

Ordonnancement des processus

- Objectifs :
 - ▶ Maximiser l'utilisation du processeur
 - Taux d'utilisation = $\text{Durée Actif (CPU)} / \text{Durée Totale}$
 - ▶ Equitable entre les différents processus
 - ▶ Présenter un temps de réponse acceptable
 - ▶ Assurer certaines priorités

- Pour cela un bon algorithme d'ordonnancement doit :
 - ▶ S'assurer que chaque processus reçoit sa part de quantum
 - ▶ Utiliser le temps processeur à 100 %
 - ▶ Minimiser le temps de réponse pour les utilisateurs en mode interactif
 - ▶ Minimiser l'attente des utilisateurs qui travaillent en batch

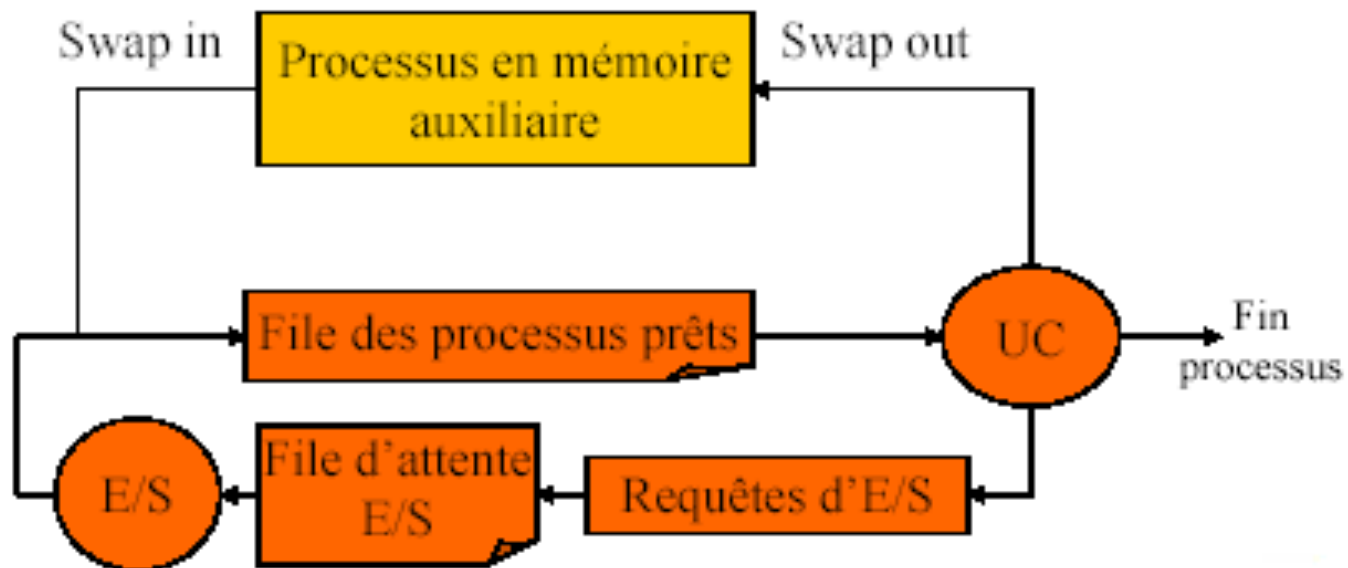
- Composant impliqué dans l'ordonnancement d'exécution
- Donne le contrôle de l'UC au processus choisi parmi les processus prêts :
 - ▶ Commutation de contexte
 - ▶ Passage en mode utilisateur
 - ▶ Branchement au bon emplacement dans le programme

- Trois niveaux :
 - ▶ Ordonnancement à long terme
 - Choix des travaux à entreprendre
 - Contrôle le degré de multiprogrammation
 - Long et peu fréquent

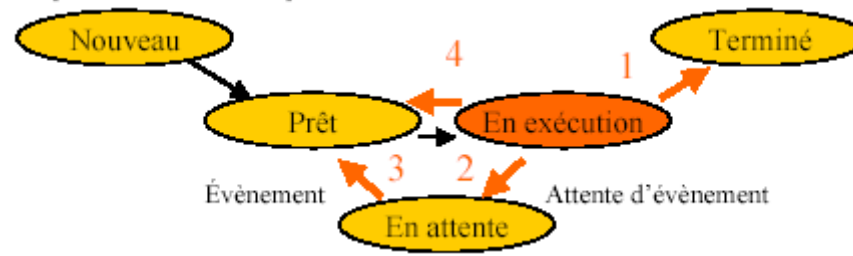
 - ▶ Ordonnancement à moyen terme
 - Permutation swapping

 - ▶ Ordonnancement à court terme (exécution)
 - Choix des processus à exécuter
 - Rapide et fréquent

■ Permutation Swapping

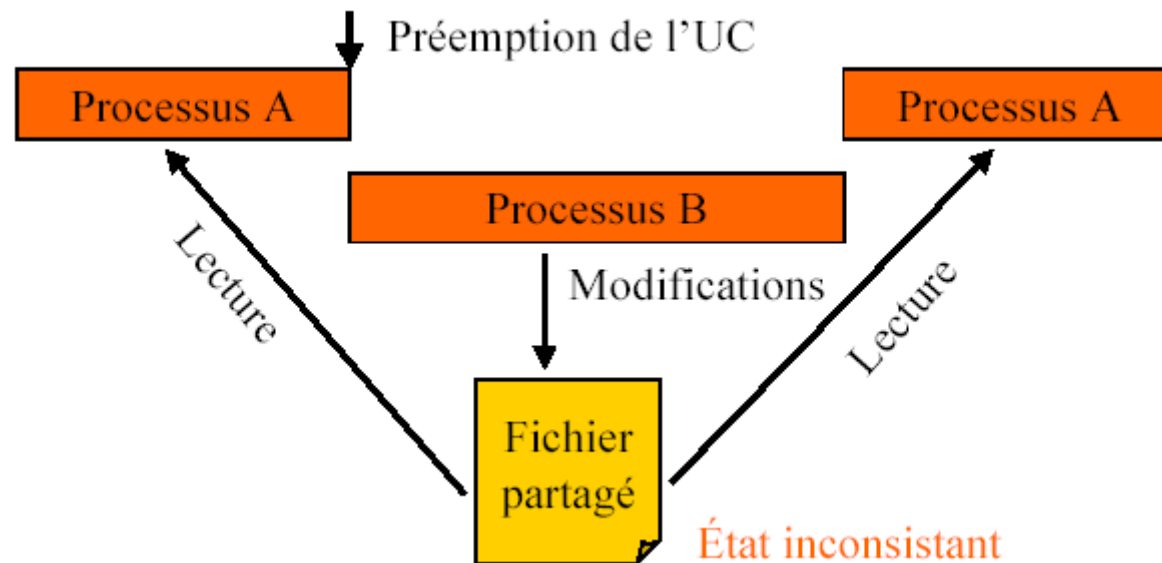


- Allocation CPU à l'un des processus prêts



- Cas 1 et 2 : non préemptif (coopératif)
- Cas 3 et 4 : préemptif

- ✓ Utilisation PCU (%)
- ✓ Débit (nb/s)
- ✓ Temps de rotation (temps)
- ✓ Temps d'attente (temps)
- ✓ Temps de réponse (temps)



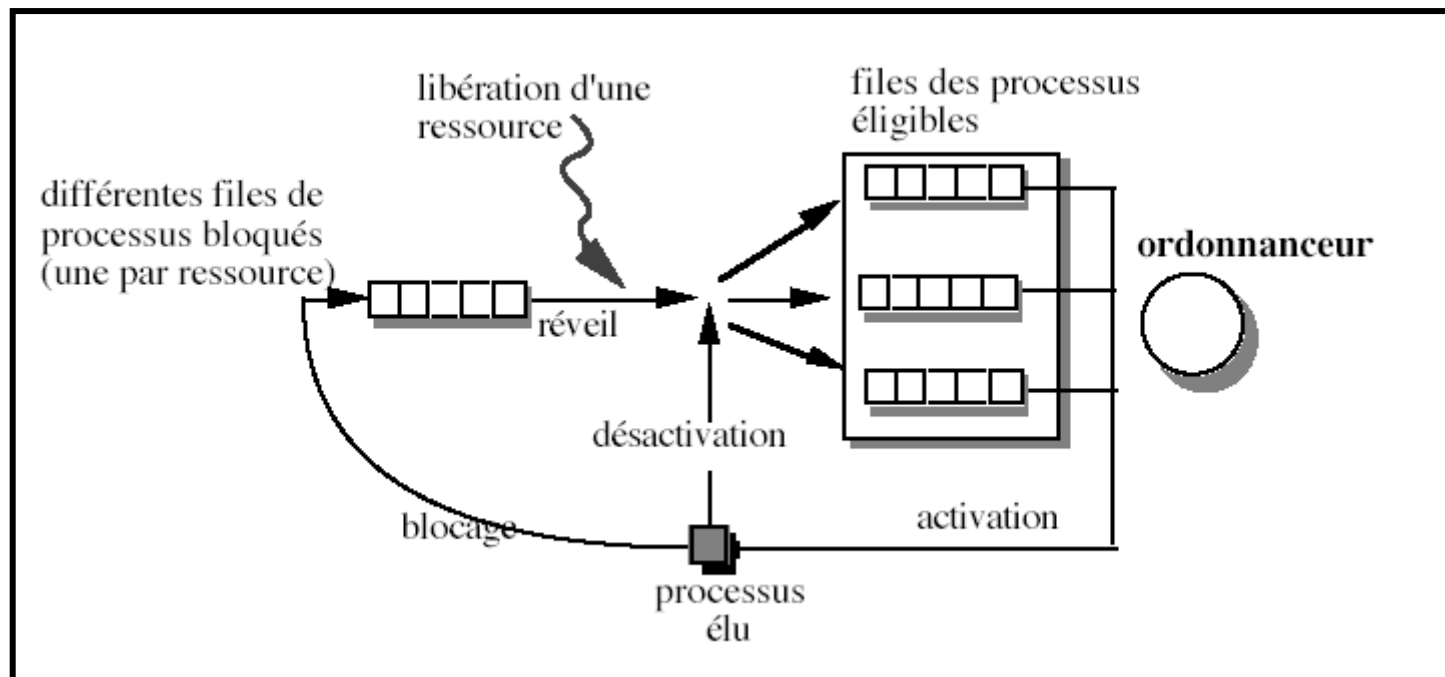
- Ressource allouée à une entité jusqu'à ce qu'elle en ait plus besoin
 - ▶ Inconvénients :
 - ☞ Ne peut convenir aux activités temps réels
 - ☞ Convient difficilement aux activités interactives tolérables aux applications faiblement multi tâche
 - ▶ Avantages :
 - ☞ Faciles à mettre en œuvre
 - ☞ Pas besoin de mécanismes matériels spécifiques
 - ▶ Au moment de la libération de la ressource :
 - ☞ l'ex détenteur de la ressource invoque l'algorithme d'ordonnancement
 - ☞ l'algorithme choisit l'utilisateur suivant
 - ☞ l'algorithme choisit la commutation de contexte

- Objectif : **Forcer le partage du temps d'utilisation** (modulo contraintes de priorités)
- Le détenteur de la ressource peut être **interrompu** avant d'avoir terminé
 - ▶ Lorsqu'un délai maximal expire
 - ▶ Lorsqu'une entité de priorité plus élevée demande la ressource
- **Utilisation d'un compteur de temps (timer)** qui génère une interruption périodiquement.
 - ▶ A chaque interruption de l'horloge : Poursuite de l'exécution ?
 - ☞ Le processus est suspendu et le processeur est alloué à un autre processus selon la politique d'ordonnancement ad oc
 - ☞ Un processus peut être suspendu à n'importe quel moment sans avoir été prévenu cela peut conduire à des conflits d'accès.
- **Politique mal adaptée au temps réel**

■ Le Tourniquet

- ▶ Chaque processus prêt dispose d'un quantum de temps pendant lequel il s'exécute
- ▶ Lorsqu'il a épuisé ce temps, ou qu'il se bloque : le processus suivant est élu et le remplace
- ▶ Le processus suspendu est mis en queue du tourniquet
- ▶ Paramètre à régler : Gestion du quantum
- ▶ Avantage : Utilisation équitable du processeur pour les processus présents en mémoire

■ Tourniquet avec priorités



■ Autres algorithmes d'ordonnancement

- ▶ PAPS (Premier Arrivé Premier Servi) ou FCFS
 - ⊕ L'ordre d'exécution est identique à l'ordre d'arrivée dans la file des processus éligibles

- ▶ PCA (Plus Court d'Abord)
 - ▶ On exécute les processus qui ont le temps d'exécution le plus court : prédiction de ce temps + durée maximale