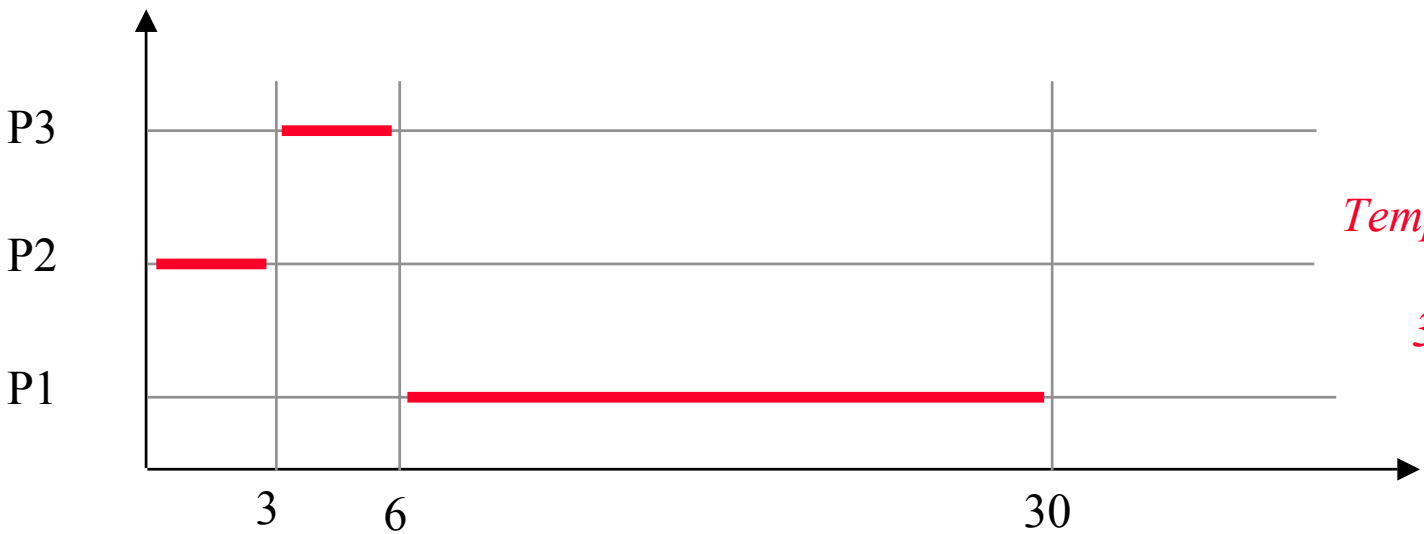
Temps moyen d'attente

17 millisecondes
(0 + 24 + 27)/3

Algorithme : Premier Arrivé Premier Servi







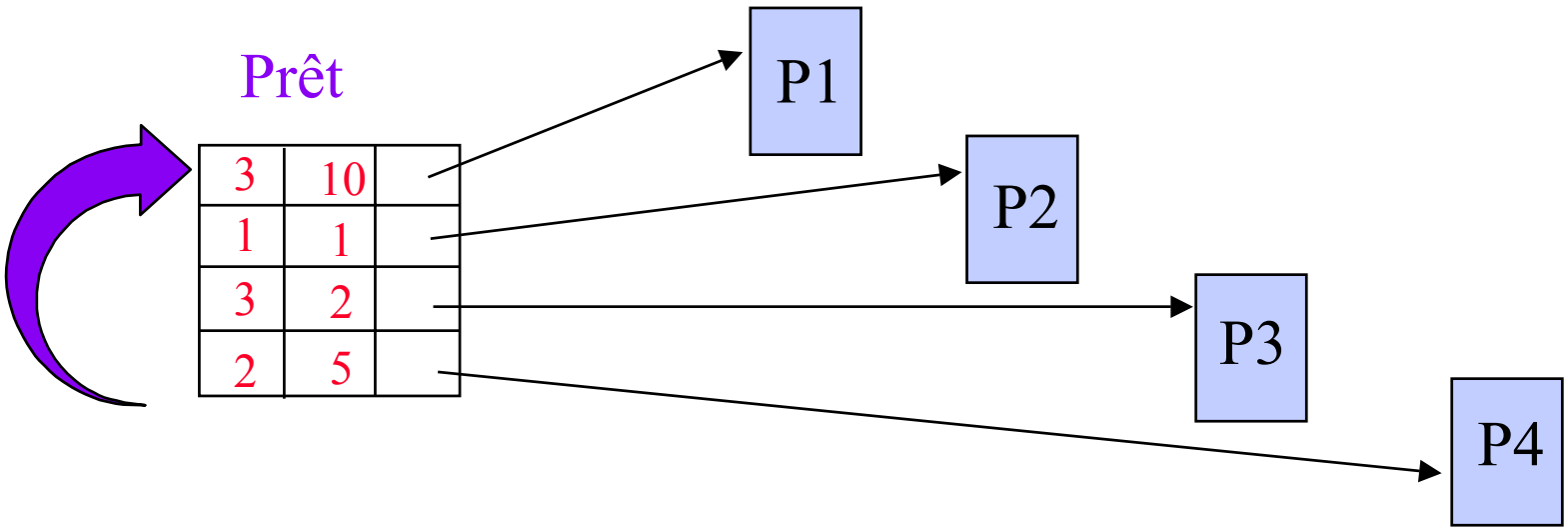
Temps moyen d'attente

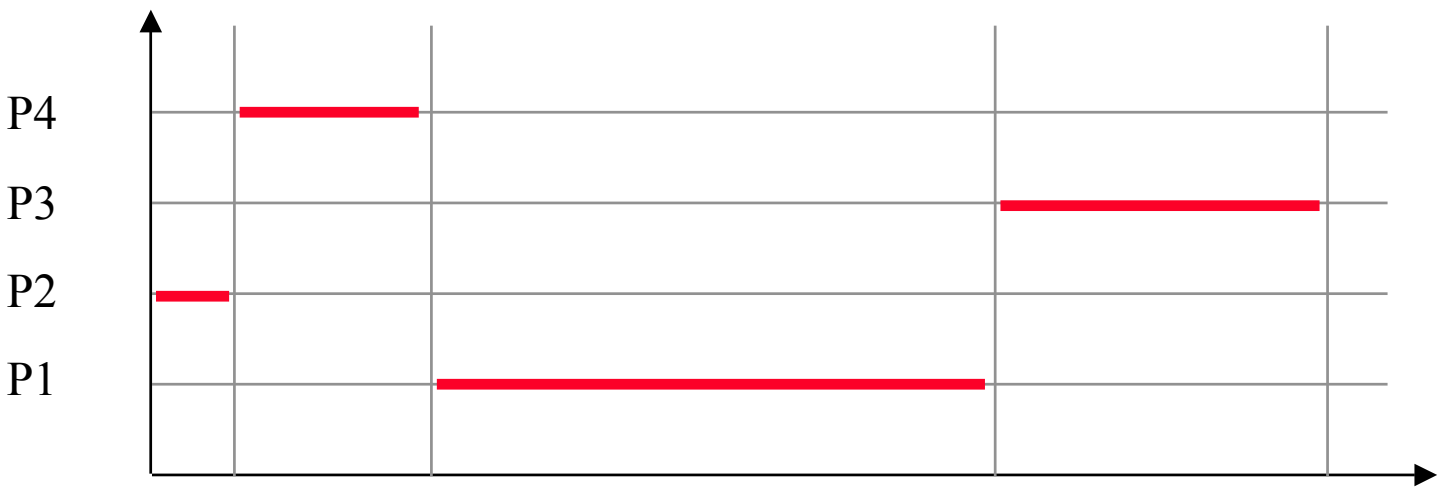
3 millisecondes
(6 + 0+ 3) /3

Politiques d'ordonnancement

- **Premier arrivé, premier servi**
- **Par priorités constantes**
  - **chaque processus reçoit une priorité**
  - **le processus de plus forte priorité est élu**
  - **Avec ou sans réquisition**
- **Par tourniquet (round robin)**
- **Par files de priorités de priorités constantes multiniveaux avec ou sans extinction de priorité**

Algorithme : avec priorités





Temps moyen d'attente

8,2 millisecondes

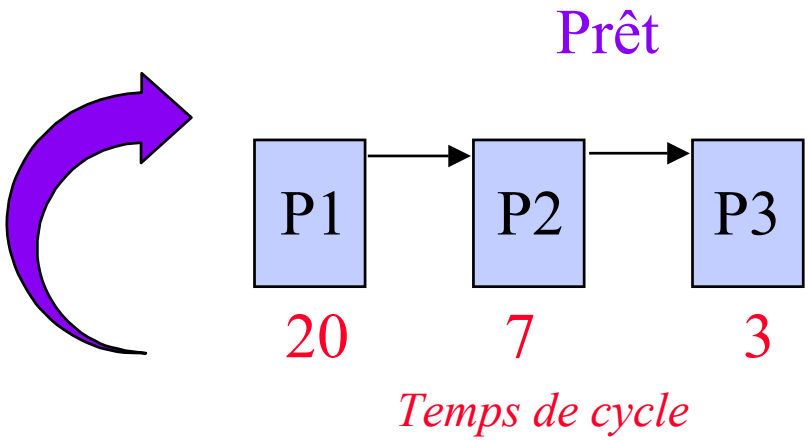
Politiques d'ordonnancement

- **Premier arrivé, premier servi**
- **Par priorités constantes**

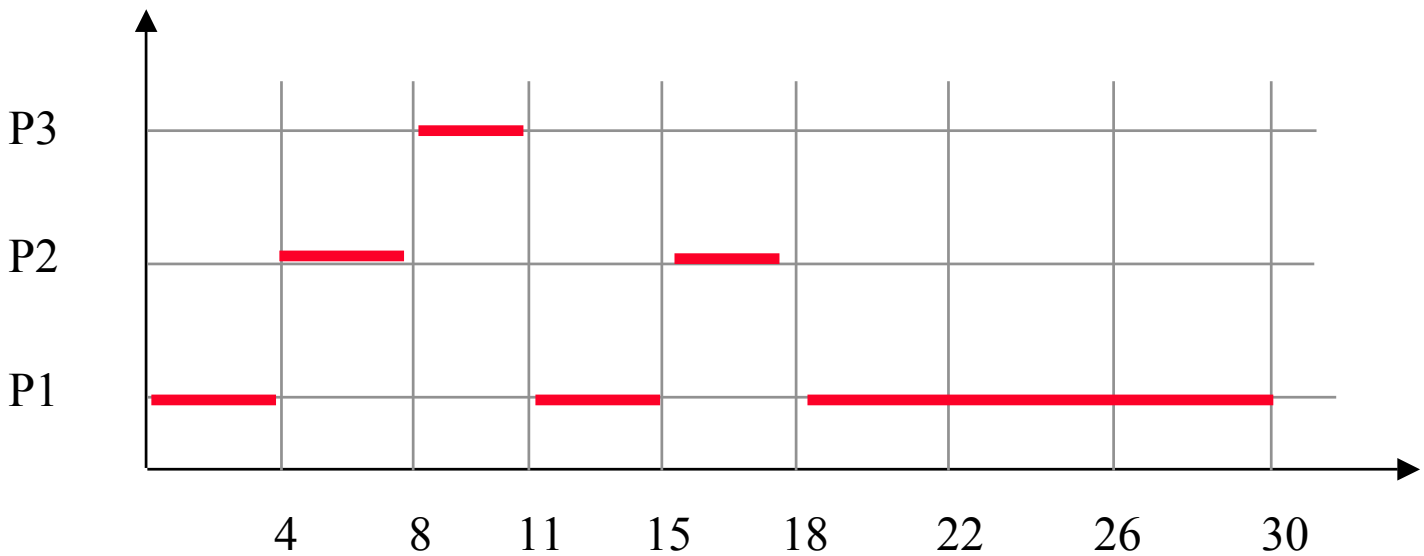
- **Par tourniquet (round robin)**
  - **Définition d'un quantum = tranche de temps**
  - **Un processus élu s'exécute au plus durant un quantum; à la fin du quantum, préemption et réinsertion en fin de file d'attente des processus prêts**

- **Par files de priorités de priorités constantes multiniveaux avec ou sans extinction de priorité**

Algorithme : tourniquet



Quantum = 4

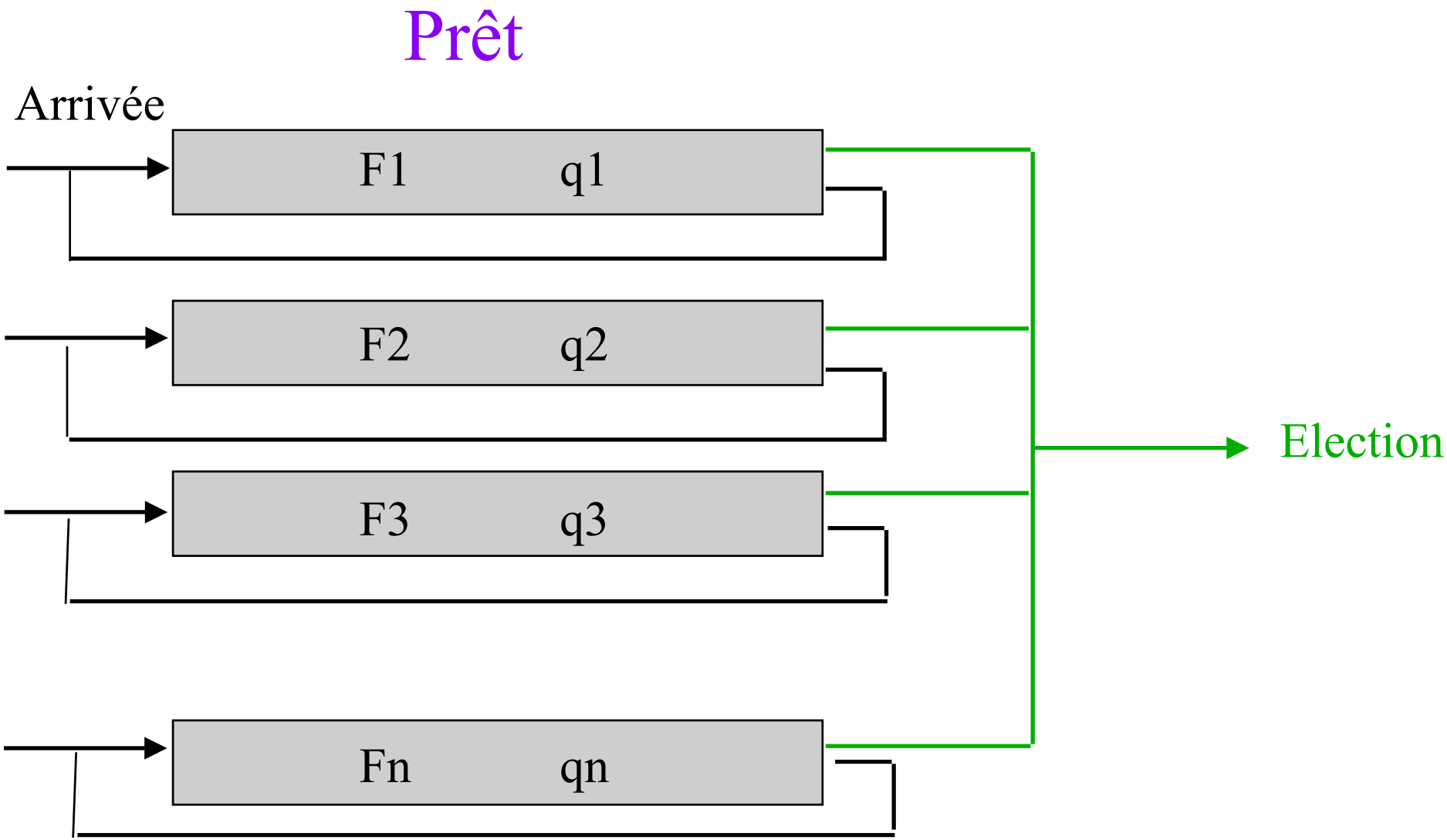


Politiques d'ordonnancement

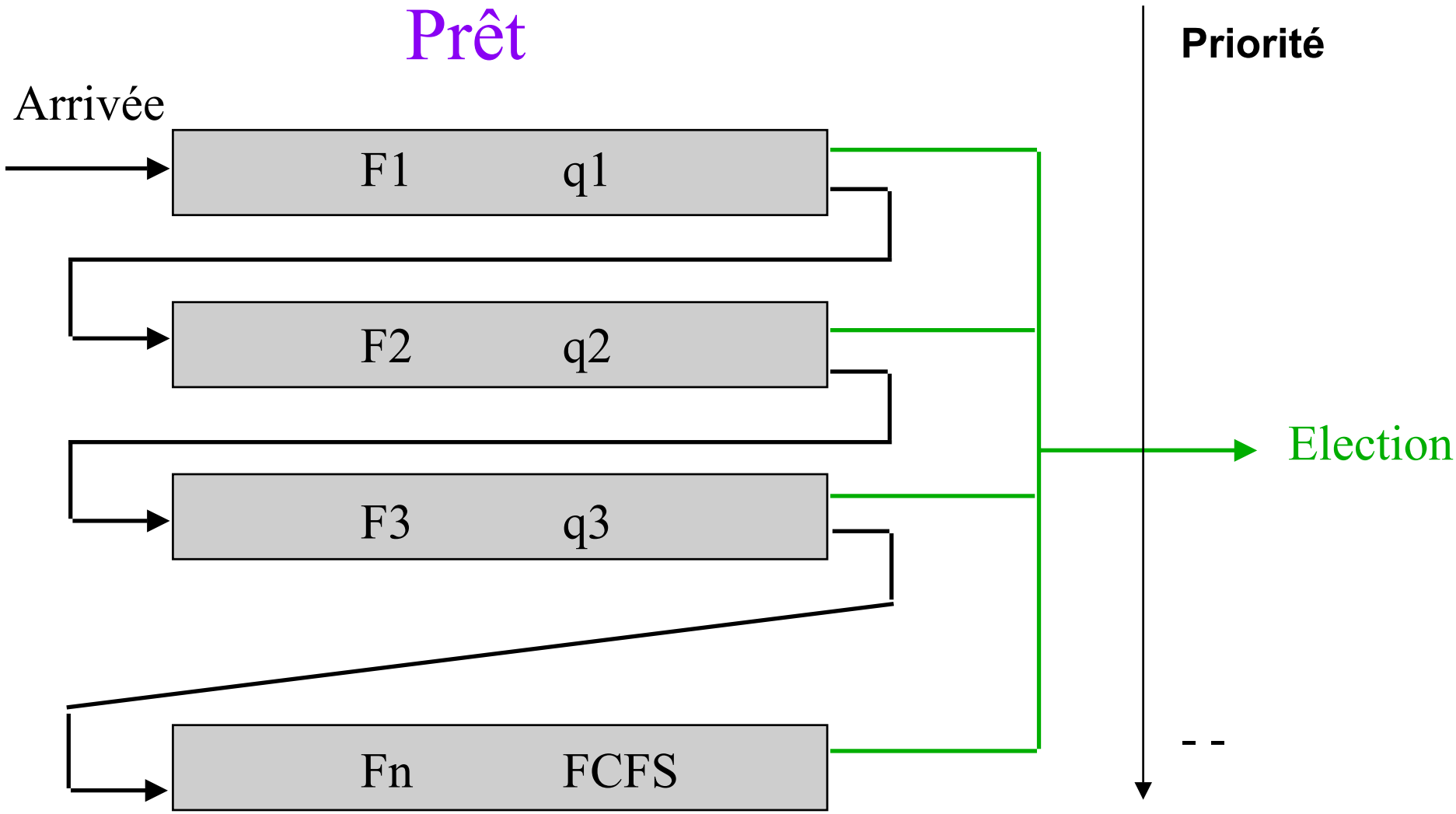
- **Premier arrivé, premier servi**
- **Par priorités constantes**
- **Par tourniquet (round robin)**

- **Par files de priorités de priorités constantes multiniveaux avec ou sans extinction de priorité**
  - **chaque file est associée à un quantum éventuellement différent**
  - **sans extinction : un processus garde toujours la même priorité**
  - **avec extinction : la priorité d'un processus décroit en fonction de son utilisation de la cpu**

Algorithme : multifiles sans extinction

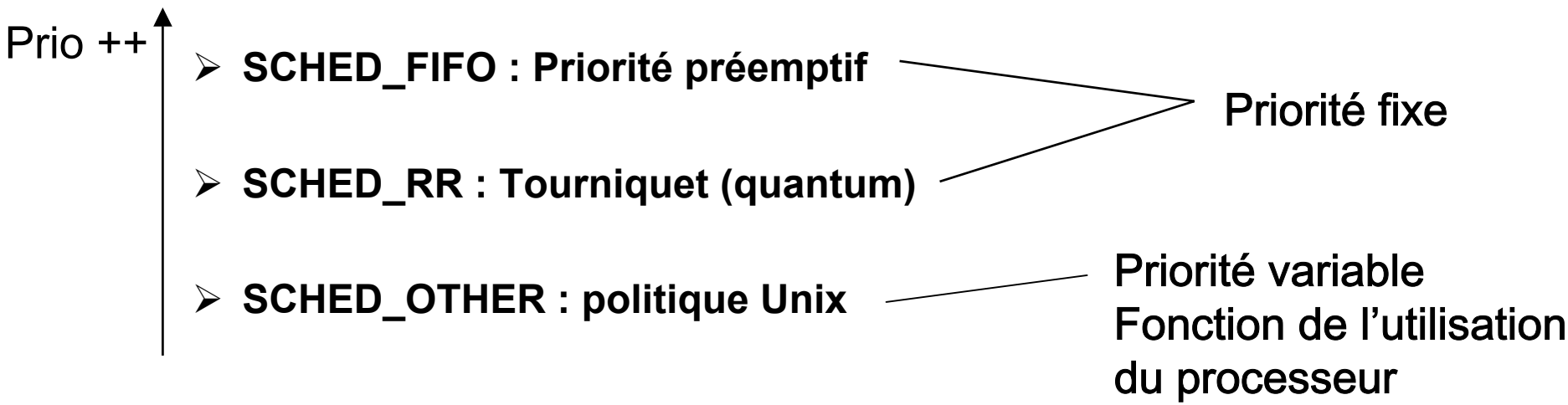


Algorithme : multifiles avec extinction



Ordonnancement : système LINUX

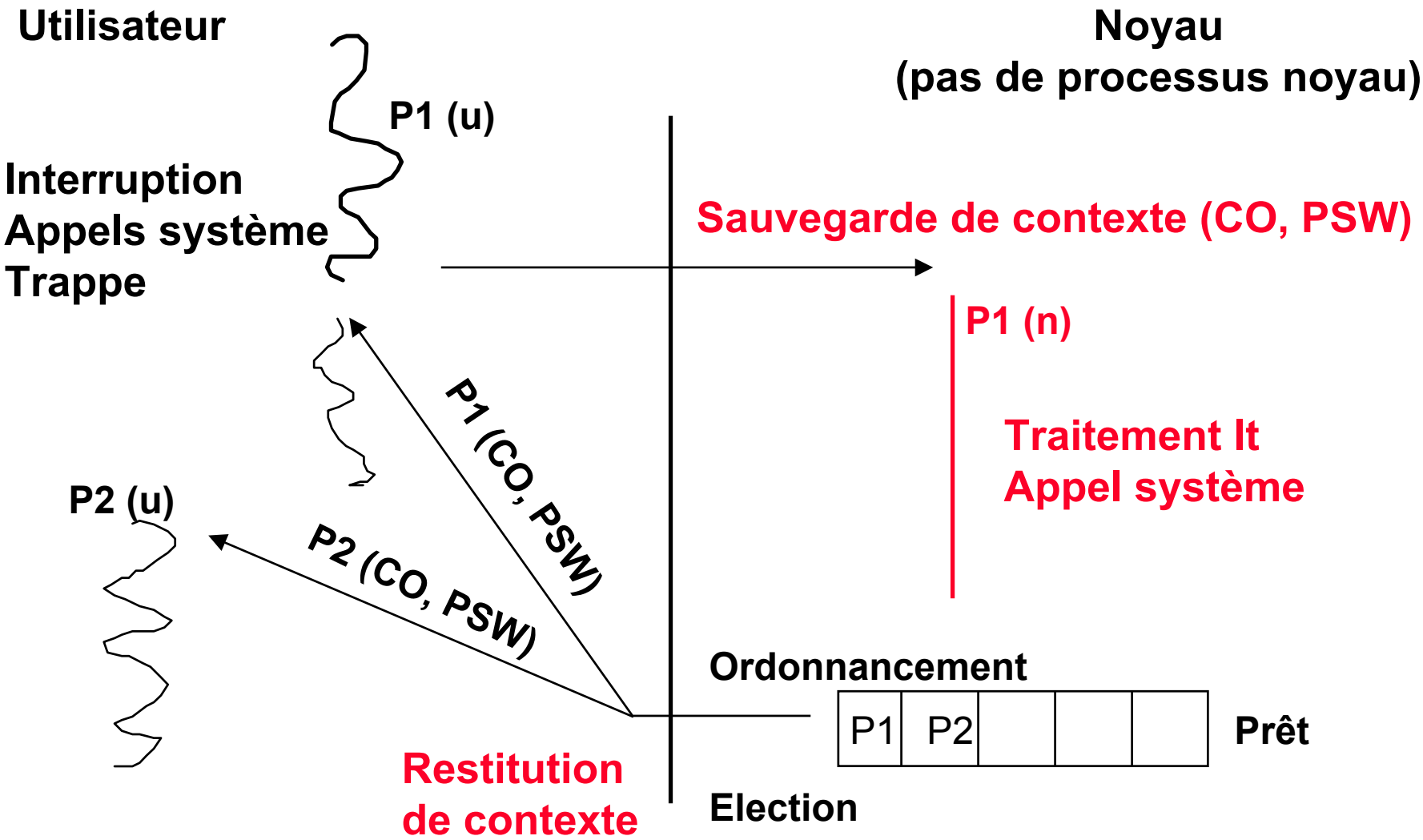
- **Trois classes d ’ordonnancement (norme POSIX) :**



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 dans le système Unix

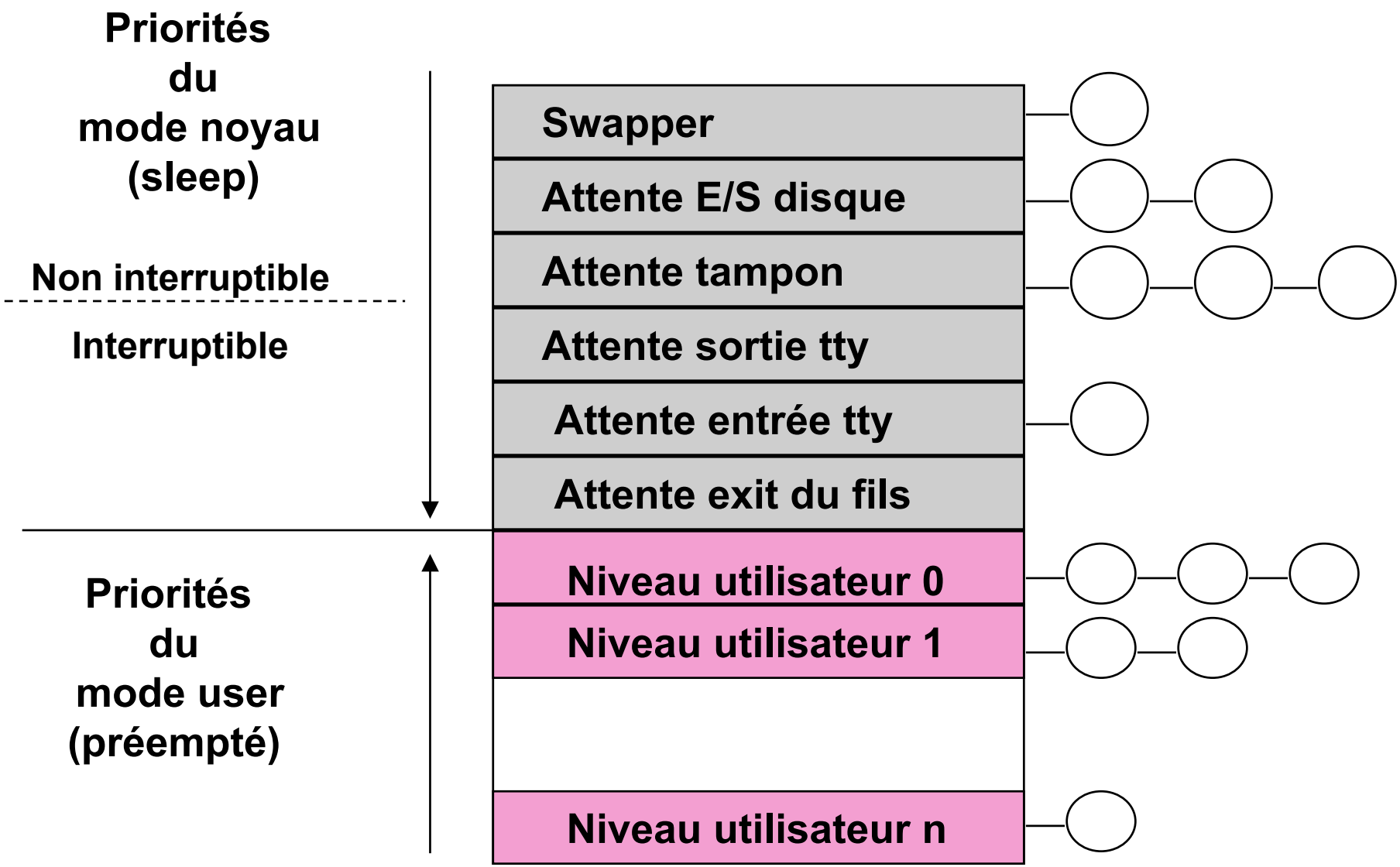
Ordonnancement dans le système Unix



Ordonnancement dans le système Unix

- **Politique en temps partagé basée sur le quantum**
- **Priorité du processus : champs dans l'entrée de la table des processus. Elle est fonction de l'utilisation de l'unité centrale**
- **Multiples files de priorité**
  - **Deux classes de priorité :**
    - **priorité utilisateur (préemption)**
    - **priorité noyau (endormis sur sleep)**

Ordonnancement dans le système Unix



Ordonnancement dans le système Unix

- **Multiples files de priorité**
  - **Priorité Noyau :**
    - **la file où le processus s'endort est fonction de l'événement attendu**
    - **la priorité correspond à une "préférence" sur les réveils suite à un événement : "éviter les embouteillages"**
    - **un processus endormi en priorité noyau demeure toujours dans la file où il s'est endormi**

Ordonnancement dans le système Unix

- **Multiples files de priorité**
  - **Priorité Utilisateur :**
    - **un processus qui se réveille quitte la priorité noyau pour réintégrer les priorités utilisateur**
    - **la procédure de traitement de l'interruption horloge ajuste les priorités des processus en mode utilisateur toutes les secondes (system V) et fait entrer le noyau dans son algorithme d'ordonnancement pour éviter qu'un processus monopolise l'unité centrale**

Ordonnancement dans le système Unix

- **Procédure de traitement de l'IT horloge et priorité des processus**
  - **A chaque IT horloge**
    ++ dans le champ "utilisation CPU" du processus élu
  - **Toutes les secondes (~~de 50 à 100 IT horloge)**

    Utilisation UC = Utilisation UC / 2

    priorité processus = Utilisation UC/2 + (priorité de base niveau utilisateur)
  - **Recalcul de la priorité; les processus se déplacent dans les files de priorité**

Ordonnancement dans le système Unix

- **Exemple**
  - **Trois processus A, B, C**
  - **Priorité initiale = 60**
  - **Priorité de niveau 0 = 60**
  - **L'It horloge se déclenche 60 fois par seconde**

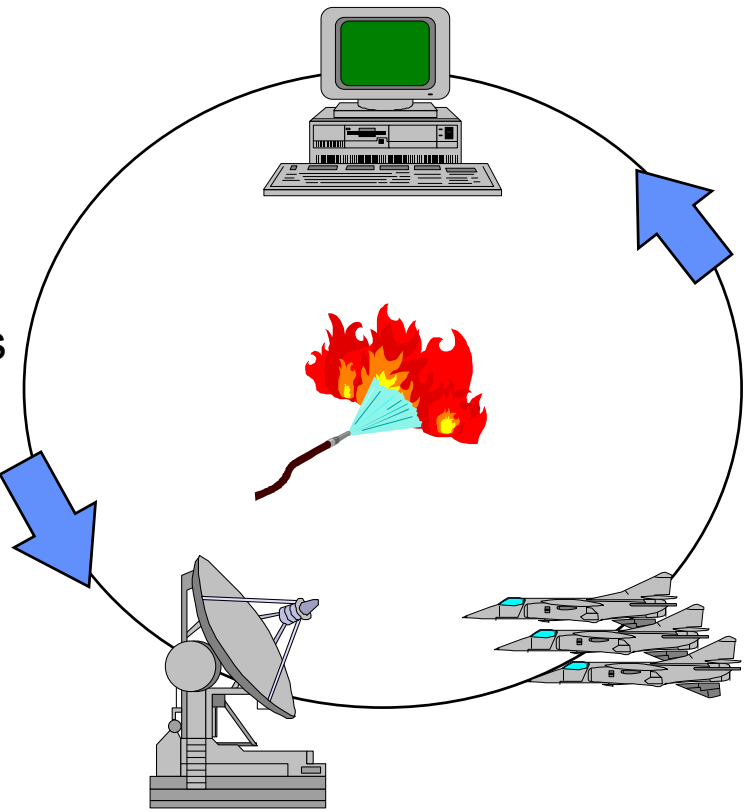
	Proc A		Proc B		Proc C	
	Priorité	Compte UC	Priorité	Compte UC	Priorité	Compte UC
0						
A	60	0 1 60	60	0	60	0
1						
B	75 (60+30/2)	30 (60/2)	60	0 1 60	60	0
2						
C	67 (60 + 15/2)	15 (30/2)	75	30	60	0 1 60
3						
A	63 (60 +7/2)	7 (15/2) 8 67	67	15	75	30
4						
B	76 (60 + 33/2)	33 (67/2)	63	7 8 67	67	15
5						

Ordonnancement dans les systèmes temps réel

Caractéristiques de l'ordonnancement temps réel

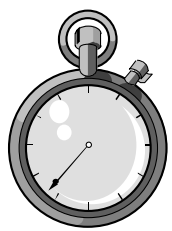
- **Contexte applicatif**

Application de contrôle multitâches



Commandes
Fonction de pilotage

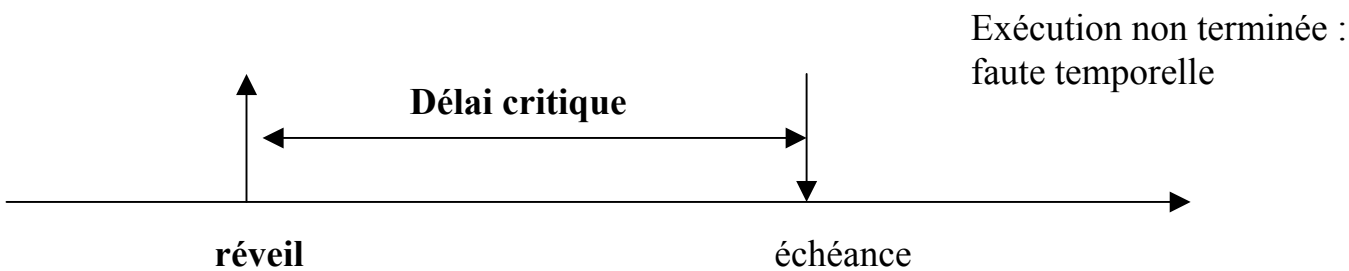
Mesures
Evénements
Fonction de suivi



Contraintes de temps

Caractéristiques de l'ordonnancement temps réel

- **Caractéristiques de l'ordonnancement temps réel**
  - But principal de l'ordonnancement : **permettre le respect des contraintes temporelles associées à l'application et aux tâches.**
  - **Chaque tâche possède un** délai critique **: temps maximal pour s'exécuter depuis sa date de réveil. La date butoir résultante est appelée** échéance**.**
  - **Le dépassement d'une échéance est appelé** faute temporelle**.**



Caractéristiques de l'ordonnancement temps réel

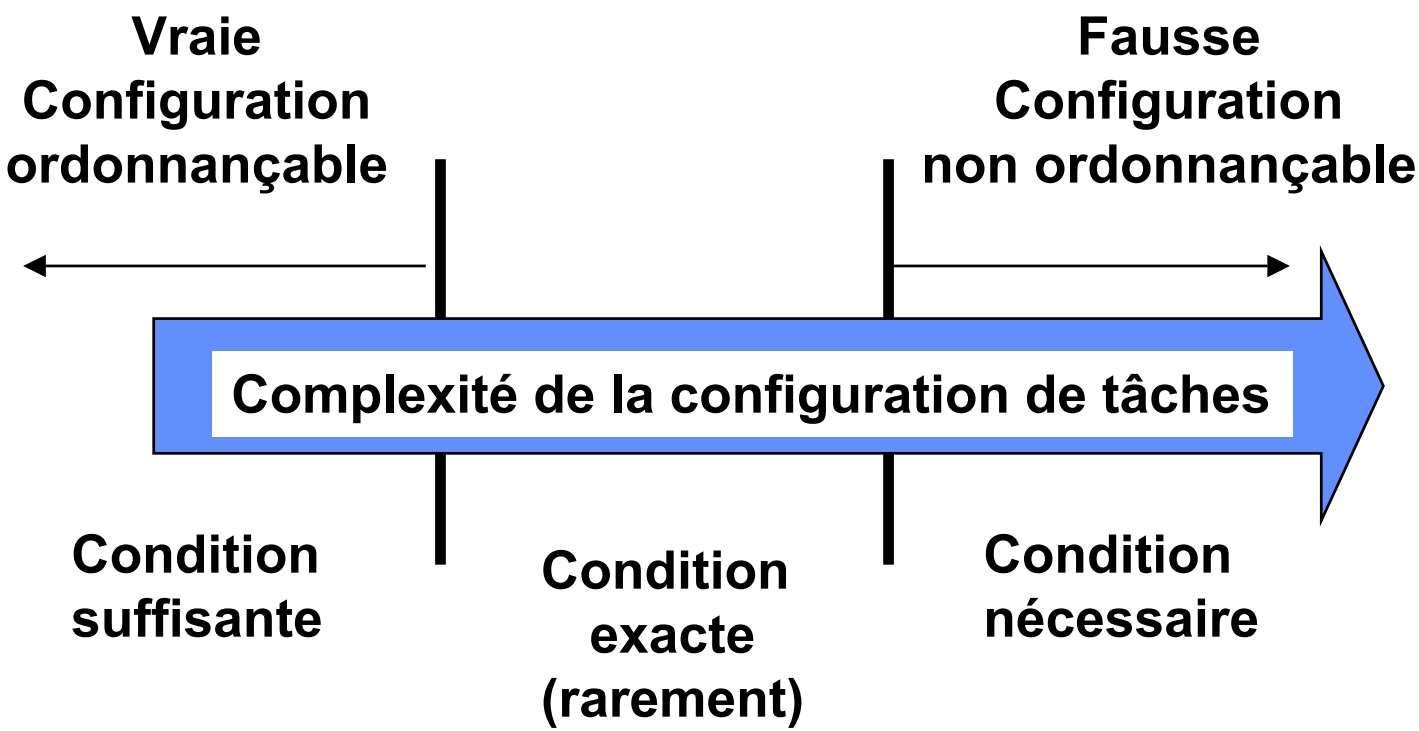
- **Caractéristiques de l'ordonnancement temps réel**

  - Applications embarquées et critiques : **nécessité de certifier l'ordonnancement réalisé, c'est-à-dire de vérifier avant le lancement de l'application (hors ligne) le respect des contraintes temporelles.**

  - **Cette certification s'effectue à l'aide de** tests d'acceptabilité **qui prennent en compte les paramètres temporels des tâches (temps d'exécutions des tâches) .**

Caractéristiques de l'ordonnancement temps réel

- **Test d'acceptabilité**



Caractéristiques de l'ordonnancement temps réel

- **Caractéristiques de l'ordonnancement temps réel**

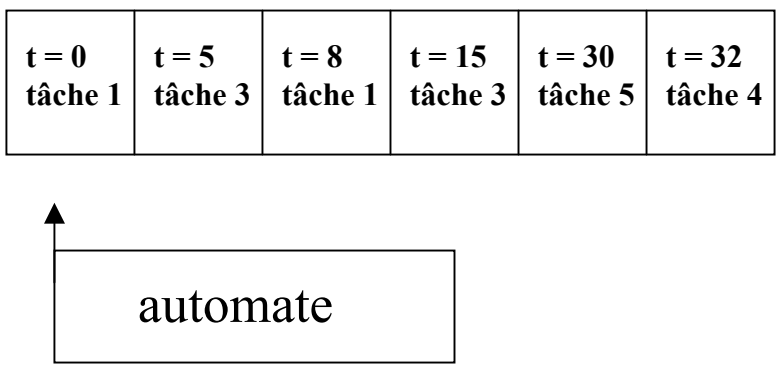
  - **Tests d'acceptabilité : utilisent les temps d'exécution des tâches.**
    - ☞**Il faut pouvoir déterminer et borner ces temps**
    - ☞**L'exécutif doit être** déterministe

  - Un exécutif déterministe **est un exécutif pour lequel les temps de certaines opérations système et matérielles élémentaires peuvent être bornés : temps de commutation, temps de prise en compte des interruptions, etc…**

- **Caractéristiques de l'ordonnancement temps réel**

  - **Ordonnancement hors ligne**

  Un ordonnancement hors ligne établit avant le lancement de l'application une séquence fixe d'exécution des tâches à partir de tous les paramètres de celles-ci. Cette séquence est rangée dans une table et exécutée en ligne par un automate



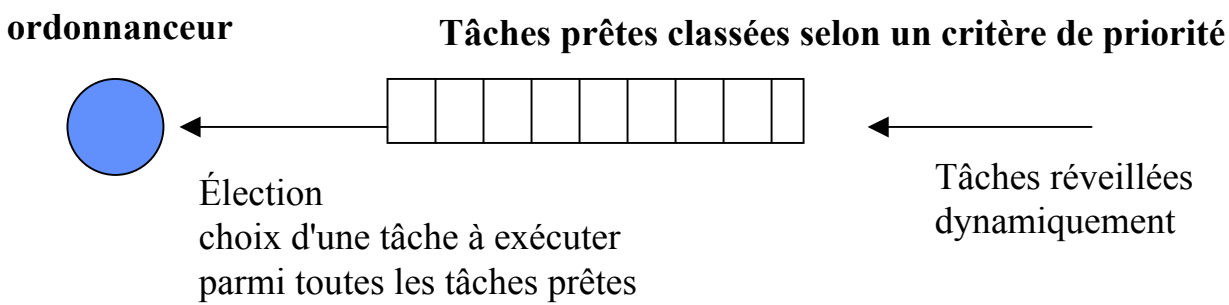
Construite hors ligne

Caractéristiques de l'ordonnancement temps réel

- **Caractéristiques de l'ordonnancement temps réel**

  - **Ordonnancement en ligne**

    La séquence d'exécution des tâches est établie dynamiquement par l'ordonnanceur au cours de la vie de l'application en fonction des événements qui surviennent. L'ordonnanceur choisit le prochaine tâche à élire en fonction d'un critère de priorité.



Caractéristiques de l'ordonnancement temps réel

- **Modélisation de l'application pour la certification**

  - **Tâches périodiques**

  Elles correspondent aux mesures sur le procédé ; elles se réveillent régulièrement (toutes les P unités de temps)

  ☞ **périodiques strictes : contraintes temporelles dures à respecter absolument**

  ☞**périodiques relatives : contraintes temporelles molles qui peuvent être non respectées de temps à autre (sans échéance)**

  ☞**périodiques à échéance sur requête (délai critique = période)**

Modèle de tâches Périodique stricte

Tp (r_0, C, R, P)

$0 \leq C \leq R \leq P$

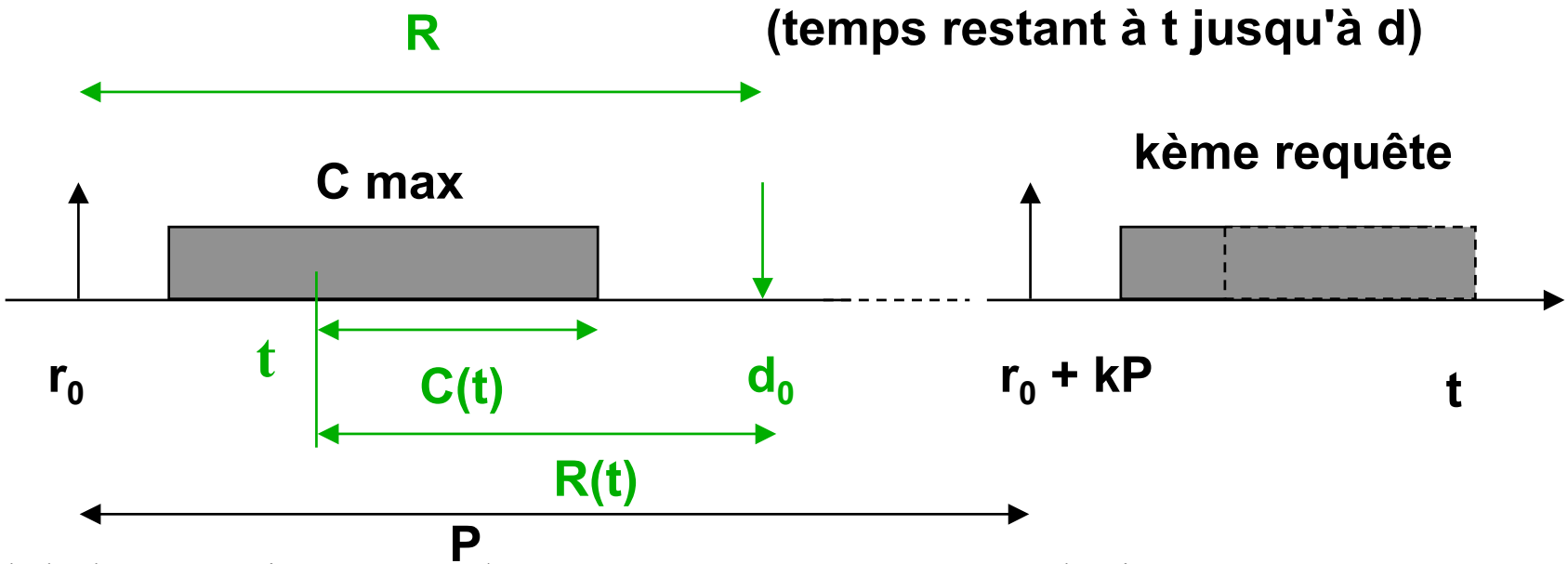
Tp (t, C(t), R(t))

R = P, à échéance sur requête

$d_k = r_{k+1}$

- **r_0, date de premier réveil**
- **P, période**
- **r_k, date de réveil de la kème requête**

  $r_k = r_0 + kP$
- **C, temps d'exécution**
- **R, délai critique**
- **d_k, échéance = r_k + R**
- **C(t) : temps d'exécution restant à t**
- **R(t) : délai critique dynamique (temps restant à t jusqu'à d)**



Caractéristiques de l'ordonnancement temps réel

- **Modélisation de l'application pour la certification**

  - **Tâches apériodiques**

  Elles correspondent aux événements ; elles se réveillent de manière aléatoire

  ☞ **apériodiques strictes : contraintes temporelles dures à respecter absolument**

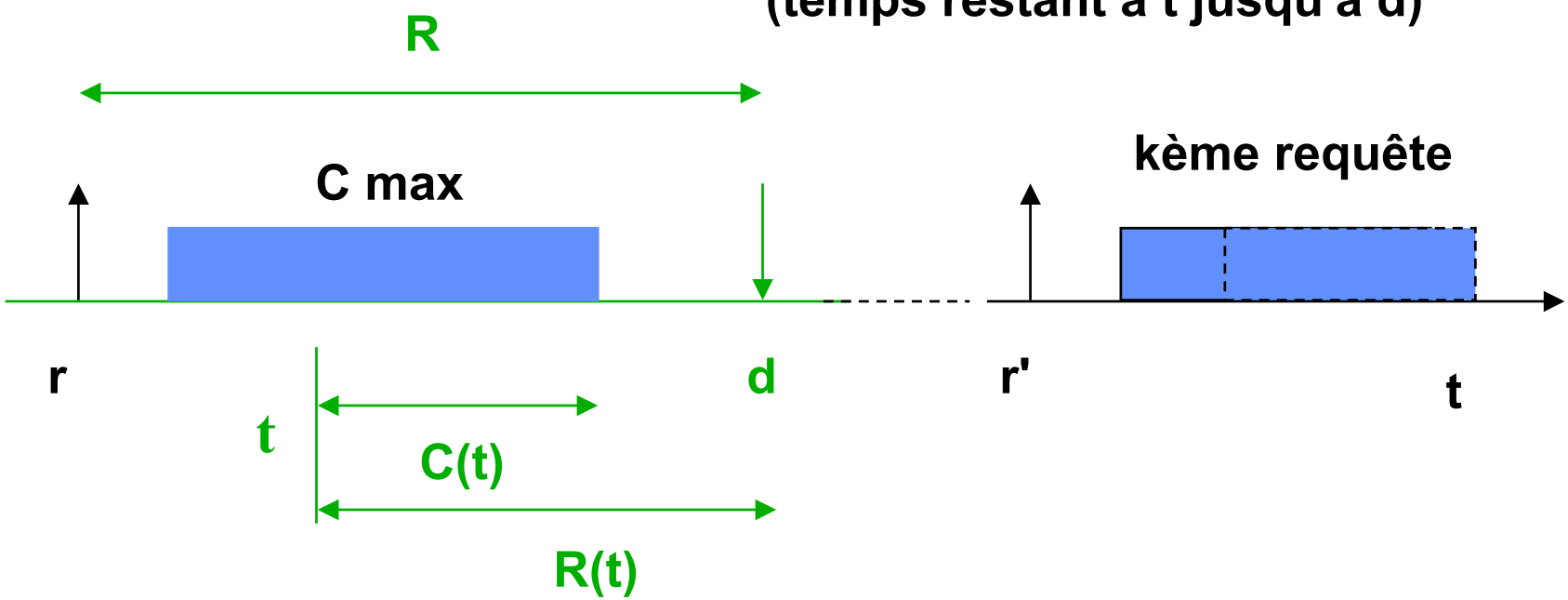
  ☞**apériodiques relatives : contraintes temporelles molles qui peuvent être non respectées de temps à autre (sans échéance)**

Modèle de tâches Apériodique stricte

Tap (r, C, R)

Tap (t, C(t), R(t))

- r, date aléatoire de réveil
- C, temps d'exécution
- R, délai critique
- d_k, échéance = r_k + R
- C(t) : temps d'exécution restant à t
- R(t) : délai critique dynamique (temps restant à t jusqu'à d)



Algorithmes d'ordonnancement pour les tâches périodiques indépendantes

- **Algorithmes en ligne et préemptifs avec un test d'acceptabilité évaluable hors ligne**

- **Nous ordonnançons un ensemble de tâches périodiques (configuration). Les priorités affectées aux tâches sont soit constantes (évaluées hors ligne et fixes par la suite), soit dynamiques (elles changent dans la vie de la tâche)**

- **L'ordonnancement d'un ensemble de tâches périodiques est cyclique et la séquence se répète de manière similaire sur ce que l'on appelle la période d'étude.**

- **Pour un ensemble de tâches à départ simultanée (t = 0), la période d'étude est : [0, PPCM(P_i)]**

Algorithmes d'ordonnancement pour les tâches périodiques indépendantes

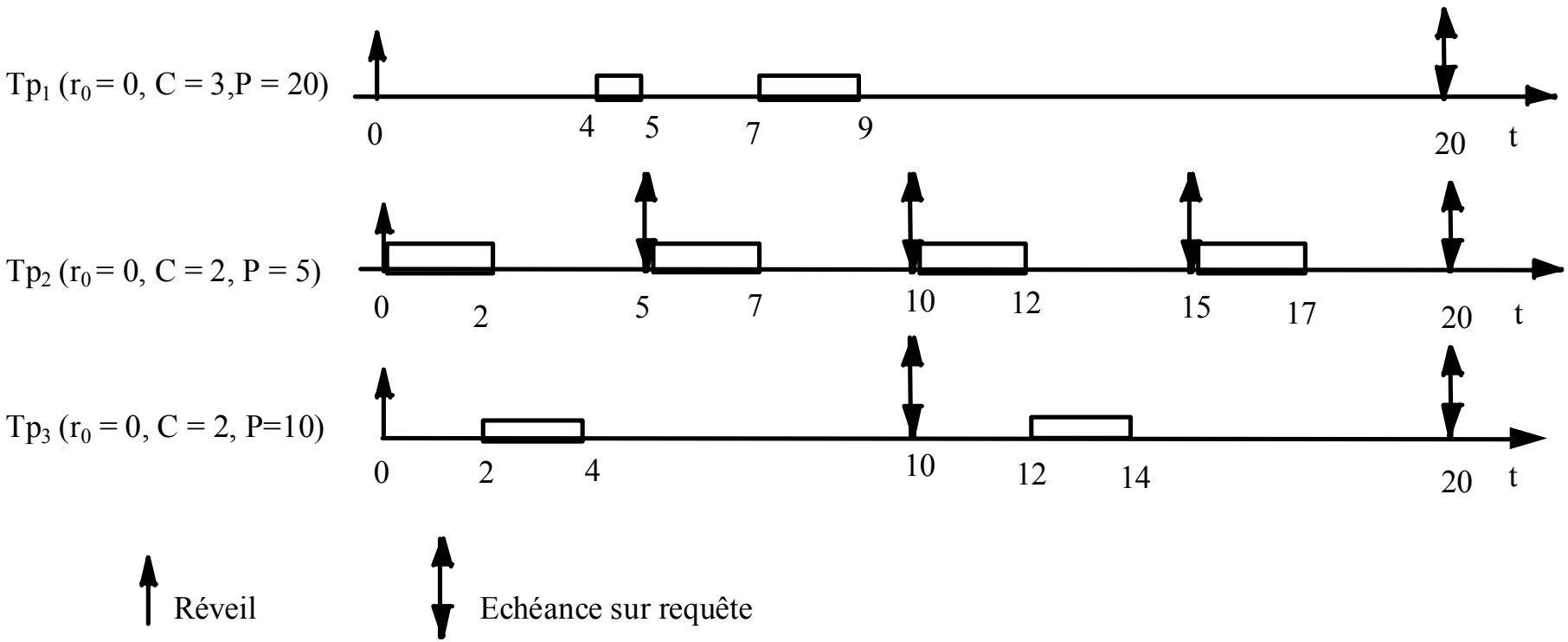
Rate Monotonic

- **Priorité de la tâche fonction de sa période. Priorité constante**
- **La tâche de plus petite période est la tâche la plus prioritaire**
- **Pour un ensemble de n tâches périodiques à échéance sur requête Tp_i (r_0, C_i, P_i), un test d'acceptabilité est (condition suffisante) :**

$$\sum_{i=1}^{n} \frac{C_i}{P_i} \leq n(2^{1/n} - 1)$$

Algorithmes d'ordonnancement pour les tâches périodiques indépendantes

Rate Monotonic



Algorithmes d'ordonnancement pour les tâches périodiques indépendantes

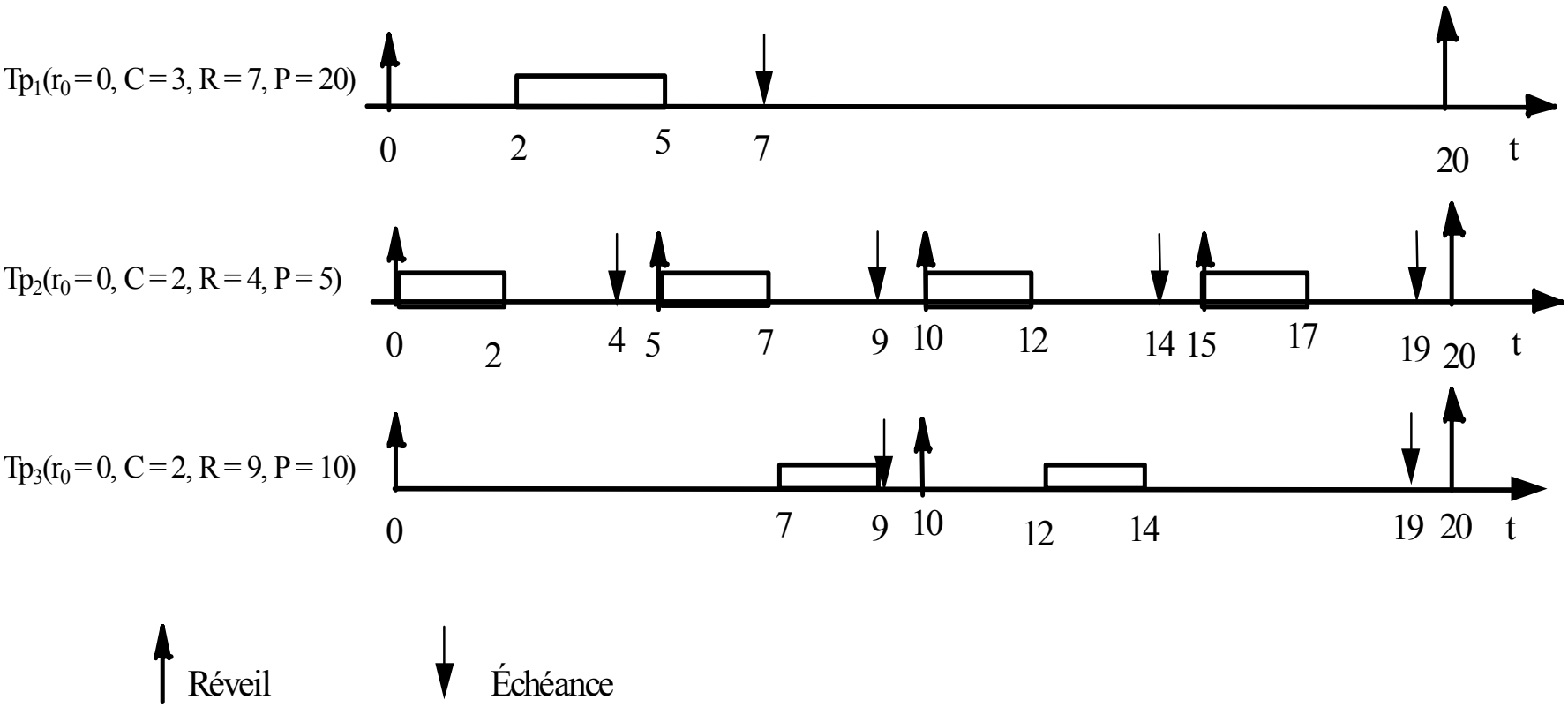
Inverse Deadline

- **Priorité de la tâche fonction de son délai critique. Priorité constante**
- **La tâche de plus petit délai critique est la tâche la plus prioritaire**
- **Pour un ensemble de n tâches périodiques à échéance sur requête Tp_i (r_0, C_i, R_i, P_i), un test d'acceptabilité est (condition suffisante) :**

$$\sum_{i=1}^{n} \frac{C_i}{R_i} \leq n(2^{1/n} - 1)$$

Algorithmes d'ordonnancement pour les tâches périodiques indépendantes

Inverse Deadline



Algorithmes d'ordonnancement pour les tâches périodiques indépendantes

Earliest Deadline

- **Priorité de la tâche fonction de son délai critique dynamique. Priorité dynamique**
- **A t, la tâche de plus petit délai critique dynamique (de plus proche échéance) est la tâche la plus prioritaire**

CNS (tâches ER)

$$\sum_{i=1}^{n} \frac{C_i}{P_i} \leq 1$$

CS (tâches quelconques)

$$\sum_{i=1}^{n} \frac{C_i}{R_i} \leq 1$$

CN (tâches quelconques)

$$\sum_{i=1}^{n} \frac{C_i}{P_i} \leq 1$$

Algorithmes d'ordonnancement pour les tâches périodiques indépendantes

Earliest Deadline