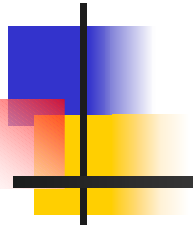
Chapitre IV : La gestion du processeur



4.1 Introduction

- De nombreux processus sont gérés par le SE
- L'efficacité théorique serait maximale si le nombre de processeurs était comparable à celui des processus
- Dans la plupart des cas la machine possède quelques processeurs (max 64 pour une machine généraliste) plus souvent 1 seul
- On a le plus souvent recours à une architecture dans laquelle le processeur effectue le travail principal, aidé par un petit nombre de processeurs spécialisés : processeurs arithmétiques, DSP spécialisés dans le traitement du son etc.

4.1 Introduction

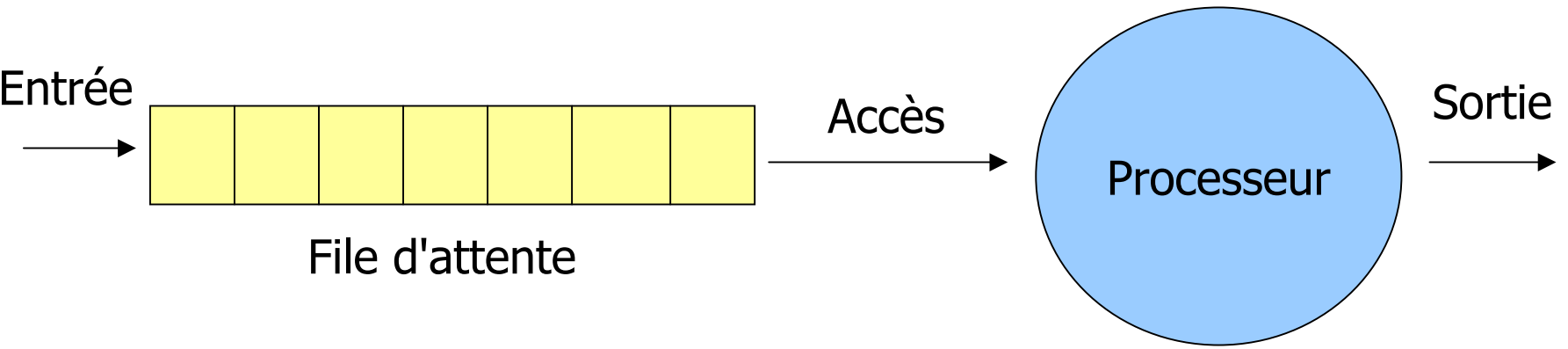
- **Problème :** un grand nombre de processus se partagent un seul processeur
- Il faut définir une politique d'accès au processeur (ordonnancement ou *scheduling*)
- Le mécanisme d'ordonnancement définit les critères selon lesquels les processus les plus prioritaires ont accès au processeur
- L'ordonnancement est indispensable dès lors que plusieurs utilisateurs se partagent la machine (ex: Win3.1 vs WinNT)

4.1 Introduction

- Un politique d'ordonnancement est nécessaire lorsque plusieurs utilisateurs (demandeurs) se partagent une ressource essentielle comme par exemple :
  - Un carrefour routier
  - Des guichets dans un administration
- Dans les tous cas les demandeur se retrouvent dans une **file d'attente**

4.1 Introduction

Composants utilisés dans l'ordonnancement des processus



4.1 Introduction

- Les principes d'ordonnancement sont les suivants :
  - **Ordre d'arrivée :** le premier arrivé est le premier servi (caisses au supermarché, à la poste etc.)
  - **Urgence :** le premier servi est celui dont le besoin d'accès rapide à la ressource est le plus grand (pompier)
  - **Importance :** le premier servi est celui dont l'accès à la ressource est le plus important (personne âgée dans les transports en commun)

4.1 Introduction

- Suivant la politique choisie le SE aura un comportement différent :
  - Démocratique
  - Place aux plus vieux processus
- Mais il ne sera pas destiné a tous les types d'utilisations (système temps réel)
- Pour s'adapter à des cas particuliers, ces principes sont souvent combinés

4.2 Pénalisation

- Lorsqu'un processus ne peut pas accéder directement à une ressource qu'il convoite on dit qu'il est **pénalisé**
- La **pénalisation** que subit un processus peut être représentée par son **temps d'attente**
- Plus précisément le temps d'attente est le nombre d'unité des temps durant lesquelles le processus est présent dans la file d'attente (sans être exécuté)

4.2 Pénalisation

- La mesure de la pénalité peut être affinée :
  - En relativisant le temps d'attente par rapport à la durée du processus
- Le taux de retard T est le rapport :
  - T=d/(a+d) où
  - d est la durée du processus
  - Et a+d le temps total du processus passé dans le système (a=durée d'attente)
- Dans le cas idéal T=1

4.3 Politique d'ordonnancement

Deux paramètres sont à prendre en compte :

- **Stratégie d'accès** au processeur via la file d'attente
- **Utilisation de processeur**

4.3.1 Traitement jusqu'à terminaison

- **Principe :** la politique de traitement du processus jusqu'à terminaison
  - accorde le processeur à un processus
  - ne l'interrompt jamais quelque soit sa durée, l'importance ou l'urgence ne sont pas pris en compte
- **Exemple :** file d'attente à la SNCF

4.3 Politique d'ordonnancement

- **Premier arrivé :** la stratégie FIFO (First In, First Out) est la plus simple
  - Le traitement du processus est séquentiel
  - Premier arrivé en file d'attente, premier traité
  - Les processus courts sont pénalisés
  - **Exemple :** photocopie utilisé par une personne qui photocopie un livre et d'autre qui veulent photocopier une page

4.3 Politique d'ordonnancement

- **Moindre durée :**
  - Il s'agit d'une évolution de la stratégie précédente
  - On garde le principe d'occupation du processeur jusqu'à terminaison
  - La file d'attente est ordonnée non plus de façon chronologique mais en fonction du temps d'exécution nécessaire (on fait passer en tête les travaux courts)

4.3 Politique d'ordonnancement

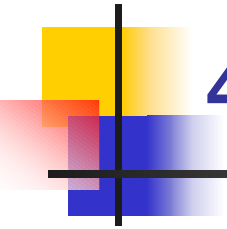
- A chaque arrivé d'un processus :
  - Il est inter-classé dans la file d'attente
  - En fonction de sa durée
- Ce traitement ralentit le traitement global
- Les travaux longs sont pénalisés au pire ils risquent de ne jamais être exécutés si beaucoup de petits travaux sont soumis au système

4.3 Politique d'ordonnancement

4.3.2 Réquisition

- Le principe de la **réquisition** (**préemption**) concerne la gestion du processeur
- Il consiste à décider en fonction de certains critères, de remettre le processus en file d'attente avant la fin de son exécution

4.3 Politique d'ordonnancement



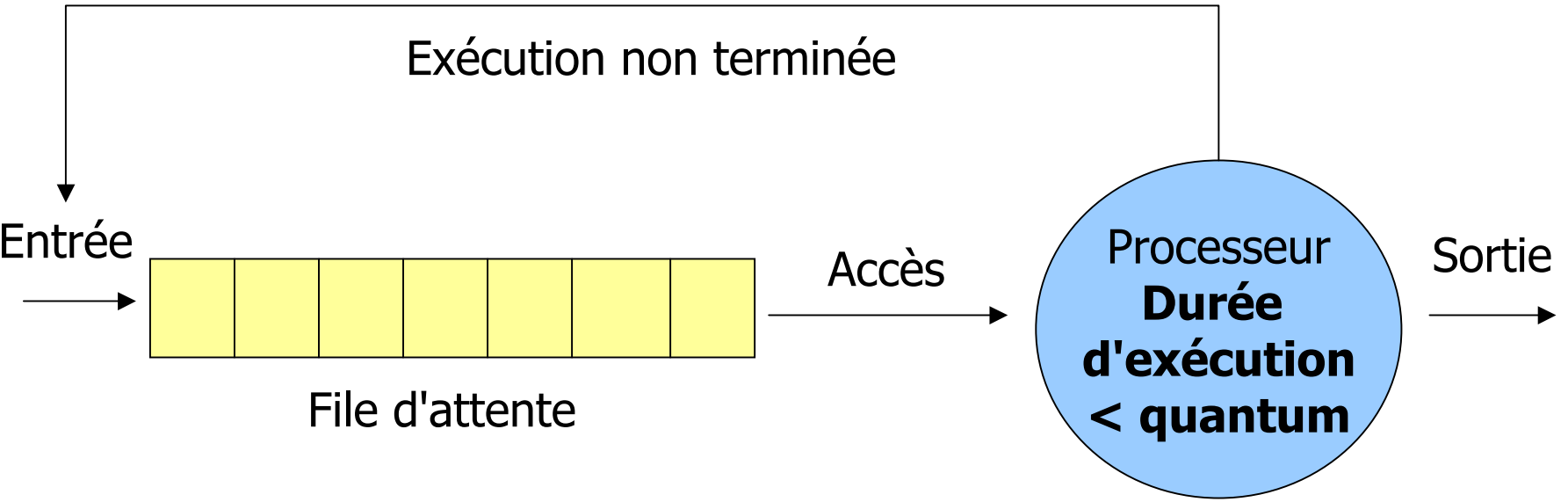
- Les processus font plusieurs passages dans la file d'attente
- La définition du taux de retard reste valable
- Le temps d'attente d'un processusest sa durée d'attente cumulée
- Ce mécanisme est surtout utilisé dans la stratégie du tourniquet

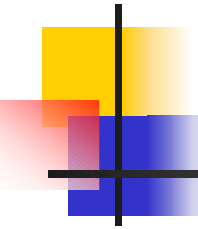
4.3 Politique d'ordonnancement

- **Le tourniquet :** consiste à vider les processus qui s'attardent trop dans le processeur
- Le système passe d'un processus à un autre
- Un processus est remis en file d'attente dès que sa durée d'occupation du processeur dépasse une durée prédéfinie : le **quantum de temps**

4.3 Politique d'ordonnancement

Tourniquet





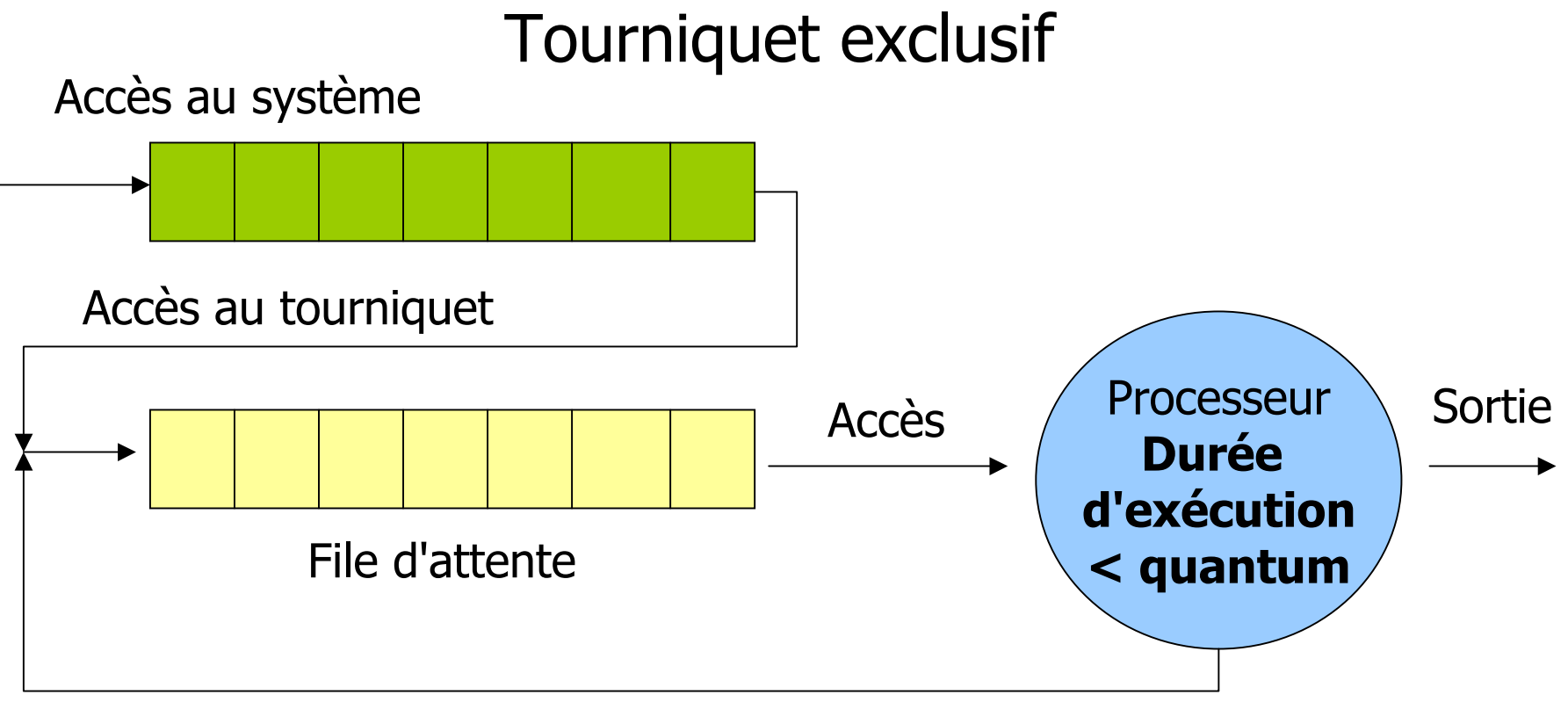
4.3 Politique d'ordonnancement

- La gestion de la file d'attente est faite selon le principe FIFO :
  - Les travaux assez courts sont vite servis
  - Le tourniquet garanti que les travaux longs sortiront du système au bout d'un temps fini
  - Cependant l'efficacité du système dépend de la valeur du quantum
    - Trop petit, la machine perd du temps à changer de contexte (swapping) ce qui la rend lente
    - Trop long, les petits travaux ne sont pas exécutés

4.3 Politique d'ordonnancement

- **Le tourniquet exclusif :** on peut vouloir éviter que les processus long ne s'attardent pas trop dans le système
- On ajoute une file d'entrée au tourniquet
- On distingue alors deux classes de processus :
  - Les **processus acceptés**, qui sont intégrés au tourniquet
  - Les **nouveaux processus** qui attendent dans le première file d'attente

4.3 Politique d'ordonnancement



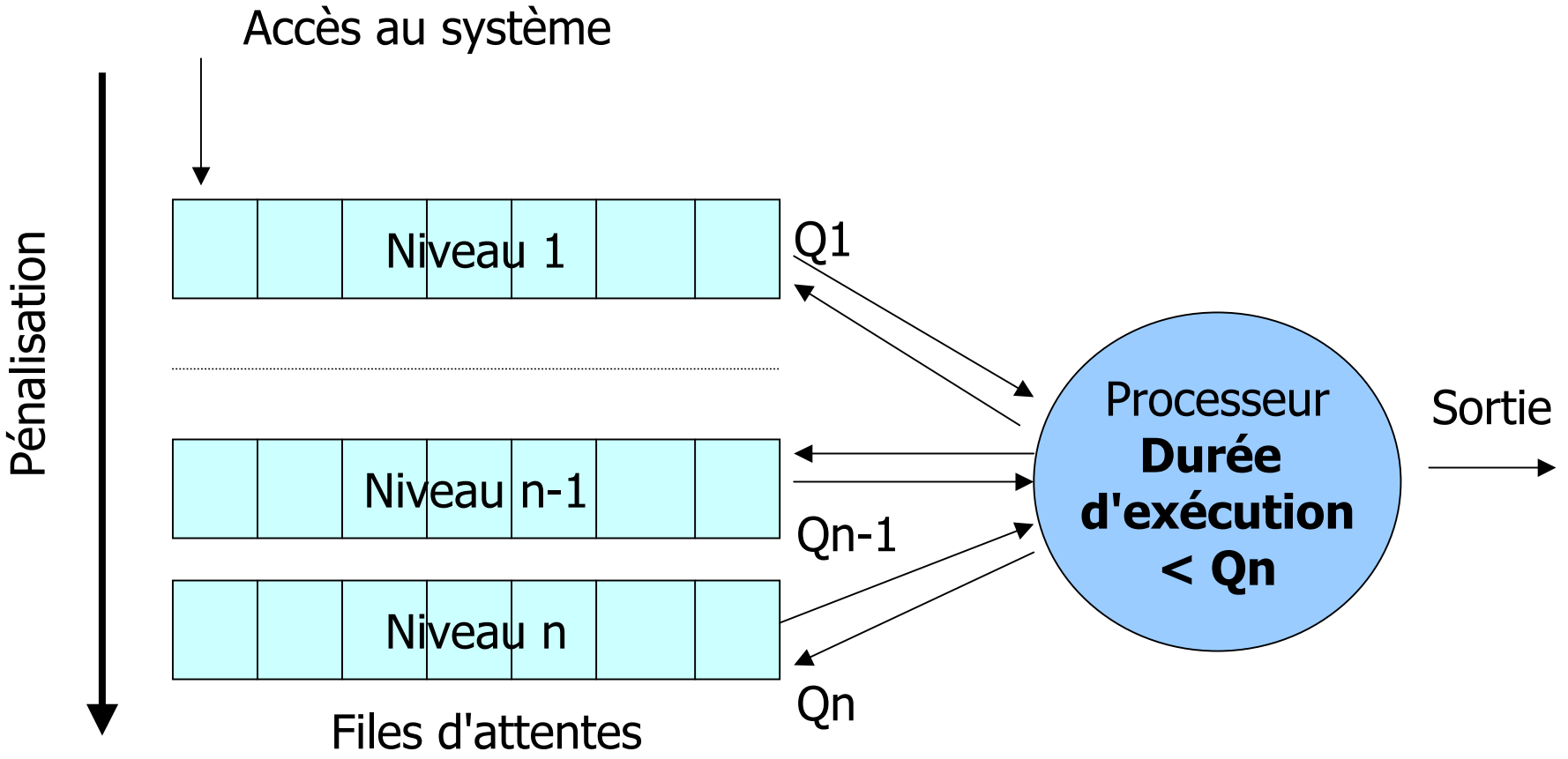
4.3 Politique d'ordonnancement

- Les processus possèdent un taux de priorité qui augmente en fonction du temps qu'ils ont passé dans le système selon des modalités différentes :
  - Le **taux de priorité** d'un nouveau processus est toujours plus faible que le plus bas taux de priorité des processus acceptés
  - Si le **taux de priorité** des nouveaux processus augmente plus rapidement que celui des acceptés il peut y avoir rattrapage
  - Un processus nouveau dont le taux de priorité à atteint celui d'un processus accepté est a son tour **intégré au système**

4.3 Politique d'ordonnancement

- **Tourniquet à plusieurs files :**
  - On travaille avec plusieurs files d'attente
  - Les files d'attente permettent d'introduire une hiérarchie entre les demandeurs
    - Fonction d'une priorité associée aux processus
    - Fonction du temps déjà passé dans le système
    - Etc.

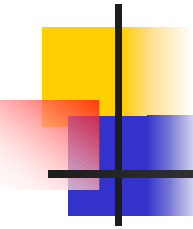
4.3 Politique d'ordonnancement



Tourniquet avec files d'attente multiples

4.3 Politique d'ordonnancement

- Les règles de fonctionnement sont les suivantes :
  - Le niveau de priorité maximal (niveau 1) correspond à la file d'attente des processus arrivants
  - A chaque file d'attente est associé un quantum de temps spécifique
  - La file la moins prioritaire à un quantum de temps important (Qn+1>Qn)
  - Quand un processus atteint la fin de son quantum de temps sans être terminé, il descend d'un étage (dans la file de rang supérieur et de priorité inférieure)



4.3 Politique d'ordonnancement

- Les processus d'un niveau de priorité ne peuvent accéder au processeur que si les niveaux inférieurs sont vides
- L'apparition d'un nouveau processus dans une file de rang inférieur à celui de la file d'origine d'un processus en cours provoque le vidage de ce processus (réquisition du CPU)