Table des matières

Chapitre II : Gestion des processus

Chapitre II : Gestion des processus

Objectif :

Dans ce chapitre nous étudions les principaux algorithmes et mécanismes employés pour l'allocation du processeur aux processus. Ainsi que la synchronisation entre processus et la communication inter-processus.

Eléments de contenu :

- ***Concept de processus***
- ***Algorithmes d'ordonnancement des processus***

I. Processus et contexte d'un processus

Définition :

Un processus représente l'exécution d'un programme comportant des instructions et des données. C'est une entité dynamique crée à un instant donné et qui disparaît, en général, au bout d'un temps fini. Chaque exécution d'un même programme donne lieu à un nouveau processus.

Définition :

Le contexte d'un processus (ou vecteur d'état) est l'ensemble des informations nécessaires pour que le processus puisse reprendre son exécution après une interruption. (Il termine l'exécution du programme et non pas commencer dés le début). Il comprend donc :

- Le mot d'état et les registres généraux (le compteur ordinal, etc.)
- Un espace de travail (segments, procédures, données statiques et dynamiques, pile d'exécution,…)
- Un ensemble d'attributs attachés au processus et spécifiant diverses propriétés (nom, priorité, droits,…) ou encore des informations de gestion (état, lien de chaînage,…)

On peut classer ces informations en deux classes :

- ♦ Des informations utilisées explicitement par le processus (variables, procédures,…)
- ♦ Des informations utilisées par le système pour gérer l'attribution des ressources

II. État d'un processus

Pour qu'un processus puisse s'exécuter, il a besoin de procédures et de données, de mémoire destinée à les contenir, de l'unité centrale, éventuellement de fichiers et de périphériques. Nous appelons toutes ces entités des ressources.

Comme les ressources du système sont en nombre limité, il n'est pas possible d'attribuer à chaque processus, dès sa création toutes les ressources dont il aura besoin. On peut aboutir à la situation ou un processus n'est pas en possession des ressources indisponibles à l'exécution de l'instruction suivante. On dit que le processus est dans l'état bloqué. Par opposition, un processus qui dispose de toutes les ressources dont il a besoin pour évoluer est dit à l'état actif (ou élu ou en exécution). Un processus est à l'état éligible (ou prêt) s'il est en attente de l'unité centrale uniquement.

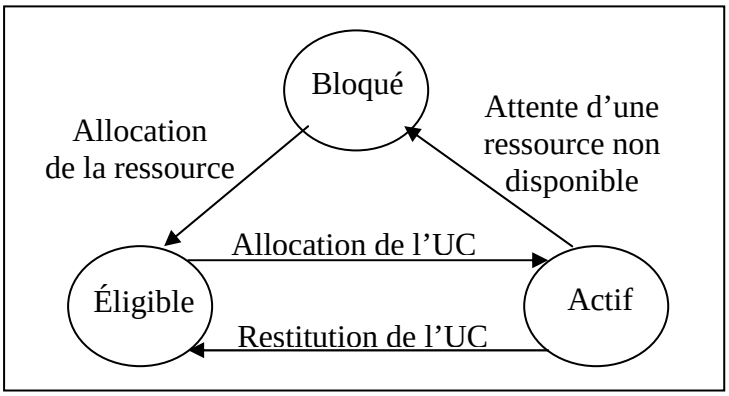

Figure 1. Diagramme d'états d'un processus

Etant une entité nécessaire à l'exécution d'un processus, une ressource peut être matérielle (unité centrale, mémoire centrale, périphériques,…) ou logicielle (variable partagée, fichier,…)

Remarque :

A chaque type de ressource est associé dans le système une procédure d'allocation et à chaque ressource correspond un descripteur. Le descripteur minimal se réduit à un bit représentant l'état libre ou alloué de la ressource.

III. Système d'exploitation multi-tâches et parallélisme

Dans un système d'exploitation multi-tâches, à un instant donné, plusieurs tâches peuvent s'exécuter logiquement en parallèle. Ce parallélisme peut être :

- Réel : si l'ordinateur dispose d'au moins deux processeurs pouvant exécuter simultanément deux activités : système multiprocesseurs.
- Virtuel : pseudo-parallélisme : consistant en le repliage des activités de plusieurs processeurs fictifs sur un seul processeur réel.

Le parallélisme peut être considéré comme l'activation de plusieurs processus. S'il y a autant de processeurs que de processus, le problème est simple, mais

habituellement il y a moins de processeurs que de processus. Si bien que le parallélisme sera obtenu en basculant les processeurs d'un processus à l'autre. Si le basculement est réalisé à des intervalles de temps suffisamment courts, le système donne l'illusion du parallélisme.

IV. Problèmes dues au multi-tâches

Le rôle principal d'un système d'exploitation multi-tâches est de répartir les ressources (en particulier le processeur) entre un certain nombre de processus progressant de manière concurrente. Cela pose un certain nombre de problèmes que l'on peut classer en trois catégories :

- ✓ Problèmes liés à la compétition entre les processus pour l'obtention du processeur. Le système doit donc définir une stratégie d'ordonnancement des processus.
- ✓ Problèmes liés à la coopération entre processus contribuant à une tâche globale. Cette coopération est généralement un échange d'informations ou de signaux entre processus. Ces relations inter-processus doivent être prévues et gérées par le système d'exploitation
- ✓ Problèmes pouvant se produire lorsque plusieurs processus tentent d'utiliser en exclusivité des ressources c'est le problème d'inter-blocage. Le système doit alors adopter une politique de gestion de ressources lui permettant de régler les conflits qui peuvent se présenter.

Ces trois problèmes sont étudiés respectivement par les trois grands chapitres suivants dans ce support de cours.

V. Gestion de ressources

Pour des raisons économiques, les ressources sont dans la majorité des cas, en quantité insuffisante pour répondre aux demandes de pointes. Ils se forment donc des files d'attente de processus demandeurs de ressources.

Le mécanisme général est le suivant :

- ✓ une demande de ressource est adressée au système d'exploitation par un processus
- ✓ si la ressource n'est pas disponible, il y a blocage et mise en file d'attente du processus demandeur
- ✓ sinon l'allocation de la ressource a lieu et le processus demandeur poursuit son déroulement.

Le système d'exploitation reçoit aussi des demandes de restitution de ressources. Le mécanisme est alors le suivant :

- ✓ la ressource libérée est remise dans l'ensemble des ressources disponibles
- ✓ si des processus sont en attente sur la disponibilité de cette ressource, un ou plusieurs d'entre eux seront débloqués

Il est important de distinguer le programme système de sélection du prochain processus à activer du programme système qui provoque la réactivation du processus choisi.

Il est aussi nécessaire d'étudier l'allocation des ressources sous deux aspects : les mécanismes et les politiques. Les mécanismes sont les moyens mis en œuvre pour réaliser l'allocation des ressources (structures de données pour décrire les ressources, les techniques pour assurer l'usage exclusif des ressources critiques, les files d'attente des requêtes qui ne peuvent être servies immédiatement,…). Les politiques gouvernent la façon d'utiliser les mécanismes pour garantir le service des requêtes. Une allocation mal avisée peut entraîner une sur-charge du système sur une classe particulière de ressources ou une impossibilité d'exploitation des processus.

Définition :

Ressource requérable : une ressource est sujette à réquisition (préemption) si le système d'exploitation peut la retirer à un processus alors que celui-ci en a encore besoin.

Exemple :

Ressource requérable : le processeur ; cela entraîne la sauvegarde du vecteur d'état du processus interrompu et la restauration de celui qui est élu : la commutation de contexte.

Ressource non requérable : L'imprimante ; elle ne peut pas être réquisitionnée

Exemple :

Pour illustrer les choix possibles en matière de politiques d'allocation, prenons l'exemple d'une mémoire allouée par zone.
Lorsqu'une zone devient libre l'allocateur essai de satisfaire les demandes mémoire en attente ; il peut selon les objectifs choisir de :

- ✓ examiner ces demandes dans l'ordre de leurs arrivées (FIFO)
- ✓ associer à chaque demande la priorité attachée au processus demandeur
- ✓ satisfaire les demandes correspondantes le mieux à la zone libre
- ✓ satisfaire avec la zone libre le plus grand nombre de demandes possibles
- ✓ satisfaire en premier lieu la demande la plus grosse présente dans la file d'attente.

Pour chaque type de ressources il existe généralement divers algorithmes. Il est difficile de définir le meilleur, car dans l'évaluation de leurs qualités interviennent des facteurs opposés tel que l'utilisation optimale et le coût de l'algorithme.

VI. Protocoles et Mécanismes d'allocation du processeur

C'est le dispatcheur (distributeur ou commutateur) qui alloue l'unité centrale aux différents processus.

A chaque activation du dispatcheur il y aura la séquence suivante :

- Le processeur est il alloué au processus le plus prioritaire ?
- Si oui reprendre le traitement à l'adresse du compteur ordinal
- Sinon
  - sauvegarder le contexte du processus en cours
  - extraire le contexte du processus le plus prioritaire
  - transférer le contrôle à la nouvelle adresse du compteur ordinal

Nous avons vu que dans un système multi-tâches, il y a exécution d'un certain nombre de tâches appartenant au système d'exploitation ou aux utilisateurs. Comme les différentes tâches actives se partagent un seul processeur (ou un nombre limité de processeurs) pour leur exécution, le problème central de la gestion des processus consiste à déterminer à chaque instant quelle tâche doit être allouée au processeur en fonction des critères d'optimisation choisis par le concepteur ou par l'exploitant du système.

Le passage de l'état éligible à l'état actif se fait toujours sous le contrôle du dispatcher dont le rôle est d'attribuer le processeur à un processus éligible après que la tâche en cours d'exécution ait été suspendue ou se soit terminée naturellement. Il se charge de sauvegarder le contexte de la tâche suspendue et d'établir le contexte de la tâche élue.

Les fonctions du scheduler sont :

- ♦ introduction de nouveaux processus
- ♦ attribution des priorités aux processus
- ♦ réalisation des politiques d'allocation des ressources.

Le scheduler a une priorité maximale, il est activé par l'un des événements suivants :

- ✓ une requête de ressource
- ✓ une libération de ressource
- ✓ un achèvement de processus
- ✓ une introduction de processus

VI.1. Stratégies

Un bon algorithme d'ordonnancement doit satisfaire les critères suivants :

- ✓ Chaque processus prend sa part légale de la CPU
- ✓ Rendre la CPU très bien exploité
- ✓ Minimiser les temps de réponse des programmes interactifs
- ✓ Minimiser les temps de réponse des programmes en batch

✓ Maximiser le nombre de programmes traités par heure.

VI.1.1. Stratégie sans réquisition

L'approche d'allocation la plus simple consiste à laisser le processeur au processus actif jusqu'à ce que ce dernier termine naturellement son exécution ou se trouve bloqué par manque de ressource.

- FIFO (First In First Out) : le premier arrivé premier servi

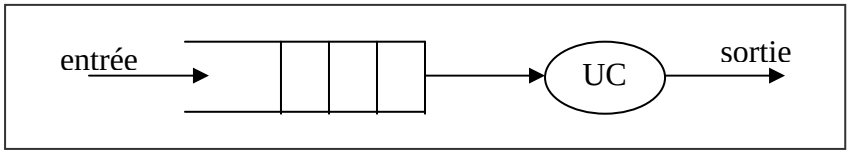

Figure 2. ordonnancement FIFO

L'unité centrale est attribuée au premier processus arrivé.

- SJF (Short Job First) : le plus court d'abord, Priorité

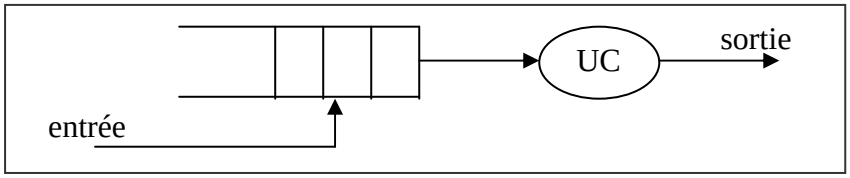

Figure 3. ordonnancement SJF

L'unité centrale est attribuée d'abord au processus demandant moins de temps d'exécution ou le plus prioritaire. L'idée de cette méthode est de réduire le temps moyen d'exécution des programmes et ce en exécutant les petits programmes (exigeant le temps CPU le plus petit) avant les grands

Ces méthodes amènent à des temps de réponse inacceptables dans le contexte de système multi-tâches on préfère mettre en œuvre une stratégie d'allocation par réquisition dans laquelle le système peut retirer le processeur à un processus pour l'attribuer à un autre processus plus urgent.

VI.1.2. Stratégie avec réquisition

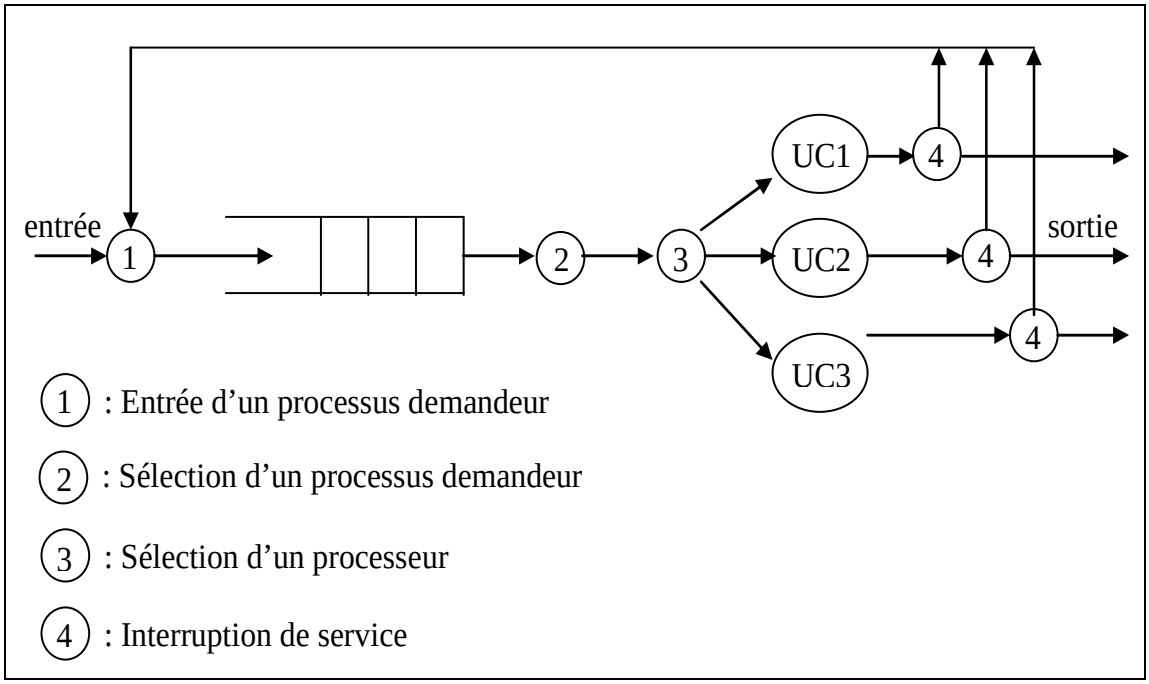

Figure 4. ordonnancement avec réquisition

Nous allons nous intéresser aux stratégies possibles pour la phase (2) : sélection du processus demandeur.

Remarque :

La réquisition accentue encore l'avantage donné aux travaux courts par contre à chaque interruption il y a perte de temps pour la gestion des vecteurs d'état.

- Round Robin : balayage cyclique

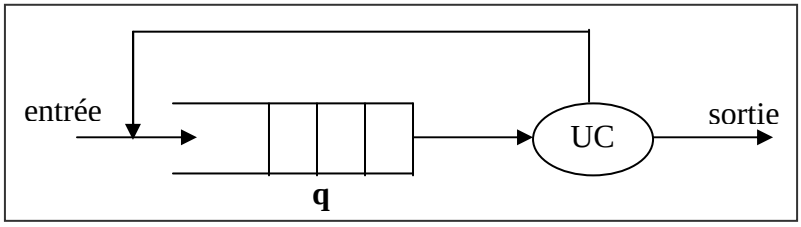

Figure 5. ordonnancement Round Robin

L'idée de cet algorithme est très simple; en effet elle consiste à attribuer à chaque processus un quantum de temps pendant lequel celui-ci s'exécute sur la CPU ; une fois le quantum est épuisé la main sera donnée à un autre processus, puis éventuellement un autre puis à nouveau au premier de façon cyclique jusqu'à l'achèvement de tous les processus.

C'est la stratégie FIFO avec réquisition. L'intervalle de temps q est le paramètre de cette stratégie : le quantum

- Recyclage à plusieurs files d'attente

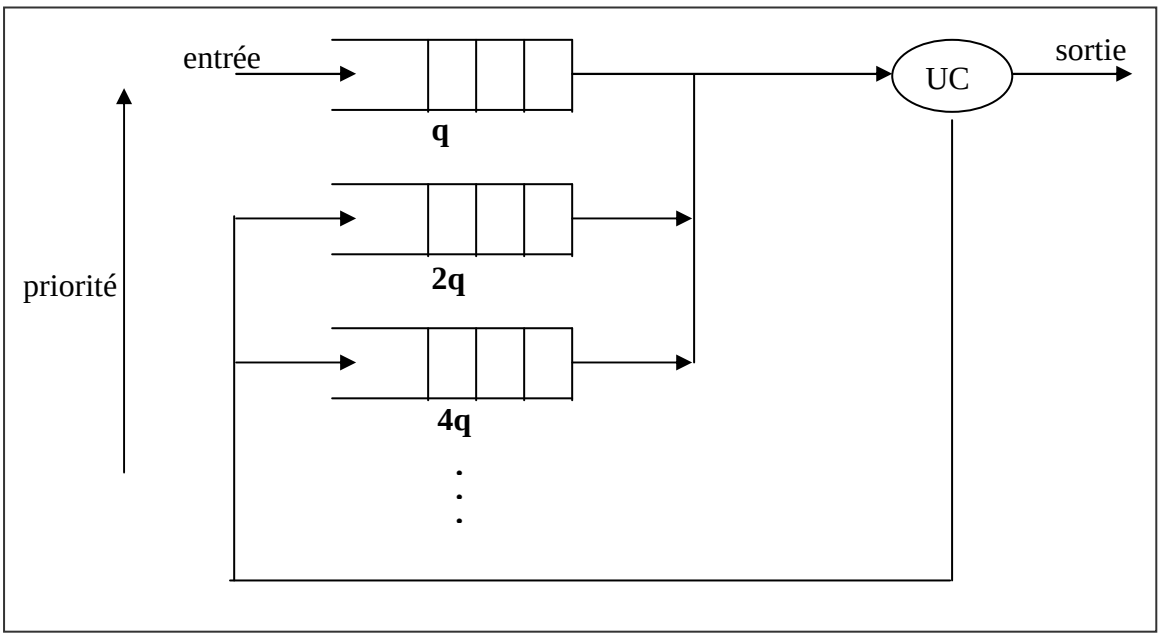

Figure 6. ordonnancement de type balayage cyclique à plusieurs files d'attente

Un processus est tout d'abord placé dans la première file d'attente (où le quantum = q), puis s'il obtient son quota de traitement, il passe dans la deuxième file d'attente (où le quantum = 2q) puis dans la file d'attente suivante jusqu'à ce qu'il termine son exécution. Le quantum de temps d'une file est le double de celui de sa précédente. Le but est de favoriser les processus interactifs qui réclament généralement peu de temps processeur. L'extraction d'un processus d'une file d'attente se fait à partir de la file non vide de plus haute priorité. Tout processus interrompu est replacé dans la file d'attente de priorité inférieure à celle dont il avait été extrait.

L'idée est de réduire les opérations de swapping lors de l'exécution d'un processus. La solution consiste à instaurer la notion de classe de priorité de la façon suivante : si un processus par exemple exige 100 quantum pour son exécution, alors il suffit de lui allouer la CPU la première fois un seul quantum puis la deuxième fois 2 quantum puis la troisième fois 4 quantum, …8 quantum …16 quantum… 32 quantum, 64 quantum pendant ce dernier tour, il ne va utiliser que 37 quantum.

- Stratégies fondées sur la priorité

L'algorithme de Round Robin considère tous les processus ayant la même priorité. Généralement les avis sont partagés à cet égard, d'ou la nécessité d'instaurer la notion de priorité différente entre les différents processus. Les SE Unix par exemple offre la possibilité à l'utilisateur de modifier la priorité d'exécution de son programme. Dans ce cas étant donné qu'à chaque processus correspond une priorité, l'exécution d'un processus d'une priorité inférieur ne peut être exécuté que s'il n'y a

aucun processus de priorité supérieur en attente. On parle dans ce cas de l'ordonnancement par classe de priorité et on continue à utiliser Round Robin.

La priorité est un nombre attaché à chaque processus et définissant le degrés d'urgence attribué à son exécution. Une stratégie fondée sur la priorité peut être mise en œuvre soit en constituant une file unique triée par priorité soit par plusieurs files d'attente de priorités différentes. Une file n'étant servie que lorsque les files de plus fortes priorités sont vides. La priorité peut être statique ou dynamique. Si pendant l'exécution d'un processus, un autre plus prioritaire est lancé. Ce dernier interrompe le premier et prend l'unité centrale pour lui.

- Stratégie fondée sur le temps d'exécution (SJF+réquisition)

L'unité centrale est attribuée au processus le plus court. Mais si pendant l'exécution de ce dernier, un autre processus demande un temps d'exécution plus court que ce qui reste au processus courant, il aura l'unité centrale.

- Stratégie fondée sur les échéances (pour les systèmes temps réel)

Dans le cas où l'on a des contraintes strictes de temps à respecter, l'unité centrale peut être allouée au processus le plus urgent (celui qui doit se terminer le premier).

Exemple :

DEC SYSTEM 10 possède trois files d'exploitation gérées en FIFO

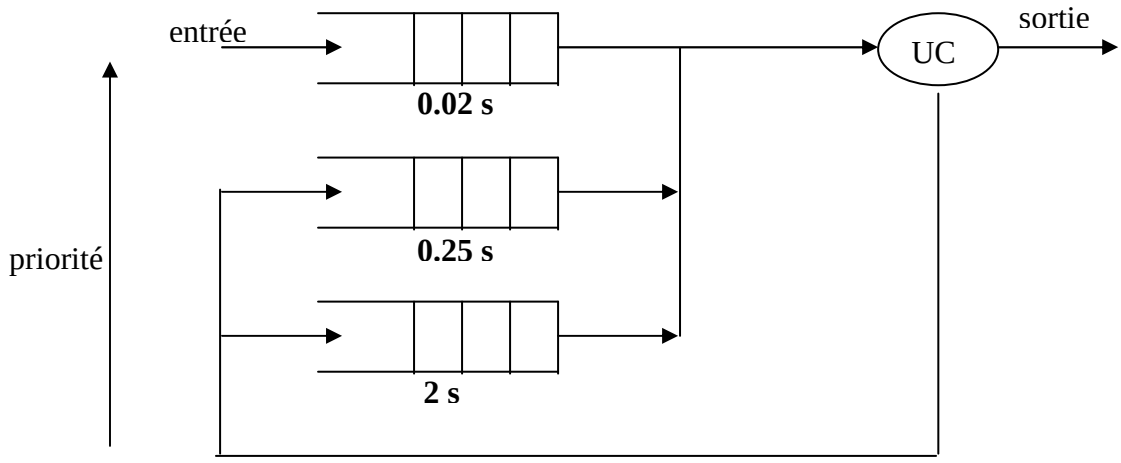

Figure 7. exemple d'ordonnancement à plusieurs files d'attente

Un nouveau processus est mis dans la file des 0.02s puis s'il obtient son quota de traitement, il passe dans la file des 0.25s et enfin dans la file des 2s. L'extraction d'un processus d'une file d'attente se fait à partir de la file non vide de plus haute priorité. Tout processus interrompu est replacé dans la file d'attente de priorité inférieure à celle dont il avait été extrait.

VI.2. Mécanismes

Chaque processus est représenté par un descripteur qui rassemble les informations essentielles le concernant :

- Pointeur suivant : pointeur permettant le chaînage des descripteurs des processus
- Contexte : champs de sauvegarde des informations nécessaires à la reprise de l'exécution d'une tâche provisoirement suspendue (identification du processus, état du processus, informations de supervision)

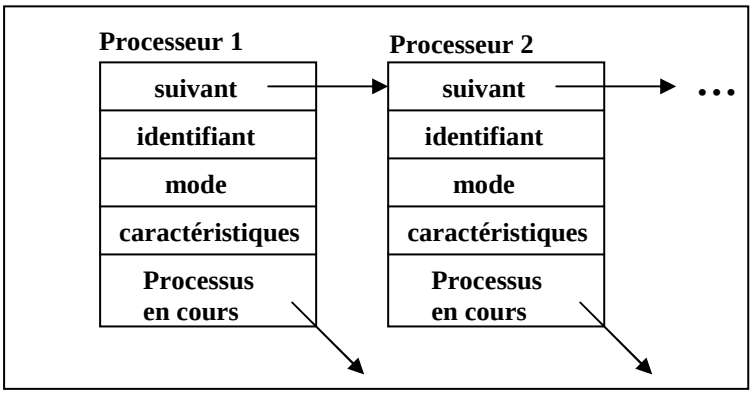

Figure 8. Descripteur d'un processeur

Le système gère les processus par l'intermédiaire de leurs descripteurs qu'il organise en files d'attente au niveau de chaque état.

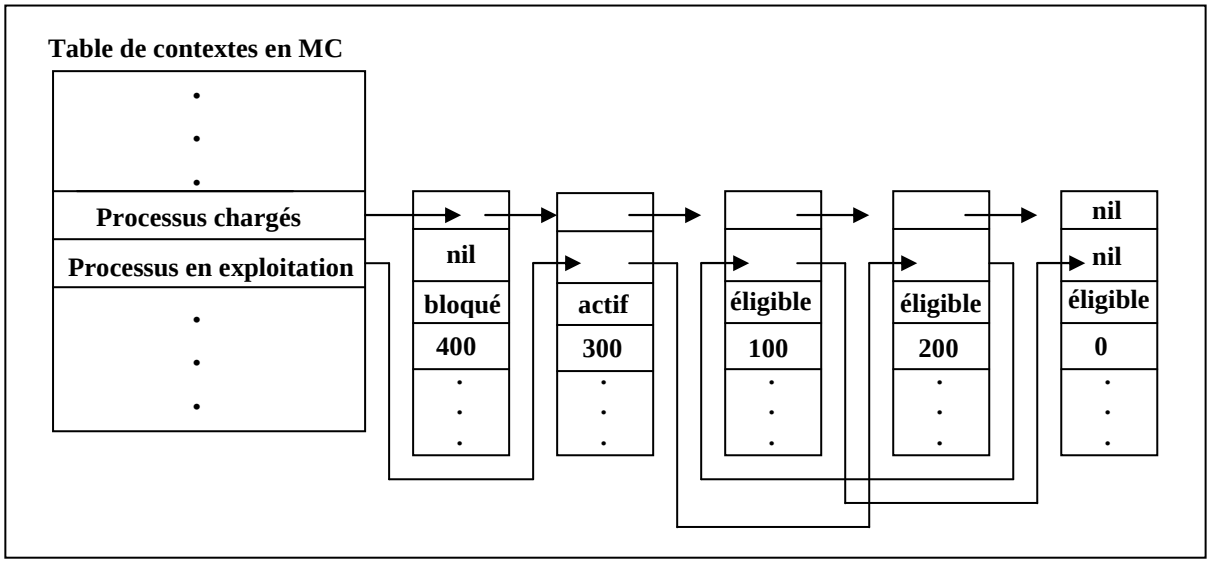

Figure 9. file d'exploitation

La liste des processus chargés contient les descripteurs de tous les processus vivants dans le système. Alors que la file d'exploitation contient seulement les processus à l'état actif ou éligible triés par ordre de priorité décroissante. Il est possible qu'à un instant donné, la file d'exploitation soit vide. Plutôt que de faire boucler le processeur en attente de travail dans le dispatcher on introduit un processus supplémentaire NULL de priorité minimale et qui soit toujours exécutable (boucle vide).

Lorsqu'une ressource devient libre et qu'un processeur passe à l'état éligible, le scheduler doit :

- ✓ modifier l'état du descripteur du processus
- ✓ relier le descripteur dans la file d'exploitation
- ✓ l'affectation du processeur est reconsidérée à chaque évènement : fin d'E/S, fin du quantum de temps, libération de ressource, création d'un nouveau processus, terminaison d'un processus,…

Exercice1 :

Soit la séquence de processus suivante :

N° processus	Date d'arrivée	Temps d'exécution
1	08 :00	5
2	08 :00	4
3	08 :02	3
4	08 :00	6
5	08 :05	2

Les dates d'arrivée sont exprimées en heures : minutes. Les temps d'exécution sont exprimés en minutes.

Donnez le diagramme de GANTT et calculez le temps global moyen et le temps d'attente moyen dans le cas où l'on utilise les algorithmes d'ordonnancement suivants : SJF, FCFS (FIFO), ROUND ROBIN (avec q= 1mn), par priorité (sachant que les priorités respectives des processus sont 3, 2, 2, 3, 4 et que 4 est la priorité la plus haute).

Bibliographie

- ✓ [TANUMBAUM] Systèmes d'Exploitation : Système Centralisés, Systèmes Distribués, édition
- ✓ [KRAKOWIAK] principes des système d'exploitation des ordinateurs, édition DUNOD
- ✓ [CROCUS] Systèmes d'exploitation des Ordinateurs : principes de conception, édition DUNOD
- ✓ [LISTER] principes fondamentaux des systèmes d'exploitation, édition EYROLLES
- ✓ [LITERMITTE] les systèmes d'exploitation, structure et concepts fondamentaux
- ✓ [BEAUQUIER & BERARD] Systèmes d'Exploitation : Concepts et algorithmes