Chapitre IV : Ordonnancement des processus

Cours Système Exploitation II

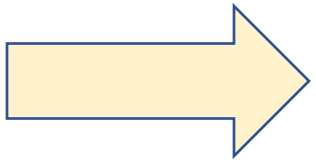
1 Linfo

ISIMM

2024/2025

Motivations

Lorsqu'un ordinateur est multiprogrammé, il possède fréquemment plusieurs processus/threads en concurrence pour l'obtention de temps processeur.

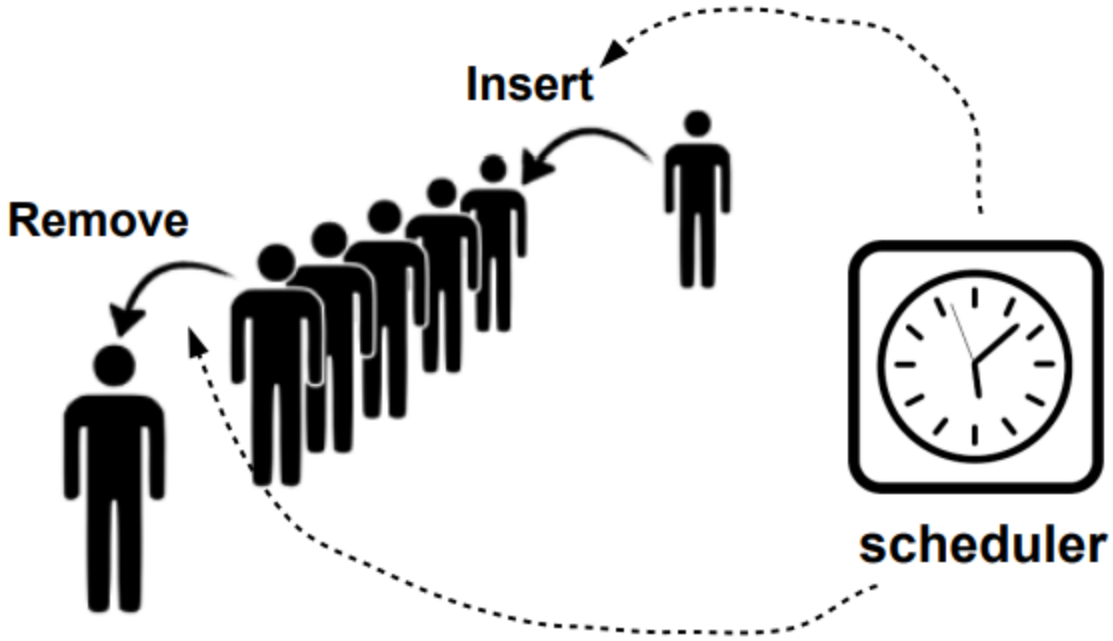


→ S'il n'y a qu'un seul processeur, un choix doit être fait quant au prochain processus à exécuter

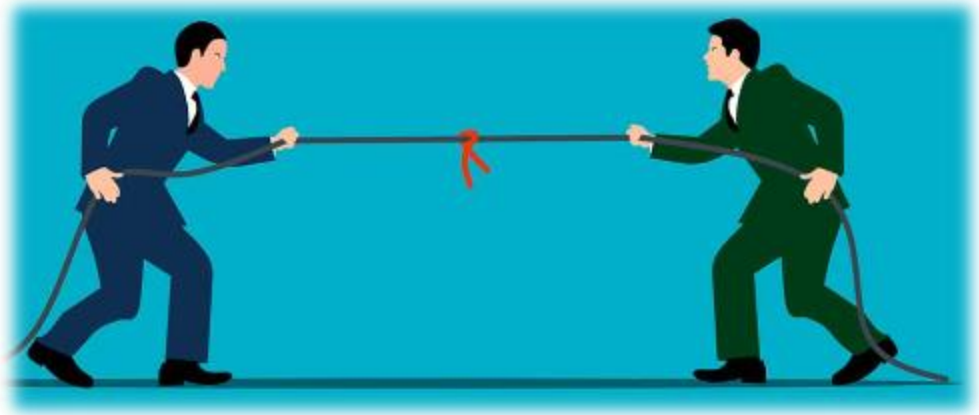
Définitions

La partie du système d'exploitation qui effectue ce choix se nomme l'ordonnanceur (scheduler) et l'algorithme qu'il emploie s'appel algorithme d'ordonnancement (scheduling algorithm).

Outre le fait de sélectionner le bon processus à exécuter, l'ordonnancement doit également se soucier de faire un usage efficace du processeur, car le passage d'un processus à l'autre sont coûteux en termes de temps de traitement



Défis

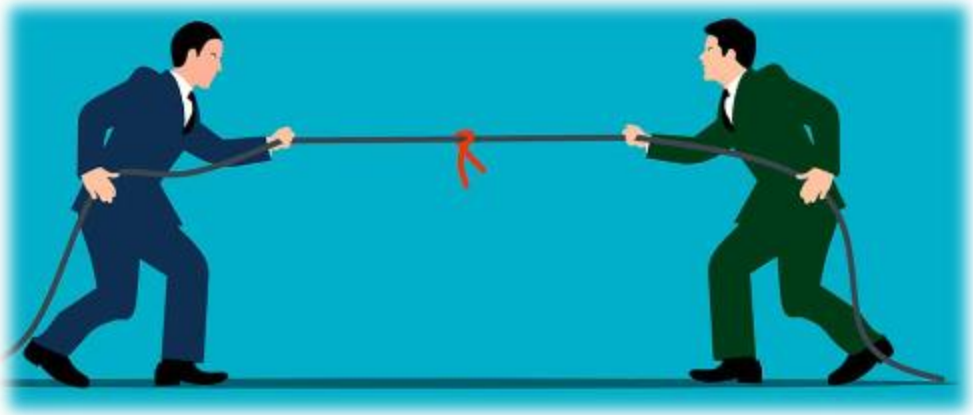


Temps de changement de contexte

Temps d’exécution

Efficacité : Utiliser au mieux le processeur

Objectifs de l'ordonnanceur



Les objectifs d'un ordonnanceur d'un système multi-utilisateur sont entre autres :

– S'assurer que chaque processus en attente d'exécution reçoive sa part de temps processeur.

– Minimiser le temps de réponse.

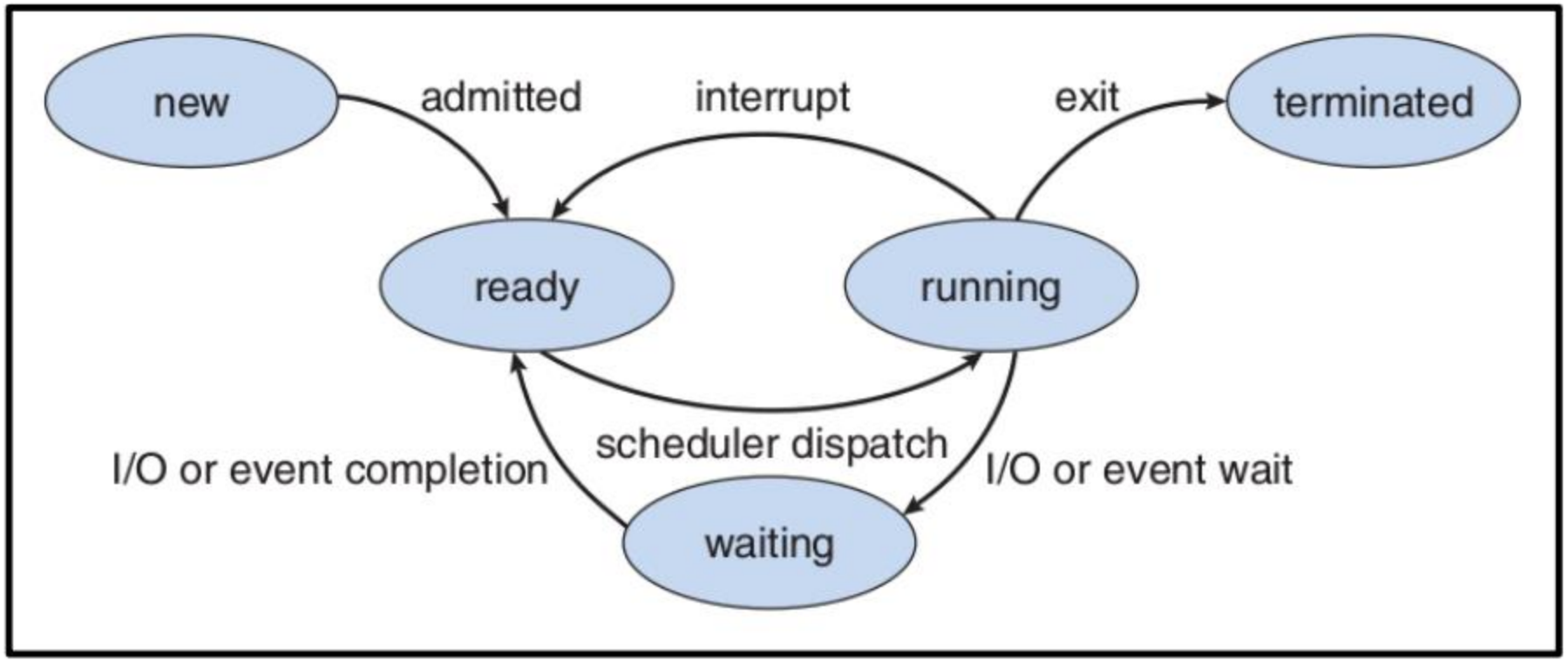
– Utiliser le processeur à 100%.

– Prendre en compte des priorités.

– Être prédictible.

États de processus et ordonnancement

Quand ordonnancer ?



Critères spécifiques d’ordonnancement

Débit = Nombre de processus terminés / Temps total écoulé

Temps d’exécution (Burst Time) : Le temps nécessaire au processeur ou à un périphérique d'E/S pour être occupé par le processus.

Temps d’arrive (Arrival Time): Le moment où un processus apparaît pour la première fois dans la file d'attente des processus prêts.

Critères spécifiques d’ordonnancement:

- **Temps de rotation ou temps de séjour (Turn Around Time (TAT))**:

TAT = temps de terminaison - temps d'arrivée, où le temps de terminaison est le moment où le processus se termine.

- **Temps d’attente (Waiting Time (WAT)):**

WAT = temps de début d'exécution - temps d'arrivée, où le temps de début d'exécution est le moment où le processus commence à être traité.

WAT = temps de séjour – temps d’exécution

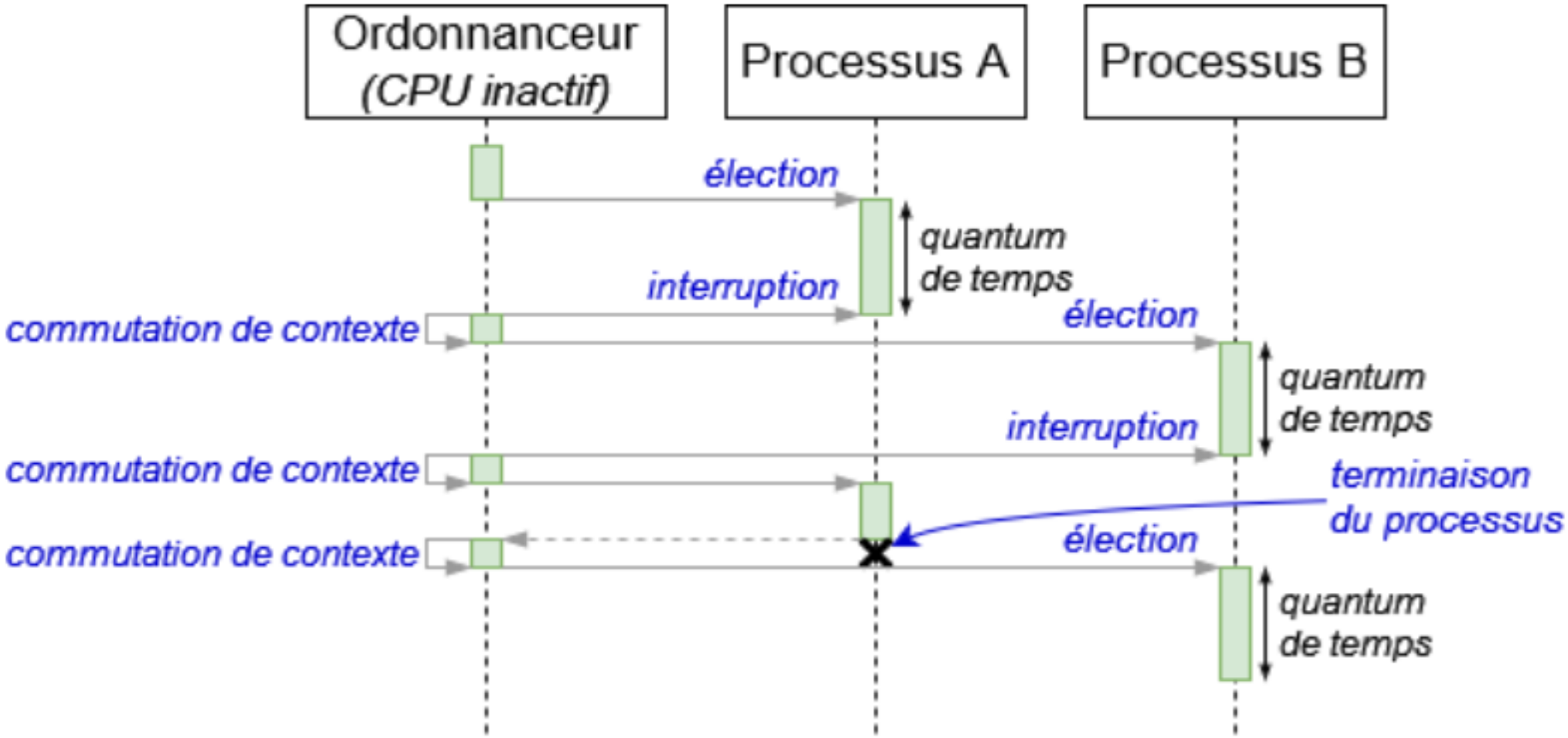
- **Temps d'attente moyen** (**Average Waiting Time):** moyenne de tous les temps d'attente des processus.

Ordonnanceurs préemptifs

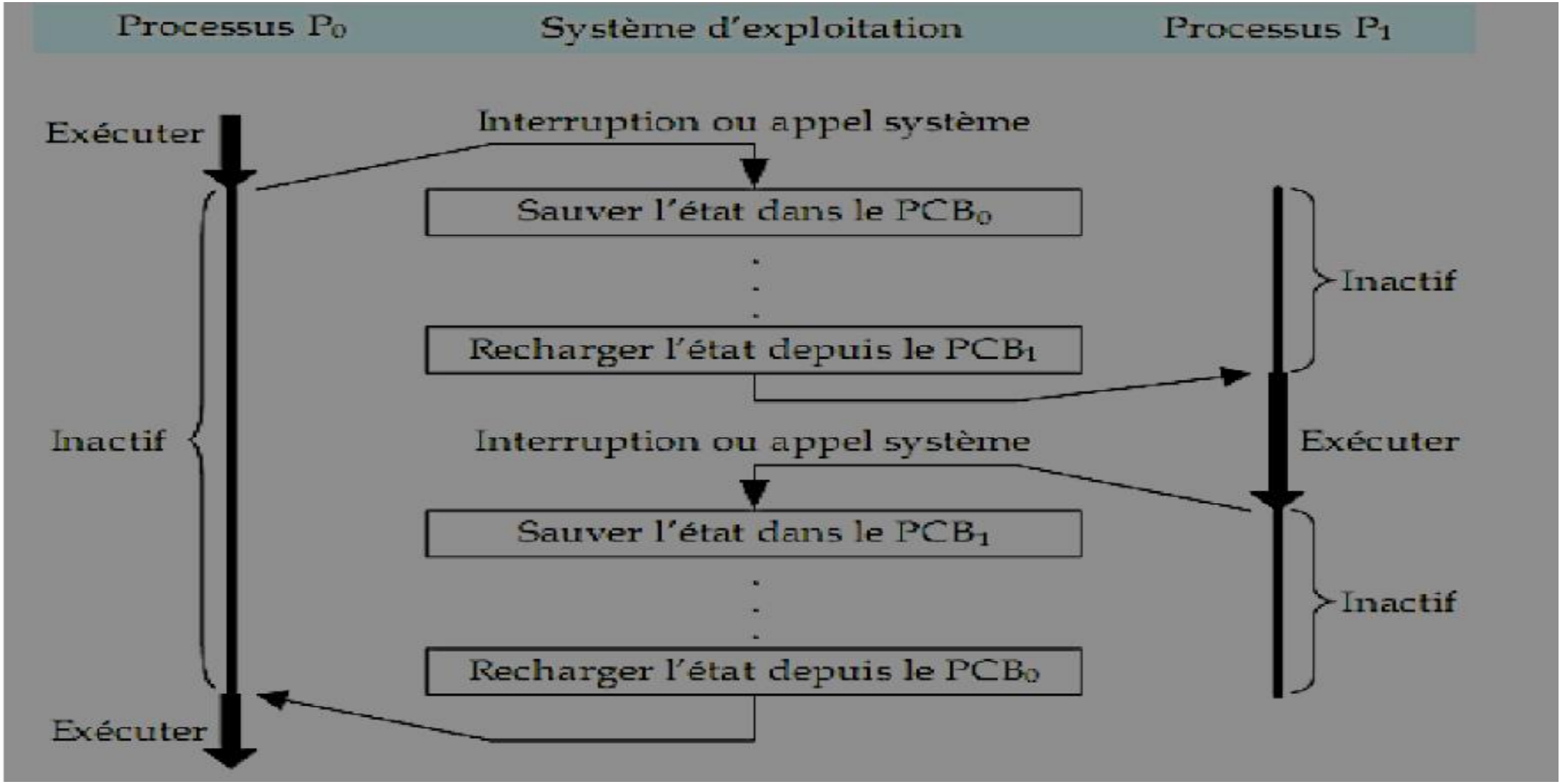
- **Si le processus courant doit suspendre son exécution au profit d'un autre :**
  - l'OS doit d'abord sauvegarder le contexte du processus
  - avant de charger le contexte du processus à lancer.

- **C'est qu'on appelle la commutation de contexte ou le changement de contexte**
  - Cette sauvegarde est nécessaire pour pouvoir poursuivre ultérieurement l'exécution du processus suspendu.

Commutation de contexte : Context switch



Commutation de contexte : Context switch



- Nous avons **4 processus** avec leur **temps d’arrivée, temps d’exécution et priorité**.
  - Règle des priorités : **1 est la plus haute priorité, et 3 la plus basse**.
  - SJF avec priorité (non préemptif) signifie que le processus en cours **ne sera pas interrompu** jusqu'à sa fin.
- **- Temps de commutation** (CT=2CT) : Chaque changement de processus **ajoute 2 unités de temps**.

Processus	Temps d’arrivée	Temps d’exécution	Priorité
P1	0	6	2
P2	0	8	1
P3	2	7	3
P4	3	3	2

Déroulement pas à pas de l’ordonnancement (SJF avec priorité) :

- **Étape 1 : Sélection du premier processus (t = 0)**
- **P1 et P2 sont arrivés.**
- **P2 a une priorité plus haute (1 vs 2).**
- **P2 commence et s'exécute de t=0 à t=8 (entièrement).**

- **Étape 2 : Sélection du processus suivant (t = 8)**
- **À t=8**, les processus disponibles sont :
  - **P1 (6 unités restantes, priorité 2)**
  - **P3 (7 unités restantes, priorité 3)**
  - **P4 (3 unités restantes, priorité 2)**
- **Le plus court parmi eux est P4(3 unités) avec une priorité de 2.**
- **Commutation (+2 unités), donc P4 démarre à t=10.**
- **P4 s'exécute de t=10 à t=13 (terminé).**

- **Étape 3 : Sélection du processus suivant (t = 13)**
- **À t=13**, les processus restants sont :
  - **P1 (6 unités restantes, priorité 2)**
  - **P3 (7 unités restantes, priorité 3)**
- **Le plus court est P1 (6 unités, priorité 2).**
- **Commutation (+2 unités), donc P1 démarre à t=15.**
- **P1 s'exécute de t=15 à t=21 (terminé).**

- **Étape 4 : Dernier processus P3 (t = 21)**
- **P3 est le seul processus restant.**
- **Commutation (+2 unités), donc P3 démarre à t=23.**
- **P3 s'exécute de t=23 à t=30 (terminé).**

Processus	Temps d'arrivée	Temps de fin	Temps de rotation TAT	Temps d'attente WT
P1	0	21	$21 - 0 = 21$	$21 - 6 = 15$
P2	0	8	$8 - 0 = 8$	$8 - 8 = 0$
P3	2	30	$30 - 2 = 28$	$28 - 7 = 21$
P4	3	13	$13 - 3 = 10$	$10 - 3 = 7$

RR avec commutation de contexte

- Voir l'exercice du TD

Effet de l'augmentation du Quantum

- **Si on augmente le Quantum :**

  - **Moins de commutations de contexte** → Moins de surcharge due au TCC, donc meilleure efficacité.
  - **Meilleur pour les processus longs** → Ils terminent en moins de tranches temporelles.
  - **Moins réactif pour les petits processus** → Ils attendent plus longtemps avant d'être exécutés.
  - **Peut se rapprocher du First-Come First-Served (FCFS)** si le quantum est trop grand, ce qui entraîne un mauvais temps de réponse pour les petits processus.

Effet de l'augmentation du Quantum

- **Si on diminue le Quantum :**
  - **Meilleur pour l'interactivité** → Les petits processus finissent rapidement.
  - **Plus de commutations de contexte** → Temps perdu dans les TCC, ce qui réduit l'efficacité globale.
  - **Peut pénaliser les processus longs** → Ils prennent plus de temps à se terminer car ils sont constamment interrompus.

Comment choisir un bon Quantum ?

- **Trop petit** → Trop de commutations de contexte, réduit la performance globale.
- **Trop grand** → Mauvaise réactivité, Round Robin devient presque du FCFS.
- **Bon équilibre** → Le **Quantum doit être légèrement supérieur au temps moyen des petits processus** pour minimiser les commutations tout en gardant une bonne réactivité.

- **Conclusion :**
- Si **TCC est élevé**, il faut **augmenter un peu le Quantum** pour éviter trop de pertes.
- Si on veut une **réactivité élevée**, il faut **un Quantum plus petit**, mais sans trop réduire l'efficacité.
- En général, un **Quantum optimal** est souvent choisi comme **entre 50% et 80% du temps moyen d'exécution des processus**.