L'ordonnanceur (1)

- L'**ordonnanceur de processus** ("scheduler") est responsable de la gestion des processus (activation, suspension, ...)

- Chaque changement d'état d'exécution du processus demande une intervention de l'ordonnanceur

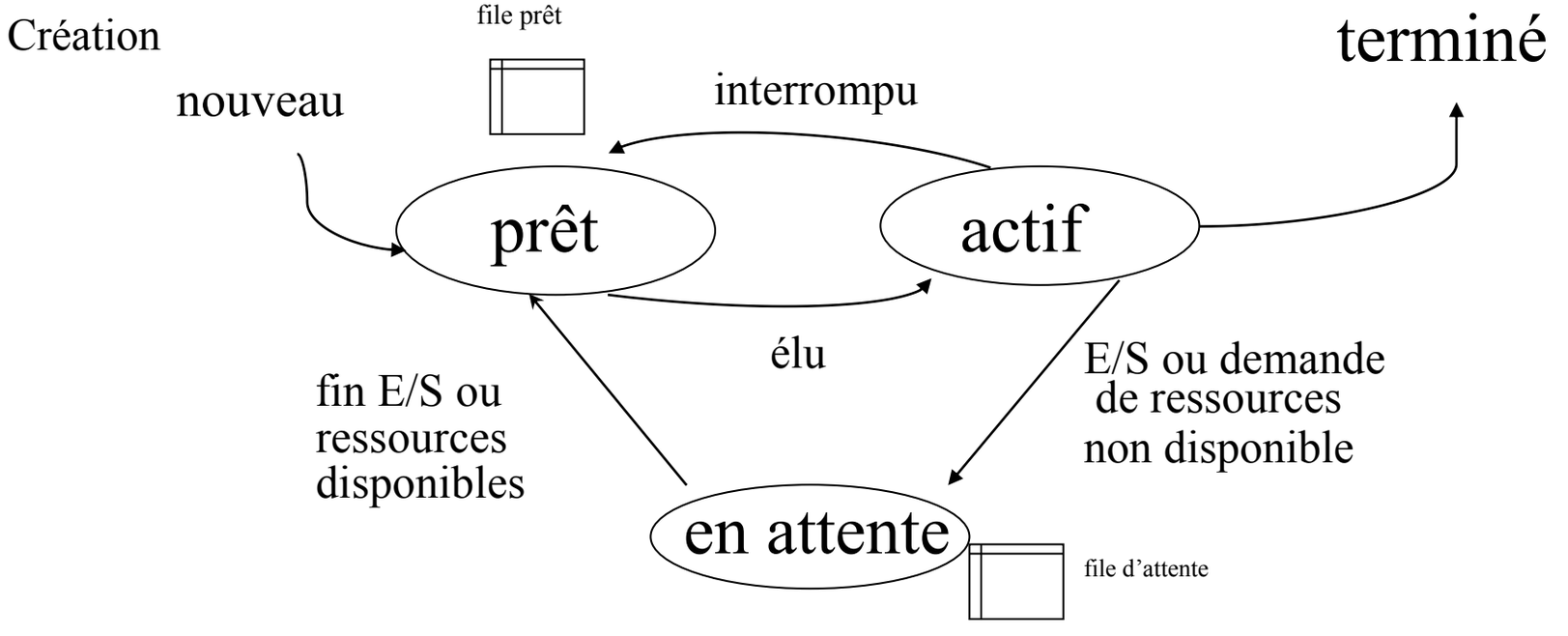


DIAGRAMME de TRANSITION

L'ordonnanceur (2)

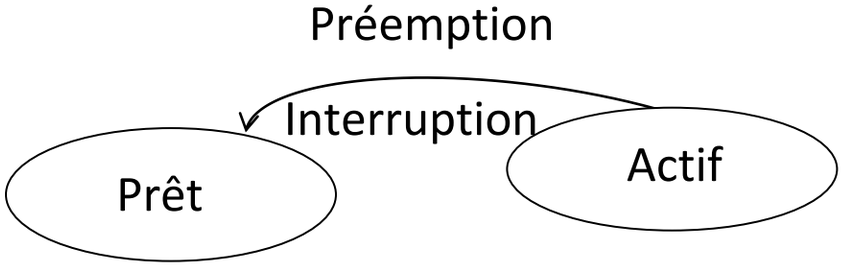
- Un ordonnanceur simple peut conserver les processus exécutables dans une **file prêt** ("ready queue")
- Sur un monoprocesseur, le processus à la tête de cette file est le **processus courant** (celui qui a le CPU présentement)
- Les nouveaux processus sont ajoutés à la file (à la fin de la **file ready**: FIFO)
- Lorsque le processus courant demande une E/S ou ressources non disponibles, l'ordonnanceur **sauve son contexte** puis **transfère le processus** de la file des processus prêts à la file d'attente, et **réveille processus en tête de la file ready**

Les objectifs de l'ordonnanceur

1. **Rendement :** Nombre de travaux exécutés par unité de temps.
2. **Temps de service:** Temps qui s'écoule entre le moment où un travail est soumis et où il est exécuté (temps d'accès mémoire + temps d'attente dans la file des processus prêts + temps d'exécution dans l'unité centrale + temps d'attente (prêt) + temps d'exécution des entrée/sortie).
3. **Temps d'attente :** Temps passé dans la file des processus prêts (file prêt).
4. **Temps de réponse**: le temps qui s'écoule entre la soumission d'une requête et la première réponse obtenue (particulièrement recherché pour les processus interactifs, p.e. editeur de texte).

Ordonnancement préemptif et non préemptif

- Si l'ordonnancement est non préemptif, la transition de l'état actif vers l'état prêt est interdite: un processus quitte le processeur s'il a terminé son exécution ou s'il se bloque (E/S);
- Si l'ordonnancement est préemptif, la transition de l'état actif vers l'état prêt est autorisée: un processus quitte le processeur s'il a terminé son exécution, s'il se bloque ou si le processeur est réquisitionné.



Algorithmes d'Ordonnancement

1. **Premier arrivé premier servi** ("FCFS"): les processus sont exécutés dans l'ordre où ils deviennent exécutables (l'ordre de création)
2. **Plus petite tâche en premier** ("Shortest-Job-First"): parmi les processus exécutables, celui qui va terminer le plus vite est exécuté en premier (suppose qu'on peut estimer le temps d'exécution à l'avance)
3. **Ordonnancement "round-robin" (cyclique)**: utilisation d'un timer (horloge) pour multiplexer le CPU; chaque processus obtient à tour de rôle un **quantum** ou **"time slice"** pour avancer son exécution
4. **Ordonnancement par priorité**: les processus exécutables les plus prioritaires exécutent en premier (FCFS pour un même niveau de priorité)

Analyse

- Soit 3 processus qui ont été soumis pour exécution dans l'ordre P1, P2, P3 et les temps d'exécution respectifs sont 24, 3, 3 (et qu'il n'y a pas d'E/S)
- Ordonnancement avec chaque approche

FCFS, Temps d'attente moyen = (0+24+27)/3=17 ;
Temps de réponse moyen= (24+27+30)/3= 27

SJF, Temps d'attente moyen = (0+3+6)/3=3 ;
Temps de réponse moyen= (3+6+30)/ 3= 13

RR (quantum=1), Temps d'attente moyen = (6+5+6)/3=6

Temps de réponse moyen= (30+8+9)/3=19

- Pour minimiser le temps d'attente, **SJF est optimal** mais RR s'en approche et a l'avantage de ne pas supposer les temps d'exécution connus à l'avance

Système de Priorité (1)

- Un système de priorité permet d'attribuer un **degré d'urgence** à chaque processus:
  - Il est critique qu'un processus de haute priorité **complète son travail rapidement**, même si c'est au dépend d'un processus de plus faible priorité
- Par exemple un processus qui doit **répondre rapidement à un événement** (panne de courant, arrivée d'un paquet du réseau) a une priorité élevée (autrement des données pourraient être perdues)
- La priorité d'un processus peut **être fixée à sa création** ou **variée pendant son exécution**

Système de Priorité (2)

- Les garanties de l'ordonnanceur vis-à-vis la priorité des processus varie d'un S.E. à un autre, mais en général:
  - ∀ processus en exécution P et ∀ processus exécutable P^0: $prio(P) \geq prio(P^0)$
  - Sur monoprocesseur, le processus en exécution a la **plus haute priorité des processus exécutables** (les autres processus doivent attendre qu'il n'existe plus de processus exécutable de plus haute priorité)

L'ordonnancement sous Linux

Chaque processus est qualifié par une priorité et attaché à l'une des politiques

- **Trois politiques d'ordonnancement**
  - SCHED_FIFO : Élit à tout instant le processus de plus forte priorité parmi les processus attachés à cette classe
  - SCHED_RR : Type tourniquet entre processus de même priorité
  - SCHED_OTHER : Politique à extinction de priorité
- Les processus attachés aux SCHED_FIFO et SCHED_RR sont plus prioritaires que les processus attachés à la politique SCHED_OTHER
- Deux types de processus :
  - ✓ Processus temps réel sont de priorité fixe: Politiques utilisées sont : SCHED_FIFO et SCHED_RR
  - ✓ Processus classiques sont de priorité dynamique : Politique utilisée est SCHED_OTHER