Ordonnancement

Introduction

Un ordonnanceur (scheduler) comporte deux parties :

- ordonnanceur de haut niveau (Long Term Scheduling): politique de décision, affectation de priorité
- ordonnanceur de bas niveau ou distributeur (Short Term Scheduling ou dispatcher) : affectation d'un processeur à un processus

Pour l'ordonnanceur de haut niveau, trois concepts sont à détailler :

- Mode de décision

  Il s'agit de spécifier des instants où les priorités des processus sont évaluées et comparées et où des processus sont sélectionnés pour exécution. Entre deux instants consécutifs, l'allocation de processeurs aux processus ne change pas.

  - mode **non pré-emptif** : un processus en exécution continue jusqu'à ce qu'il se termine ou se bloque (non adéquat pour les systèmes temps réel et temps partagé)

  - mode **pré-emptif** : un processus en exécution peut être interrompu par diverses causes : un nouveau processus arrive, un processus existant est réveillé, un temps q s'est écoulé, la priorité d'un processus prêt est devenue plus grande que celle du processus actif, ...

  compromis : la pré-emption sélective. Elle consiste à assigner à chaque processus p une paire de bits (u_p, v_p) :

  u_p = 1 si p peut pré-empter un autre processus et 0 dans le cas contraire
  (u_p représente la capacité de préemption d'un processus)

  v_p = 1 si p peut être pré-empté par un autre processus et 0 dans le cas contraire
  (v_p représente une priorité d'exécution).

- Fonction de priorité

  A tout moment une priorité est affectée à un processus du fait du résultat de l'application d'une fonction de priorité : ses variables sont divers paramètres (liés au processus ou au système) et la valeur retour est la priorité . Les paramètres utilisés sont usuellement :

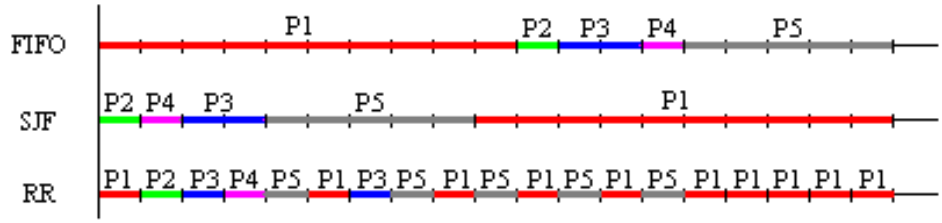
  - besoins mémoire
  - temps d'utilisation CPU
  - temps réel dans le système (temps d'utilisation CPU + temps d'attente)
  - temps total de service

Ordonnancement

Solution des exercices

Solution de l'exercice 1

Le schéma ci-dessous décrit l'enchaînement des processus :

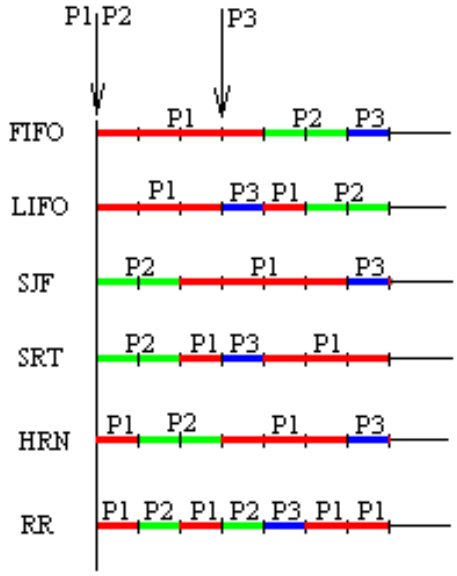


	P1	P2	P3	P4	P5	tot.	moy.
FIFO	10	11	13	14	19	67	13,4
SJF	19	1	4	2	9	35	7
RR	19	2	7	4	14	46	9,2

Le tableau, ci-dessus, indique les temps de présence, dans le système, des processus. La dernière colonne décrit le temps moyen passé par chaque processus. Il est clair que, sur cet exemple, la stratégie SJF est la meilleure.

Solution de l'exercice 2

Comme pour l'exercice précédent, on trouve ci-dessous la chronologie des événements :

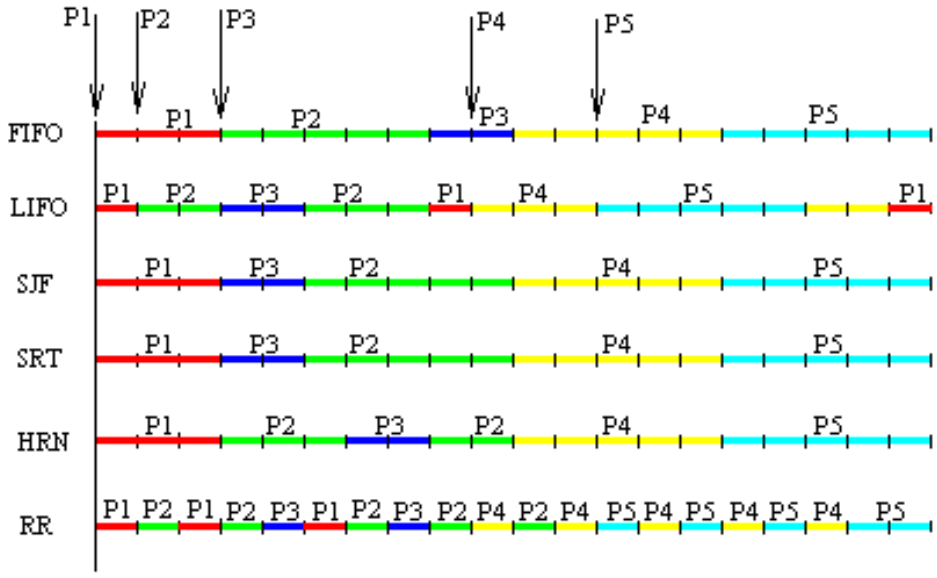


	P1	P2	P3	total	moyenne
FIFO	4	6	4	14	4,66
LIFO	4	7	1	12	4
SJF	6	2	4	12	4
SRT	7	2	1	10	3,33
HRN	6	3	4	13	4,33
RR	7	4	2	13	4,33

Les résultats ci-dessus indiquent que SRT est, pour cet exemple, la meilleure politique.

Solution de l'exercice 3

On reprendra les politiques de l'exemple précédent et les résultats chronologiques et synthétiques sont consignés dans le tableau ci-dessous :



	P1	P2	P3	P4	P5	total	moyenne
FIFO	3	7	7	6	8	31	6,2
LIFO	20	7	2	10	5	44	8,8
SJF	3	9	2	6	8	28	5,6
SRT	3	9	2	6	8	28	5,6
HRN	3	9	5	6	8	31	6,2
RR	6	10	5	9	8	38	7,6

Les deux politiques les plus intéressantes sont ici SJF et SRT.

Solution de l'exercice 4

Rappelons que chaque estimation est calculée à partir de l'estimation précédente et de la durée effective :

$$S_{n+1}=aT_n+(1-a)S_n$$

Le tableau suivant donne les durées estimées et réelles des bursts :

étape n	T_n	a = 0,8		a = 0,5	
		S_n	S_{n+1}	S_n	S_{n+1}
0		10	2	10	5
1	6	2	5,2	5	5,5
2	4	5,2	4,24	5,5	4,75
3	6	4,24	5,65	4,75	5,38
4	4	5,65	5,93	5,38	4,69
5	13	5,93	11,59	4,69	8,84
6	13	11,59	12,72	8,84	10,92
7	13	12,72	12,94	10,92	11,96

Le calcul des écarts entre les prévisions et la réalité donne un écart type de 3,28 pour a=0,8 et 3,73 pour a=0,5.

Solution de l'exercice 5

On peut résumer dans un tableau la progression des deux processus P1 et P2 :

processus	temps de service atteint a	temps réel r	Priorité
P1 P2	2 3	2 5	1 4,5
P1 P2	2 4	3 6	2 3,67
P1 P2	2 5	4 7	3 6,16

P1 P2	2 6	5 8	4 7
P1 P2	2 7	6 9	5 7,83
P1 P2	2 8	7 10	6 8,66

etc.....

P1 P2	2 20	20 23	19 19,5
P1 P2	2 21	21 24	20 20,33
P1 P2	2 23	22 25	21 21,16
P1 P2	2 24	23 26	22 22

On constate que le processus P2, bénéficiant d'une plus grande priorité, obtient le premier le processeur. Cette situation se poursuit jusqu'à ce que la priorité de P1 rattrape celle de P2, c'est à dire 21 unités de temps après le départ de l'observation.

Ordonnancement

Exercices

Exercice 1

5 processus, P1, P2, P3, P4, P5 sont dans une file d'attente dans cet ordre (P1 est le premier, P5 est le dernier). Leur exécution demande un temps total de service exprimé en unités arbitraires :

processus	P1	P2	P3	P4	P5
temps de service estimé	10	1	2	1	5

1) Décrire l'exécution des processus dans le cadre des politiques d'ordonnancement FIFO, SJF, RR (avec un quantum de 1).

2) Quelle est ,de ces trois politiques, celle qui correspond à un temps minimal d'attente moyen par processus ?

Exercice 2

Trois processus P1, P2, P3 ont été chargés sur un système informatique aux dates indiquées ci-dessous ; leur demande en durée de service est également indiquée (unités de temps arbitraires).

processus	date arrivée	durée de service
P1	0	4
P2	0	2
P3	3	1

On considère les politiques d'ordonnancement suivantes : FIFO, LIFO, SJF, SRT, HRN, RR . Pour les politiques pré-emptives, le quantum est égal à 1 unité de temps.

Comparer ces diverses politiques.

Exercice 3

On considère l'ensemble suivant de processus :

processus	date d'arrivée	temps de service
1	0	3
2	1	5
3	3	2
4	9	5
5	12	5

Comparer, sur cet exemple, les diverses politiques d'ordonnancement.

Exercice 4

Un processus interactif effectue les "bursts" successifs de durées 6, 4, 6, 4, 13, 13, 13. La durée initiale estimée d'un burst est 10.

Comparer l'estimation et la réalité dans le modèle de la moyenne pondérée exponentielle (on prendra a = 0.8 et a = 0.5).

Exercice 5

Soit deux processus p1 et p2 dans un système utilisant la politique d'ordonnancement PDS. La table suivante donne pour chaque processus le temps a de service atteint, le temps réel r passé dans le système et la fonction $f^{-1}(a)$,à un instant donné :

processus	a	r	$f^{-1}(a)$
p1	2	2	a/2
p2	3	5	a/6

1) Quel est le processus qui obtient à cet instant le processeur ?

2) Après quelle période de temps les priorités des deux processus seront égales ?

- priorités externes
- adaptation temporelle
- charge du système

Le temps total de service peut être estimé dans quelques systèmes. La priorité affectée est alors d'autant plus grande que le temps total de service est court :

- un processus "court" devrait être terminé rapidement
- traiter d'abord les travaux courts réduit le temps total d'exécution de tous les processus

Les priorités externes sont utilisées pour différencier les classes d'utilisateurs et de processus

exemple :

processus temps réel : hautes priorités
processus interactifs : moyennes priorités
processus batch : basses priorités

L'adaptation temporelle prend en compte le fait que l'urgence à terminer une tâche peut varier dans le temps.

exemple : accroître la priorité au fur et à mesure que le temps du processus dans le système augmente
(problème des prédictions météo : elles ne peuvent arriver après la date correspondante).

- Règle d'arbitrage

Cette règle est faite pour les conflits entre processus de même priorité. Plusieurs possibilités choix au hasard, choix suivant la position dans la file, ...

Principaux algorithmes

Examinons maintenant quelques algorithmes d'ordonnancement

FIFO (First In/ First Out)

L'ordre est celui de l'arrivée des processus dans la file des processus prêts. La fonction de priorité est P(r) = r où r est le temps réel passé dans le système. Le mode de décision est non pré-emptif; la règle d'arbitrage est le choix au hasard pour des processus qui arrivent au même moment.

LIFO (Last In / First Out)

L'ordre est l'inverse du précédent : le dernier arrivé est le premier servi. La fonction de priorité est P(r) = -r où r représente toujours le temps réel passé dans le système.Le mode de décision et la règle d'arbitrage sont comme dans FIFO.

SJF (Shortest Job First)

On suppose que le temps total de service t est connu à l'avance (bien adapté aux processus batch où l'utilisateur peut indiquer un temps max). La fonction de priorité est P(t) = -t. Le mode de décision est non pré-emptif. La règle d'arbitrage est la chronologie ou le choix au hasard.

La difficulté est évidemment l'estimation du temps de service. Ceci peut s'opérer, par exemple, de la manière suivante, pour un processus interactif qui s'exécute comme une série de "bursts" de durée d'exécution réelle T_i. Au n+1 ème burst, on peut estimer le temps de service par la moyenne :

$$S_{n+1} = \frac{1}{n}\sum_{i=1}^{n} T_i$$

Cette relation peut être ré-écrite sous la forme

$$S_{n+1} = \frac{1}{n} T_n + \frac{n-1}{n} S_n$$

En fait, dans la pratique, on généralise cette relation en employant des facteurs de pondération a (compris entre 0 et 1) et 1-a :

$$S_{n+1} = a\ T_n + (1-a) S_n$$

Cette méthode s'appelle la pondération exponentielle.

SRT (Shortest Remaining Time)

Version pré-emptive du SJF. Si a est le temps CPU consommé et t le temps total estimé de service, la fonction de priorité est P(a,t) = a-t. La pré-emption a lieu à l'arrivée du nouveau processus dans la file des prêts.

HRN (Highest Ratio Next)

Cet algorithme permet de prendre en compte l'attente d'un processus en basant la priorité sur le rapport :

$$P(w,t) = (w+t)/t$$

où w est le temps passé à attendre et t le temps total estimé de service. Un processus qui n'attend pas possède un rapport de 1 (cas notamment d'un processus qui vient d'entrer dans le système) ; un processus qui attend longtemps possède un rapport R bien plus grand que 1.

RR (Round Robin)

Un quantum de temps Q est imposé. Lorsqu'un processus actif a consommé son quantum Q, il doit quitter le processeur et est placé en queue d'une file. La règle d'arbitrage est donc cyclique. Tous les processus ont par contre la même priorité. Notons que le quantum Q peut être constant dans le temps ou varier avec la charge du système : soit T le temps nécessaire à un cycle de tous les processus ; on peut poser :

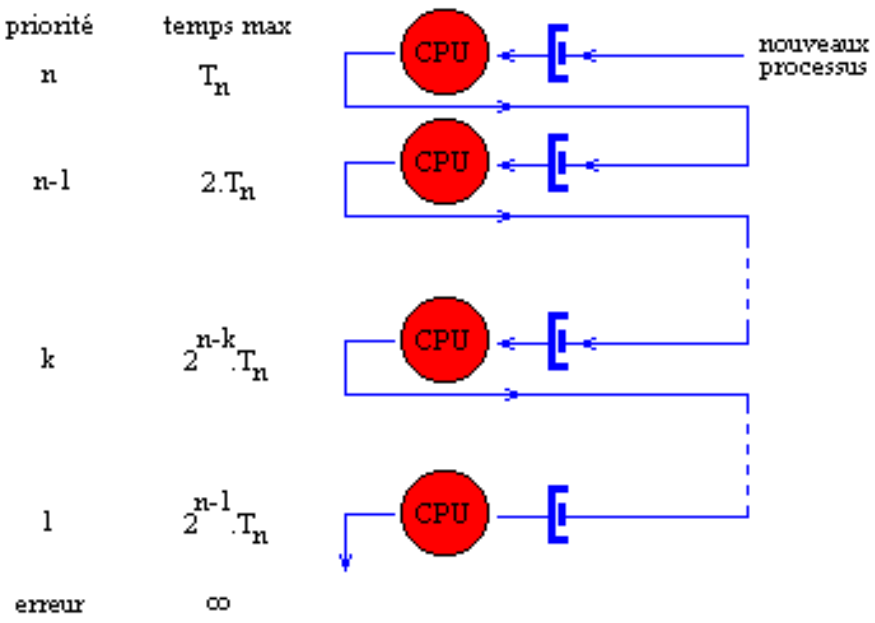
Q = T/n où n est le nombre de processus actifs pendant T unités de temps.

Très utilisé dans les systèmes en temps partagé.

MLF (MultiLevel Feed-back)

Il est défini n niveaux de priorité. Pour chaque niveau i est défini un temps T_i qui est le temps maximum d'utilisation du processeur au niveau i.S'il y a dépassement, la priorité du processus est décrémentée de 1 et celui-ci passe donc au niveau de priorité immédiatement inférieur. Le temps T_i de chaque niveau peut être défini par :

$$T_i = 2^{n-i}T_n$$



T_n est le temps maximum d'utilisation du processeur au niveau n ; $T_k = 2^{n-k}.T_n$ est le temps maximum d'utilisation du processeur au niveau k.

La règle d'arbitrage peut être chronologique (type FIFO) ou cyclique (type RR). Le temps total consommé CPU pour un processus au niveau k (0 = k < n) est a tel que :

$$T_n \cdot \sum_{i=0}^{n-k-1} 2^i <= a < T_n \cdot \sum_{i=0}^{n-k} 2^i$$

En utilisant la relation

$$2^{x+1} - 1 = \sum_{i=0}^{x} 2^i$$

on obtient

$$T_n \cdot (2^{n-k} - 1) \leq a < T_n \cdot (2^{n-k-1} - 1)$$

Cette relation permet de déterminer la priorité du processus, sachant qu'il a atteint un temps de service a . P(a) = k et, en posant i = n-k, P(a) = n - i où i est déterminé par :

$$T_n \cdot (2^{n-k} - 1) \leq a < T_n \cdot (2^{n-k-1} - 1)$$

On constate que :

- si a < T_n alors i = 0 et P(a) = n, plus haute priorité
- si a > T_n.(2^n - 1), alors P(a) < 1 ce qui est interprété comme une erreur.

PDS (Policy-Driven Scheduling)

Cette méthode est adaptative en ce sens que l'affectation des priorités dépend de la charge du système. Elle est basée sur l'utilisation d'une fonction f(r), appelée fonction politique, qui prescrit la corrélation entre le temps réel passé par le processus dans le système et le montant des services que le processus devrait avoir à cet instant. On peut mesurer ce montant de service par le temps pendant lequel le processus a utilisé des ressources; plus précisément, à l'instant r, le montant de service s(r) reçu par un processus est de la forme :

$$s(r) = \sum_{j=1}^{n} w_j . t_j(r)$$

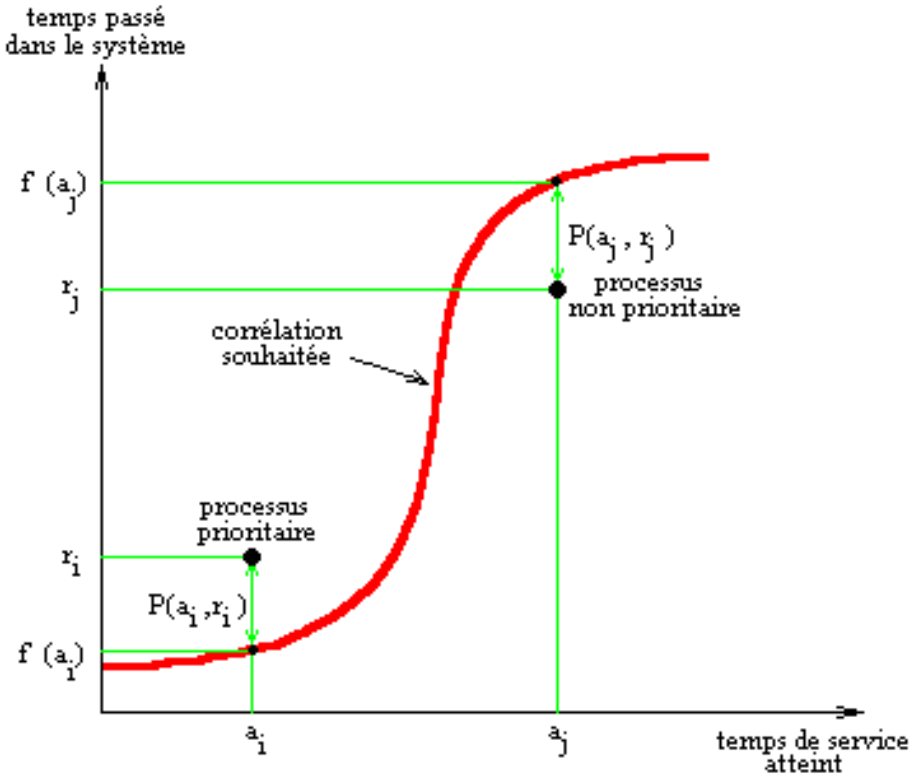
$t_j(r)$ est la durée d'utilisation de la ressource j à l'instant r ; $w_j$est le poids associé à la ressource j et n est le nombre de ressources distinctes gérées par le scheduler..

Pour simplifier ici la méthode PDS, on ne prendra en considération qu'une seule ressource, le processeur; donc s(r) = a. Dans un diagramme (a, r), un point de coordonnées (a_i , r_i) exprime la relation entre le temps passé dans le système et le temps "utile" d'utilisation du processeur.

Un point ($a_i = f(ri)$, $r'_i = f^{-1}(a_i)$) exprime ce qui est souhaité. Pour un temps de service ai on peut donc mesurer la différence $r_i - f^{-1}(a_i)$ entre ce qui est réel et ce qui est souhaité. On pourra alors prendre comme priorité $P(a_i , r_i)$ cette différence

$$P(a_i , r_i) = r_i - f^{-1}(a_i)$$

La priorité s'accroît lorsque le temps passe et que le processus ne reçoit pas les services souhaités. En fait, lorsque P est positive, cela signifie que le service accordé n'est pas à la hauteur des espérances, tandis que si P est négatif, cela signifie que l'on prend "de l'avance" et, dans ce cas, on pourrait stopper le processus pour laisser la place à des processus moins favorisés.



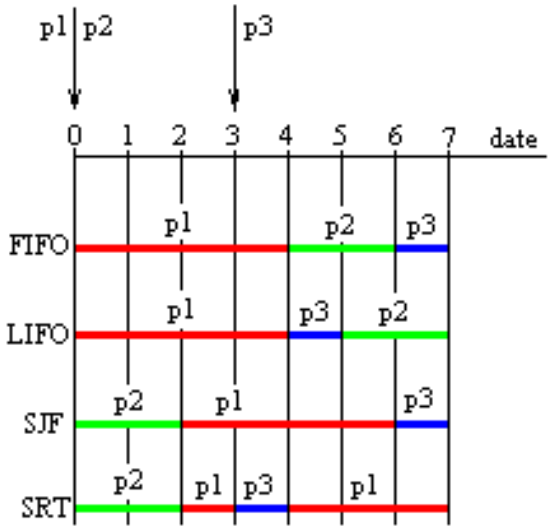
Parmi les exemples précédents, représentatifs des algorithmes d'ordonnancement des divers systèmes d'exploitation, il paraît inévitable de rechercher lequel est le meilleur. La réponse à cette question n'est pas simple; tout dépend du critère de performance retenu, mais aussi du mode d'utilisation (batch, conversationnel, temps partagé, temps réel, ...)

Les 5 premiers algorithmes ont été définis, à l'origine, pour des systèmes fonctionnant en traitement par lot. On peut donc tenter de les comparer entre eux en prenant un exemple simple : 3 processus sont présents dans le système, ils diffèrent par leur temps d'arrivée et le temps de service demandé :

processus	date d'arrivée	temps de service
p1	0	4
p2	0	2
p3	3	1

On considérera que le quantum de temps est de 1 unité

Résultats :



calcul des temps				
	p1	p2	p3	moyenne
FIFO	0+4	4+2	3+1	4,66
LIFO	0+4	5+2	1+1	4,33
SJF	2+4	0+2	3+1	4,00
SRT	3+4	0+2	0+1	3,33
x+y = temps d'attente + temps d'exécution				

Contrairement à la situation précédente, dans les systèmes à temps partagé, le critère de sélection est le temps de réponse ce qui implique l'utilisation d'un algorithme pré-emptif : SRT, RR, MLF, PD.

- RR : le plus simple à implémenter, mais le choix du quantum Q reste délicat
- MLF : permet de distinguer entre plusieurs groupes de processus (une priorité par groupe)
- PD : plus adaptative, mais plus délicate à mettre en oeuvre.