Système d'Exploitation

Sous-système de Gestion de Processus

Université de Tours
Faculté des Sciences et Techniques
Antenne Universitaire de Blois

Licence Sciences et Technologies
Mention : Informatique
2ème Année

Sameh.kchaou@univ-tours.fr
Supports de cours adaptés de Mohamed TAGHELIT

Plan

1. Sous-système de gestion de Processus
2. États d'un Processus
3. Ordonnancement d'Exécution

Sous-système Gestion de Processus

- **Sous-système de gestion des processus** dans un **système d'exploitation** (SE) :
  - ✓ Constitue le cœur d'un SE
  - ✓ coordonne toutes les actions nécessaires pour la gestion des processus
    - ➢ Cycle de vie d'un processus : création, exécution et terminaison
    - ➢ Ordonnancement (Scheduling) : les processus sont placés dans des files et leur exécution est ordonnée en fonction de leurs classe et niveau de priorité

Sous-système Gestion de Processus

- Commutation (*Switching*) → détermine combien de temps allouer l'UC à un processus et quand le lui retirer pour l'allouer à un autre

- Temps (*Timing*) → suivre le temps d'exécution d'un processus en surveillant le temps consommé

- Mémoire utilisée →coordination avec le sous-système de gestion mémoire pour les besoins d'allocation/libération mémoire.

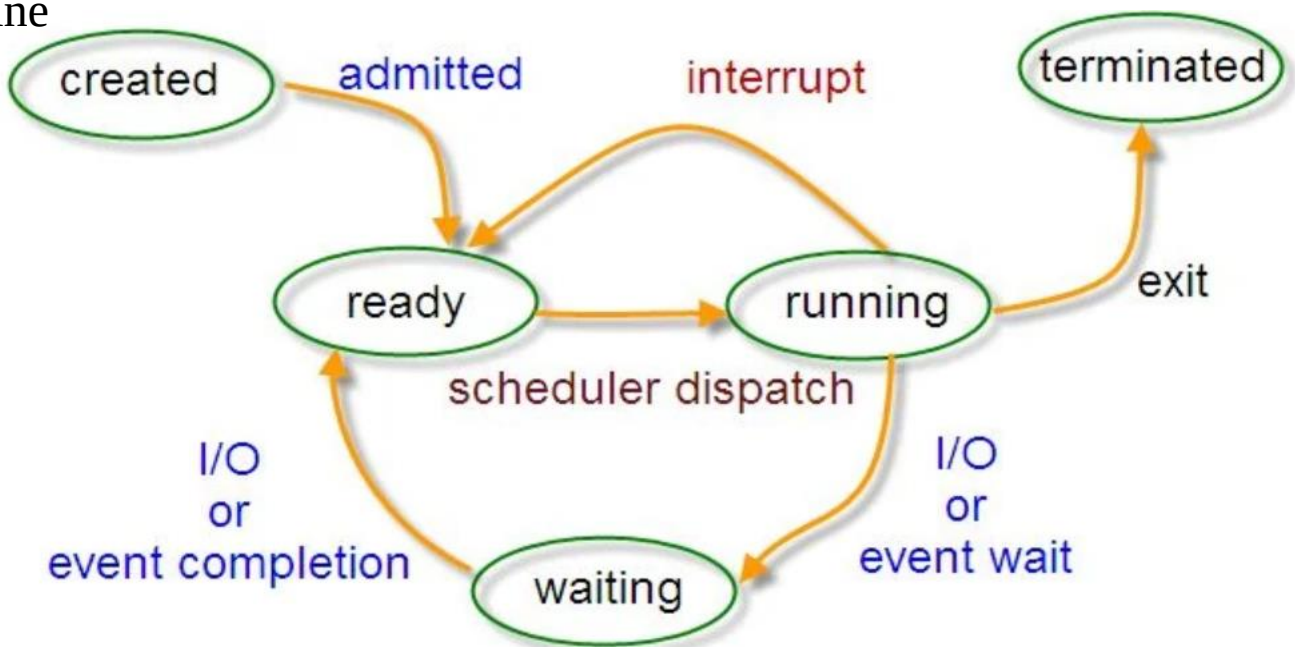
Sous-système Gestion de Processus

- Coordination avec le sous-système de fichier → association processus/fichiers, localisation et transfert en mémoire pour exécution

- Gestion des exceptions → émission et réception des signaux (erreurs durant l'exécution)

États d'un processus

- Quand un processus s'exécute, il change d'état.
- Chaque processus peut se trouver dans chacun des états suivants :
  - ✓ État Nouveau
  - ✓ État Prêt
  - ✓ État En Exécution
  - ✓ État Bloqué
  - ✓ État Suspendu
  - ✓ État Terminé



Ordonnancement d'Exécution

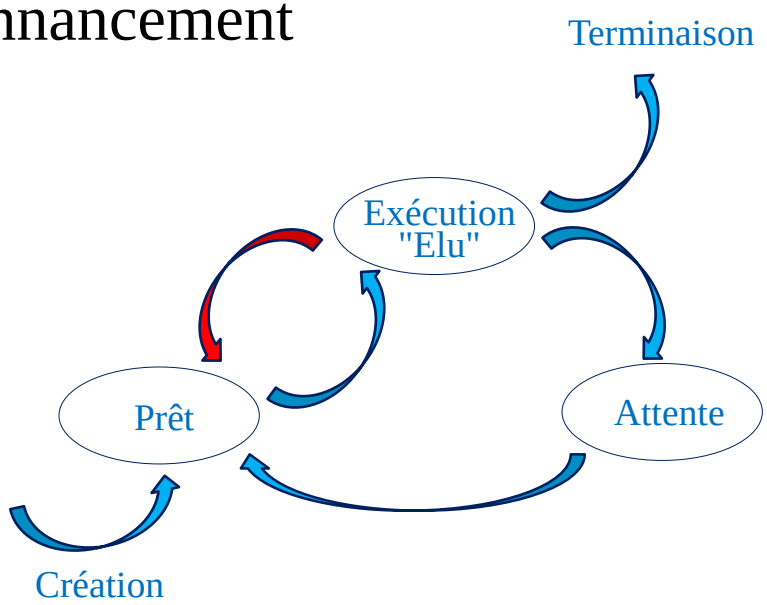
Ordonnancement d'Exécution

- Base des SE multiprogrammés → optimiser l'utilisation de l'Unité Centrale

- Circonstances exigeant un ordonnancement

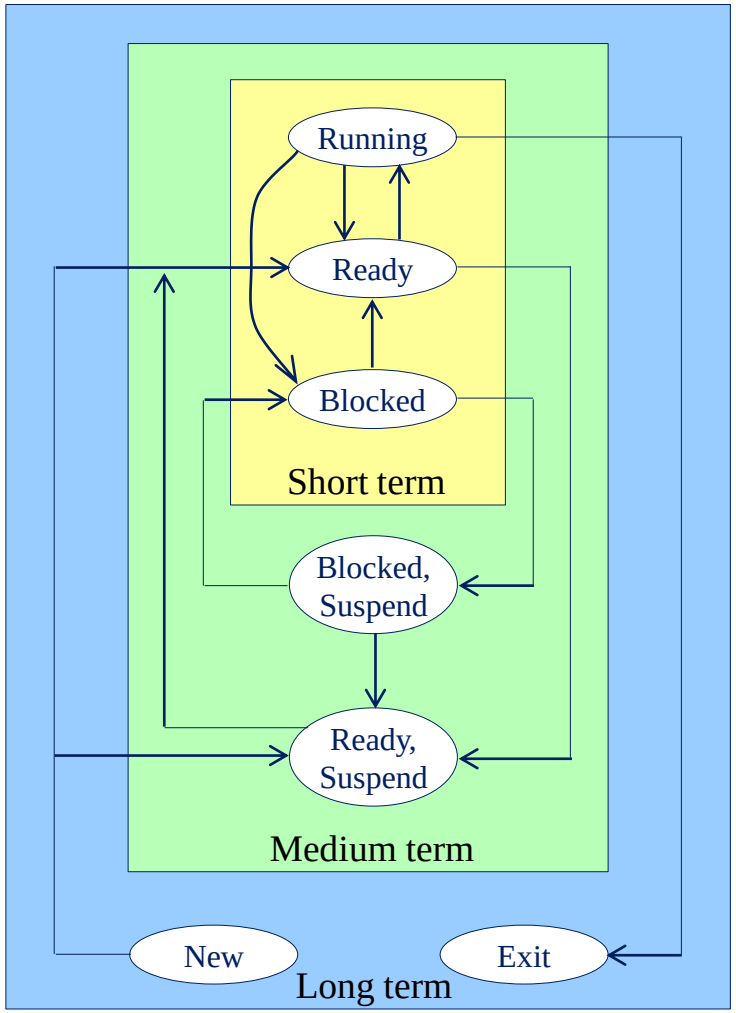
  - État d'exécution → État d'attente
  - État d'exécution → État prêt
  - État d'attente → État prêt
  - État d'exécution → Terminaison
  - Création → État prêt

Ordonnancement préemptif



Types d'Ordonnancement

- Ordonnancement à long terme
  - Décision d'ajout de processus pour exécution
  - Contrôle le degré de multiprogrammation
- Ordonnancement à moyen terme
  - Décision d'ajout de processus en mémoire
- Ordonnancement à court terme
  - Décision du choix du processus devant s'exécuter



Ordonnancement Préemptif

- Ordonnancement préemptif
  - Interruption de l'exécution du processus en cours
  - Possibilité de réquisition du processeur

- Ordonnancement non préemptif
  - Changement de contexte à la fin de l'exécution du processus en cours
  - Changement de contexte volontaire du processus en cours

Critères d'Ordonnancement

- Utilisation de l'UC → occuper l'UC au mieux [40% ↔ 90%]
- Débit ou Rendement (*throughput*) → nombre de processus terminés par unité de temps
- Temps de rotation ou de Service (*turnaround time*) → intervalle de temps entre la soumission d'un processus et son achèvement (¬ système interactif)
- Temps d'attente (*waiting time*) → somme des périodes passées en attente dans la file des processus prêts
- Temps de réponse (*response time*) → intervalle de temps entre la soumission d'une requête et la production de la première réponse

Méthodes d'ordonnancement

Ordonnancement FCFS

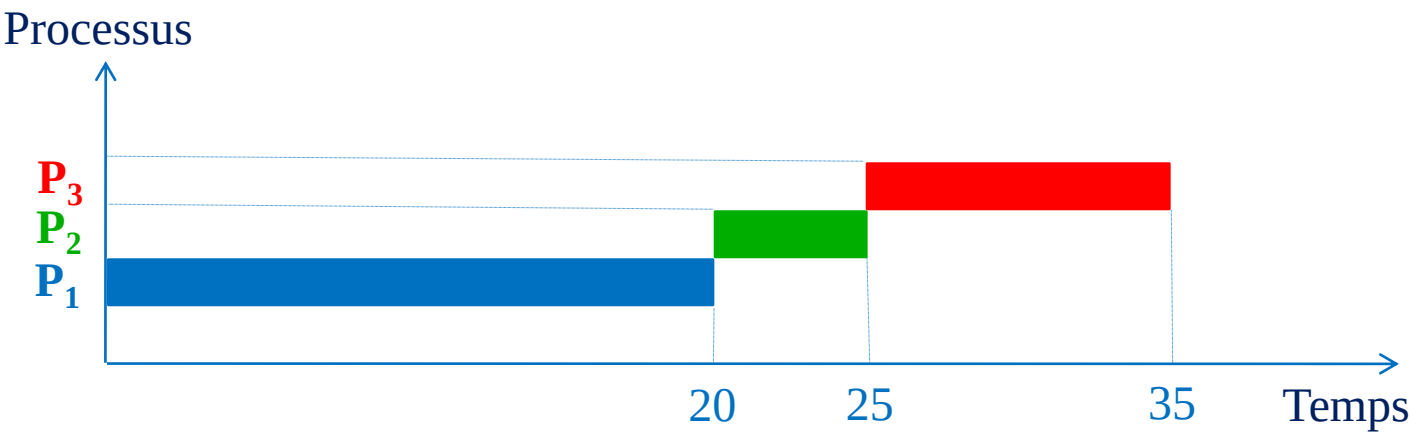
First-Come, First-Served

- Ordonnancement le plus simple
- Principe : premier arrivé, premier servi. Le premier processus qui demande l'unité centrale la reçoit en premier
- Pas de réquisition
- Implémentation → une seule file d'attente (processus prêts) FIFO
- Temps d'attente assez long
- Les processus courts peuvent être pénalisés

Exemple d'Ordonnancement FCFS

Processus	Durée	Instant Soumission
P_1	20	0
P_2	5	0
P_3	10	0

① ② ③ Ordre d'exécution



- Temps d'attente moyen = 15 ms

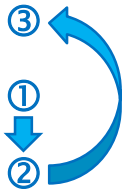
Ordonnancement SJF

Shortest-Job-First

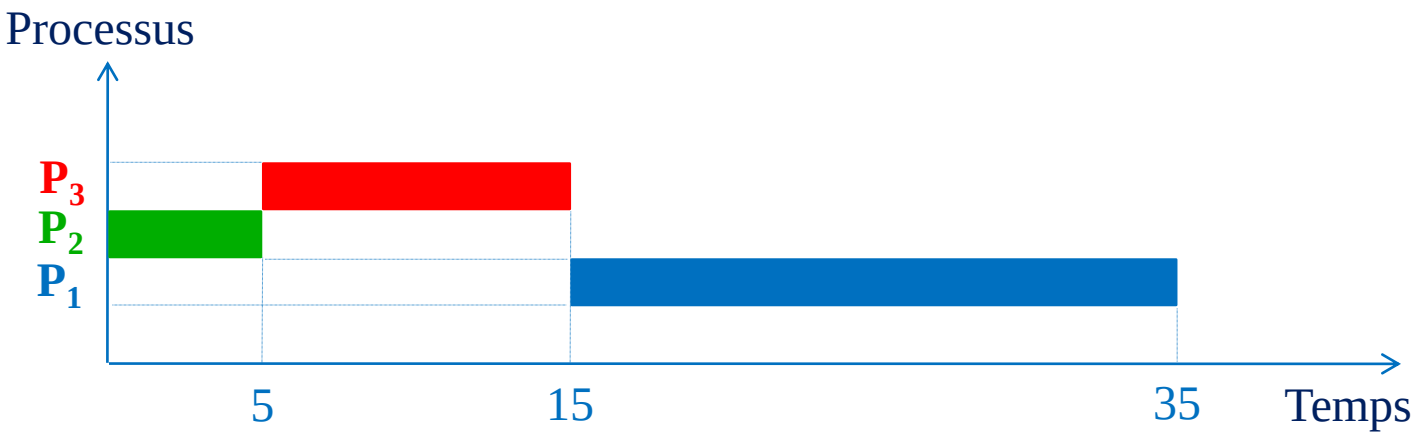
- Principe : le processus suivant le plus court d'abord
- Si deux processus ont une même durée, alors appel à l'ordonnancement FCFS
- Implémentation → une seule file d'attente (processus prêts) non FIFO
- Pas de réquisition
- Temps d'attente moyen optimal → en exécutant un processus court avant un processus long, on diminue le temps d'attente du processus court plus que n'augmente le temps d'attente du processus long ⇒ le temps moyen décroît
- Temps d'attente moyen < Temps d'attente moyen FCFS
- Difficulté à connaître le temps d'exécution des processus

Exemple d'Ordonnancement SJF

Processus	Durée	Instant Soumission
P_1	20	0
P_2	5	0
P_3	10	0



Ordre d'exécution



- Temps d'attente moyen = 6,66 ms

Commutation de Processus

- Une commutation est possible à chaque fois que le SE reprend le contrôle par rapport au processus en cours d'exécution.
- Événements pouvant donner le contrôle au SE :

Mechanism	Cause	Use
Interrupt	External to the execution of the current instruction	Reaction to an asynchronous external event
Trap	Associated with the execution of the current instruction	Handling of an error or an exception condition
Supervisor call	Explicit request	Call to an operating system function

- Interruptions
  - Interruption Horloge
  - Interruptions E/S
  - Défaut de page

Interruption Horloge

- Horloge matérielle interrompant le système à des intervalles de temps fixes
- La période de temps entre deux interruptions est appelée *CPU tick*, *clock tick* ou *tick* (Linux → *jiffy*)
- Sous Unix, généralement, *tick* = 10 ms
- Fréquence de l'horloge (nombre de *ticks/s*) → HZ (*param.h*)
- Les fonctions du noyau mesurent toujours le temps en nombre de *ticks*
- Routine de traitement dépendant du matériel
  - Doit être courte !
  - Très prioritaire !
- Définition d'un quantum → 6 ou 10 *ticks*

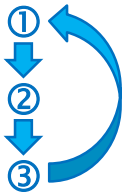
Ordonnancement par Tourniquet (RR)

Round-Robin

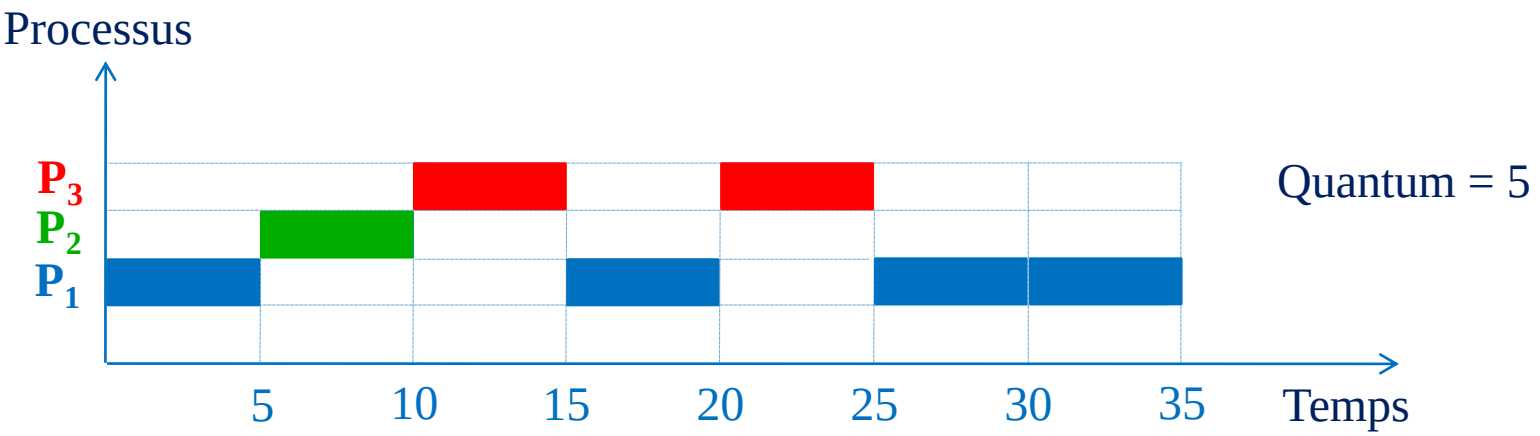
- Conçu spécialement pour les systèmes à temps partagé
- Principe : allouer le processeur à tour de rôle aux processus pour une durée prédéfinie = quantum
- Similaire FCFS + préemption
- Implémentation → une seule file d'attente (processus prêts) FIFO
  - Création d'un processus → insertion en queue de file
  - Élection → choisir le processus en tête de file
  - Fin quantum → réinsertion en queue de file
- Choix du quantum primordial
  - Trop petit → nombre important de commutations
  - Trop grand → temps de réponse important

Exemple d'Ordonnancement RR

Processus	Durée	Instant Soumission
P_1	20	0
P_2	5	0
P_3	10	0

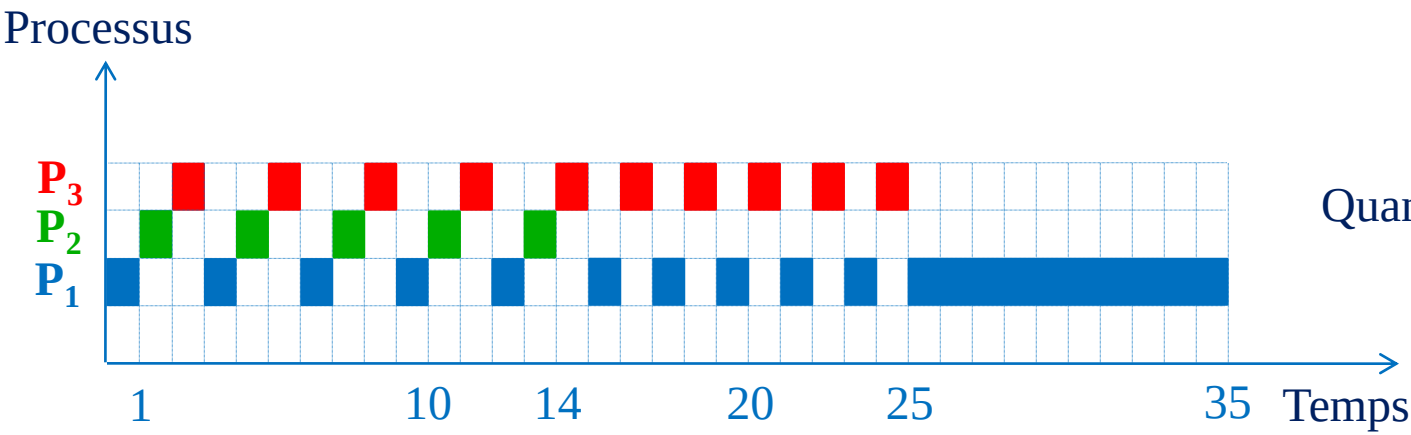


Ordre d'exécution



- Temps de réponse moyen = 5 ms

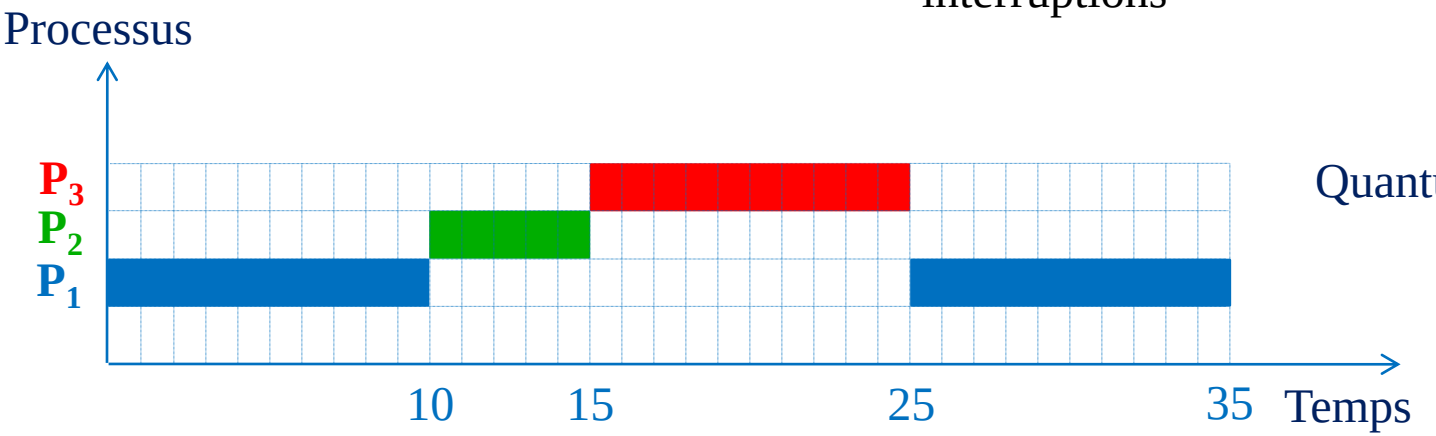
Exemple d'Ordonnancement RR



Quantum = 1

- Temps de réponse moyen = 1 ms

=> surcharge due à de fréquentes interruptions



Quantum = 10

- Temps de réponse moyen = 8,33 ms

Exercice : Analyse de l'Ordonnancement des Processus

Contexte :
Vous êtes un ingénieur système chargé de concevoir un ordonnanceur pour un système d'exploitation. Vous devez analyser différents algorithmes d'ordonnancement et leur impact sur le temps de réponse des processus.

Données :
Considérez les trois processus suivants, chacun avec une durée d'exécution et une priorité initiale :

Exercice 1 : Analyse de l'Ordonnancement des Processus

Processus	Durée (ms)	Priorité (PUSER)
P1	20	60
P2	5	60
P3	10	60

1. Calculez le temps d'attente moyen et le temps de réponse moyen pour les processus P1, P2 et P3 en utilisant l'algorithme FCFS.

2. Supposons que vous utilisez un algorithme Round-Robin avec un quantum de 5 ms. Calculez le temps d'attente moyen et le temps de réponse moyen pour les processus P1, P2 et P3. Expliquez en quoi l'utilisation du quantum affecte le temps de réponse des processus.

3. Calculez le temps d'attente moyen et le temps de réponse moyen pour les processus P1, P2 et P3 avec l'algorithme SJF.

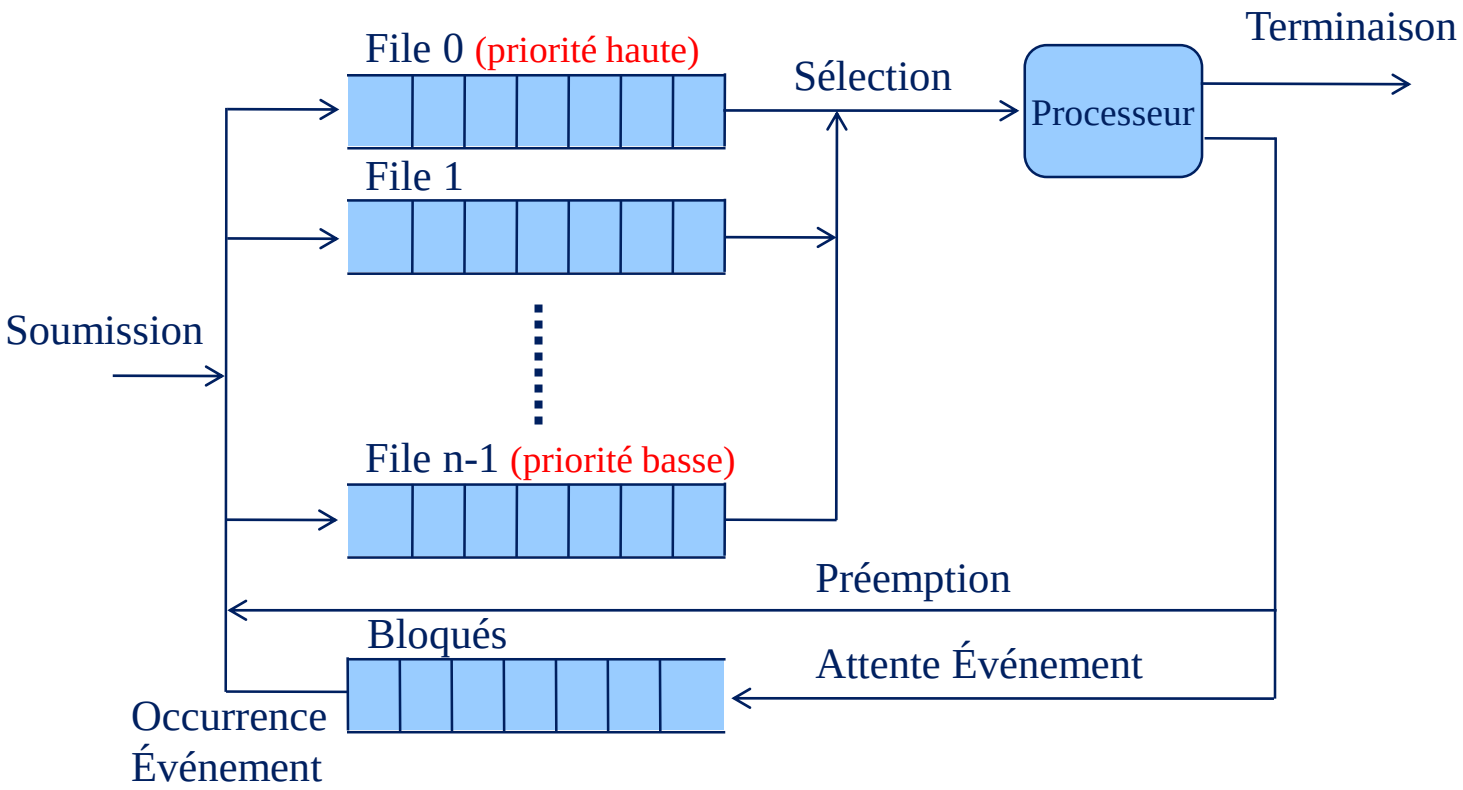
4. Si les processus ont des priorités différentes (par exemple, PP1 : 70, PP2 : 60, PP3 : 50), comment cela influence-t-il l'ordre dans lequel ils sont exécutés ?

Exercice : Analyse de l'Ordonnancement des Processus

Ordonnancement à Files Multiniveaux

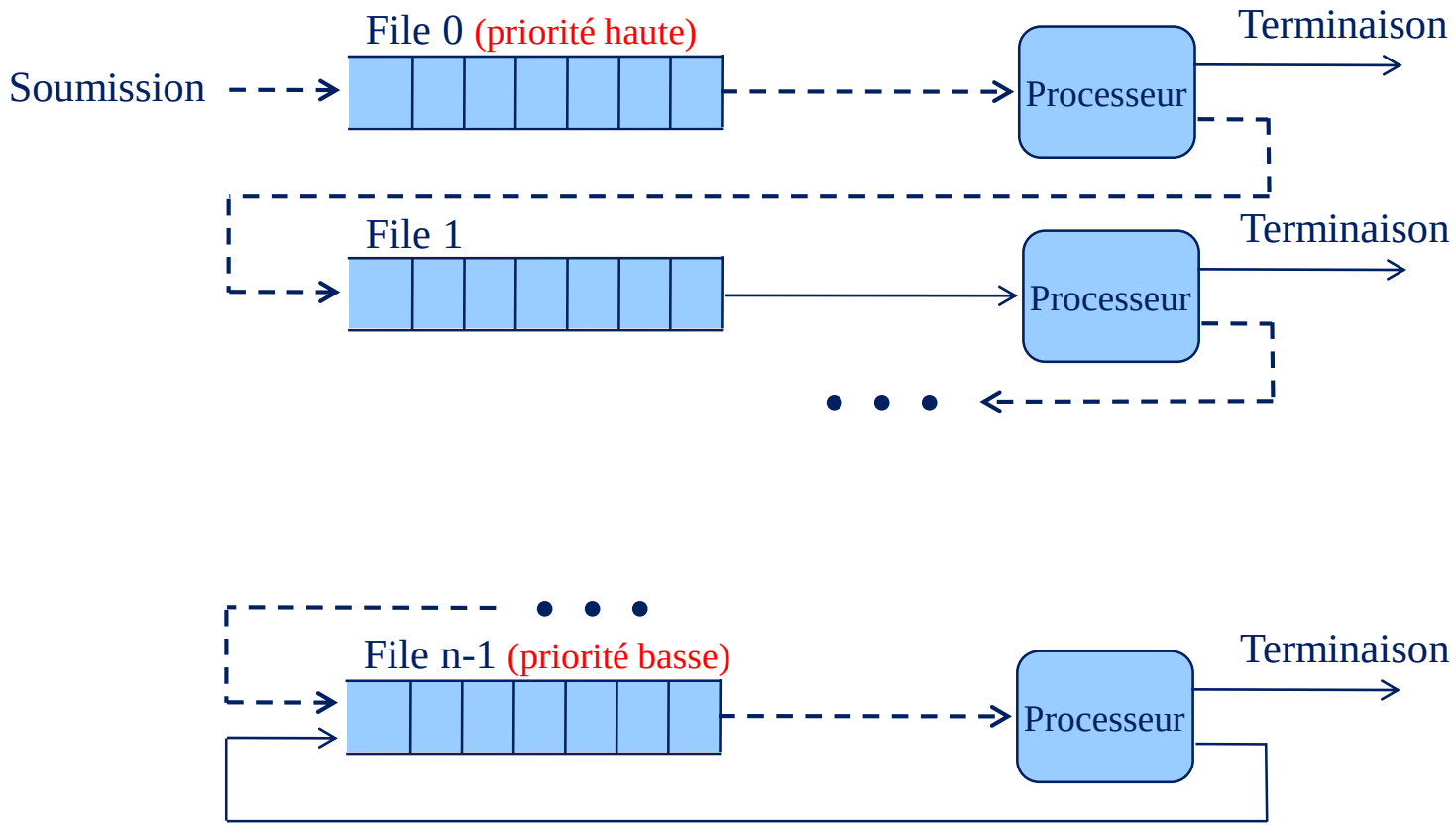
- Objectif : donner la possibilité de classer en différents groupes les processus et satisfaire chacun des groupes en fonction de ses besoins
  - Par exemple, les processus en avant-plan et les processus en arrière-plan qui ont des besoins de temps de réponse différents, …
- Principe :
  - Partitionner la file des processus prêts en plusieurs files séparées
  - Les processus sont affectés à une file (en fonction du type ou de la priorité des processus, par exemple)
    - Le processus est toujours affecté à la même file
    - Le processus change de file au cours de son exécution
  - Chaque file possède son propre algorithme d'ordonnancement
  - Établir un ordonnancement entre les files (généralement, ordonnancement préemptif à priorité fixée)
- Implémentation → une file par niveau

Ordonnancement avec Priorités Statiques



Risque de famine si les files + prioritaires sont toujours approvisionnées.

Ordonnancement avec Priorités Dynamiques



Remonter régulièrement (x quanta) d'un niveau toutes les tâches pour éviter la famine.

Contexte d’un processus

- Le contexte d'un processus fait référence à l'ensemble des informations nécessaires pour décrire l'état d'un processus à un moment donné.
  - Espace d’adressage utilisateur
    - code, données, segments de mémoire partagée, …
  - Informations de contrôle
    - zone u et structure proc
  - Appartenance (credentials)
    - UID, GID
  - Variables d’environnement
    - “variable = valeur”

Zone u (user)

- Importance de la Zone u :

  - Isolation et Sécurité : les processus ne peuvent pas interférer les uns avec les autres,
  - Gestion Efficace : Les informations stockées dans la zone utilisateur permettent au système d'exploitation de gérer efficacement les processus,
  - Facilitation de la Communication : Les arguments et les valeurs de retour dans la zone utilisateur facilitent la communication entre les processus et le système d'exploitation

Zone u (user)

- struct user → <sys/user.h>
- Fait partie de l'espace du processus
- mappable et visible uniquement lorsque le processus s'exécute (référence : u)
  - process control block (pcb),
  - pointeur vers la structure proc du processus,
  - UID et GID effectifs et réels,
  - arguments, valeurs de retour ou code d'erreur de l'appel système courant,
  - entête du programme (tailles du code, des données et de la pile ainsi que des informations sur la gestion mémoire),
  - statistiques d'utilisation CPU, quotas disques, limites des ressources

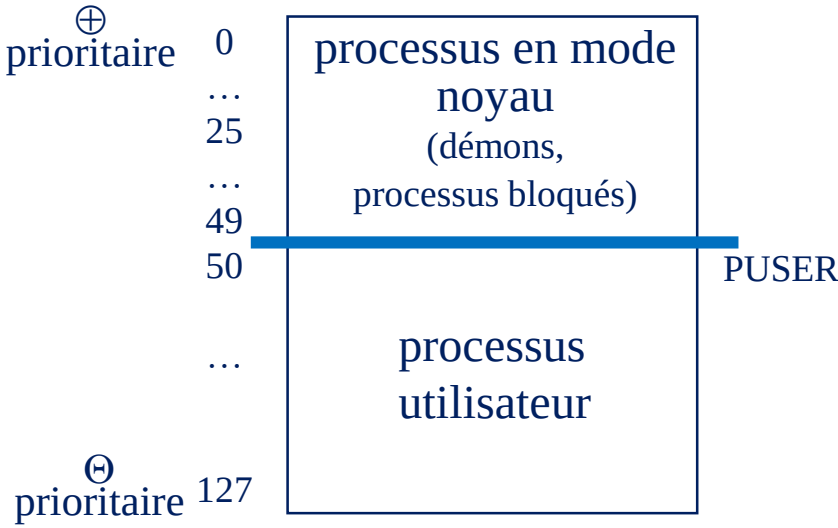
Structure `proc`

- struct proc → proc.h
- Appartient à l'espace noyau → visible par le noyau à tout instant, même quant le processus n'est pas en cours d'exécution,
  - identification : PID, GID,
  - état courant du processus,
  - pointeurs (avant, arrière) vers listes de processus prêts, bloqués, ...
  - événement bloquant,
  - priorité + information d'ordonnancement,

Priorités des processus

- La priorité d'un processus peut être une valeur entière comprise entre 0 et 127
- La structure proc contient les champs (relatifs à la priorité) suivants :

–	p_pri	Priorité (d'ordonnancement) courante
–	p_cpu	Mesure de l'utilisation récente de la CPU
–	p_nice	Incrément de priorité contrôlable par l'utilisateur
–	p_slptime	Temps passé par le processus dans l'état endormi



Calcul de Priorité

- Répartir équitablement le processeur →
  - baisser la priorité des processus lors de l'accès au processeur
- A chaque *tick*, la routine horloge incrémente p_cpu pour le processus courant (p_cpu++)
- Régulièrement, le noyau appelle *schedcpu()* pour réduire la valeur de p_cpu
  - appelée 1 fois par seconde
  - Pour tous les processus prêts :

    p_cpu = p_cpu * decay

    decay = ½ SVR3

    decay = (2*load_average)/(2*load_average+1) 4.3BSD

Exercice : Calcul de la priorité d'un processus en fonction de l'utilisation du processeur (p_cpu) et de l'incrément de priorité contrôlable par l'utilisateur (p_nice)

- **Énoncé** : Vous avez un processus avec les informations suivantes :

p_cpu : Utilisation récente du processeur, exprimée en unités (valeur entière entre 0 et 100).

p_nice : Valeur contrôlable par l'utilisateur, indiquant un ajustement de la priorité (valeur entière entre -20 et 19, où -20 est la priorité la plus haute et 19 est la plus basse).

- La formule de calcul de la priorité est la suivante :

$$\text{p\_pri} = 50 + \left(\frac{\text{p\_cpu}}{2}\right) + (2 \times \text{p\_nice})$$

50 est une priorité de base, p_cpu / 2 ajuste la priorité en fonction de l'utilisation du processeur, 2 * p_nice ajuste la priorité en fonction de la valeur p_nice

➢ Écrire un programme en C qui permet de calculer la priorité d'un processus à partir des valeurs suivantes : p_cpu : 70 et p_nice : -5. Utilisez la formule donnée pour calculer la priorité. Vérifiez si la priorité calculée dépasse les limites de 0 et 100 et ajustez-la si nécessaire.

QUIZ !

Quelle est la principale tâche d'un ordonnanceur dans un système d'exploitation ?

a) Allouer de la mémoire aux processus
b) Planifier l'exécution des processus
c) Gérer les fichiers du système
d) Contrôler les périphériques

QUIZ !

Quel est l'objectif principal de l'algorithme d'ordonnancement à priorité ?

a) Maximiser le temps de réponse
b) Éviter l'injustice dans le traitement des processus
c) Donner la priorité aux processus critiques
d) Réduire l'utilisation du processeur

QUIZ !

Dans l'algorithme Round Robin (RR), que représente le "quantum" ?

a) Le temps maximum pour exécuter tous les processus
b) La durée pendant laquelle un processus peut s'exécuter avant de céder le processeur
c) Le nombre de processus dans la file d'attente

QUIZ !

Quel algorithme minimise le temps d'attente moyen des processus ?

a) Premier arrivé, premier servi (FCFS)
b) Shortest Job Next (SJN)
c) Round Robin (RR)

QUIZ !

Dans un système à file d'attente multiniveau, comment les processus sont-ils classés ?

a) Selon leur priorité et leur type de tâche
b) Par ordre d'arrivée
c) Par la durée estimée de leur exécution

QUIZ !

La zone utilisateur (user space) est :

A. Une partie de la mémoire réservée exclusivement au noyau du système d'exploitation.
B. Une partie de la mémoire où les applications s'exécutent et interagissent avec le noyau via des appels système.
C. Une partie de la mémoire où sont stockés les pilotes matériels.
D. Une zone inaccessible aux applications.

QUIZ !

Le temps de réponse correspond à :

A. Le temps nécessaire pour qu'un processus passe de l'état prêt à l'état exécuté.
B. Le temps écoulé entre le moment où une requête est soumise et le moment où le système produit un résultat.
C. Le temps qu'un processus attend dans la file des processus prêts.
D. Le temps maximal qu'un système peut tolérer avant une défaillance.

QUIZ !

Le temps d'attente est défini comme :

A. Le temps nécessaire pour qu'un processus effectue une entrée/sortie.
B. Le temps écoulé entre la soumission d'un processus et son achèvement.
C. Le temps total qu'un processus passe dans la file des processus prêts avant d'être exécuté.
D. Le temps maximal pour qu'une interruption soit traitée.

QUIZ !

La principale différence entre le temps de réponse et le temps d'attente est :

A. Le temps de réponse inclut le temps d'attente, mais pas l'inverse.
B. Le temps d'attente inclut le temps de réponse, mais pas l'inverse.
C. Le temps de réponse mesure la durée de traitement réelle, tandis que le temps d'attente mesure uniquement le délai avant l'exécution.
D. Les deux sont des synonymes et mesurent exactement la même chose.

QUIZ !

La zone utilisateur joue un rôle important dans le système car :

A. Elle permet d'isoler les processus utilisateurs du noyau pour garantir la sécurité et la stabilité du système.
B. Elle donne un accès direct au matériel, évitant les appels système.
C. Elle exécute les processus critiques du système d'exploitation.
D. Elle est utilisée uniquement pour les processus en arrière-plan.