Ordonnancement

Thomas Lavergne
lavergne@lisn.fr

Multiprogrammation et temps partagé

- Plusieurs processus veulent la même ressource

Problématique

Multiprogrammation et temps partagé

- Plusieurs processus veulent la même ressource
- File d'attente de PCB

Multiprogrammation et temps partagé

- Plusieurs processus veulent la même ressource
- File d'attente de PCB

Ordonnancement

Choisir un processus parmi tous les processus dans une liste d'attente pour une ressource.

Multiprogrammation et temps partagé

- Plusieurs processus veulent la même ressource
- File d'attente de PCB

Ordonnancement

Choisir un processus parmi tous les processus dans une liste d'attente pour une ressource.

Exemple : Processeur

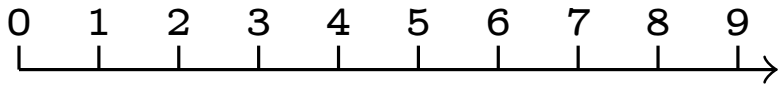
Choisir parmi les processus prêt et en exécution

Diagramme de Gantt

Représentation temporelle de l'allocation de ressources.

Diagramme de Gantt

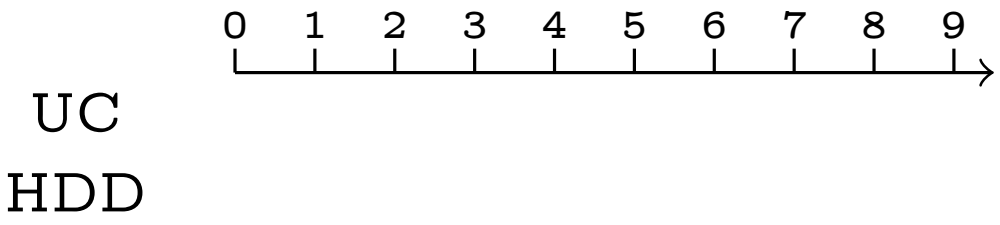
Représentation temporelle de l'allocation de ressources.



Proc.	date	Durée
P1	0	1
P2	1	2
P3	3	4

Diagramme de Gantt

Représentation temporelle de l'allocation de ressources.



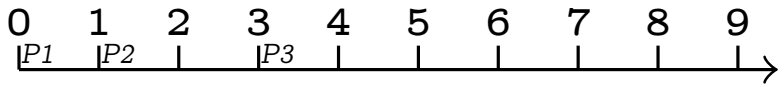
Proc.	date	Durée
P1	0	1
P2	1	2
P3	3	4

Diagramme de Gantt

Représentation temporelle de l'allocation de ressources.

UC

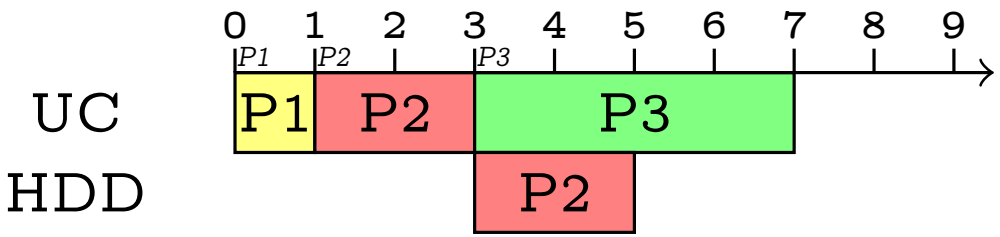
HDD



Proc.	date	Durée
P1	0	1
P2	1	2
P3	3	4

Diagramme de Gantt

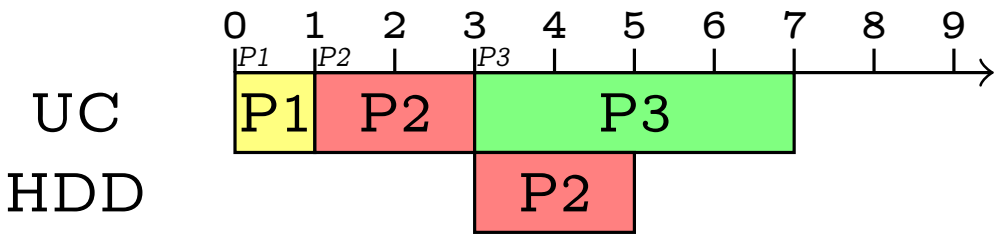
Représentation temporelle de l'allocation de ressources.



Proc.	date	Durée
P1	0	1
P2	1	2
P3	3	4

Diagramme de Gantt

Représentation temporelle de l'allocation de ressources.



Proc.	date	Durée
P1	0	1
P2	1	2
P3	3	4

Application

- Ordonnancement
- Génie logiciel
- Gestion d'équipe

Taux d'utilisation

Minimiser le temps où une ressource n'est pas utilisée.

Évaluation

Taux d'utilisation

Minimiser le temps où une ressource n'est pas utilisée.

Temps d'attente

Temps moyen ou maximum dans la file d'attente.

Évaluation

Taux d'utilisation

Minimiser le temps où une ressource n'est pas utilisée.

Temps d'attente

Temps moyen ou maximum dans la file d'attente.

Rotation

Durée moyenne totale des processus.

Évaluation

Taux d'utilisation

Minimiser le temps où une ressource n'est pas utilisée.

Temps d'attente

Temps moyen ou maximum dans la file d'attente.

Rotation

Durée moyenne totale des processus.

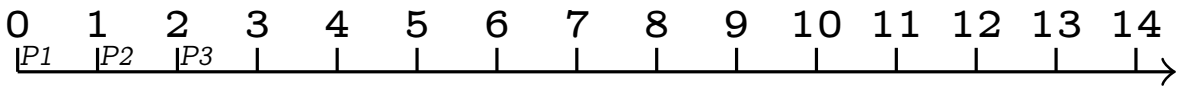
Débit

Nombre de processus fini par unité de temps.

First in, First out (FIFO)

Principe

File d'attente simple : premier arrivé, premier servi.

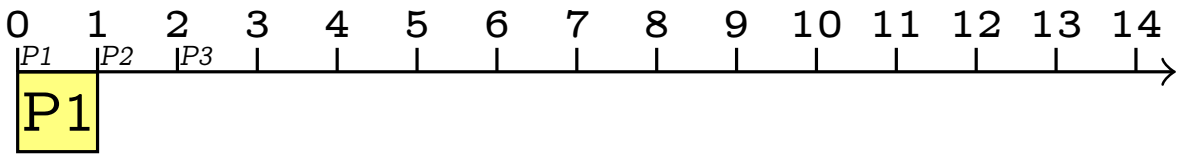


Proc.	date	Durée
P1	0	10
P2	1	2
P3	2	2

First in, First out (FIFO)

Principe

File d'attente simple : premier arrivé, premier servi.

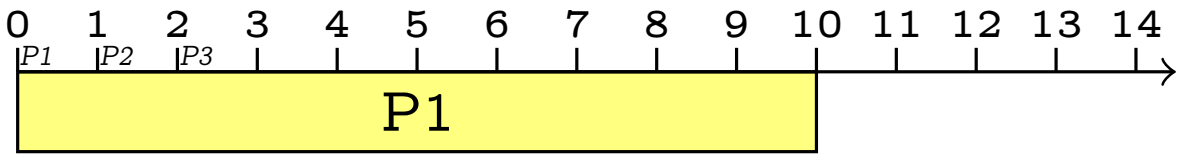


Proc.	date	Durée
P1	0	10
P2	1	2
P3	2	2

First in, First out (FIFO)

Principe

File d'attente simple : premier arrivé, premier servi.

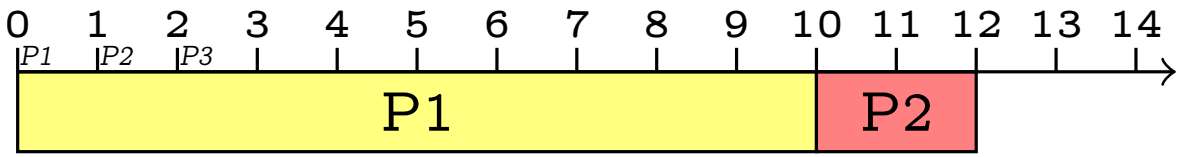


Proc.	date	Durée
P1	0	10
P2	1	2
P3	2	2

First in, First out (FIFO)

Principe

File d'attente simple : premier arrivé, premier servi.

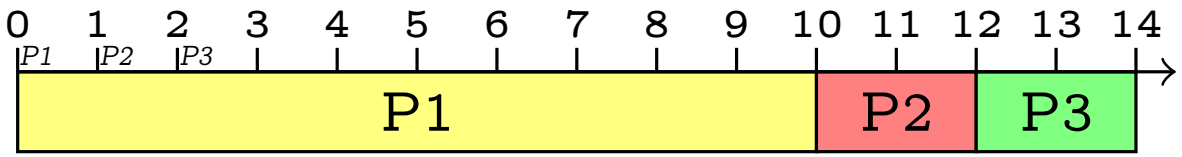


Proc.	date	Durée
P1	0	10
P2	1	2
P3	2	2

First in, First out (FIFO)

Principe

File d'attente simple : premier arrivé, premier servi.

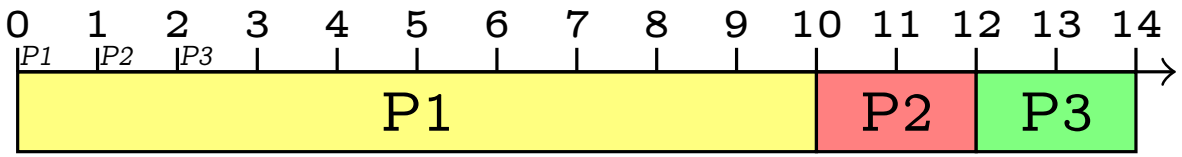


Proc.	date	Durée
P1	0	10
P2	1	2
P3	2	2

First in, First out (FIFO)

Principe

File d'attente simple : premier arrivé, premier servi.



Proc.	date	Durée
P1	0	10
P2	1	2
P3	2	2

Temps d'attente

- Moyen $\frac{0+9+10}{3} \approx 6.33$
- Maximum 10

First in, First out (FIFO)

Avantages

✓ Très simple à implémenter.

First in, First out (FIFO)

Avantages

✓ Très simple à implémenter.

Inconvénients

X Peu efficace : les processus de calcul bloquent le processeur.

X Peu réactif : les petits processus attendent longtemps.

First in, First out (FIFO)

Avantages

✓ Très simple à implémenter.

Inconvénients

X Peu efficace : les processus de calcul bloquent le processeur.

X Peu réactif : les petits processus attendent longtemps.

En pratique

Sert de brique de base à d'autres algorithmes.

Principe

- Associer à chaque processus une valeur de priorité
- Une file d'attente par priorité

Principe

- Associer à chaque processus une valeur de priorité
- Une file d'attente par priorité

Calcul de la priorité

- Type du processus
- Durée estimée, ressources utilisées...
- Ajustement par l'utilisateur
- Variable dans le temps
- ...

Réactivité

Attention

On ne s'intéresse pas à la durée totale du processus mais au temps pendant lequel il va garder le processeur.

Durée d'activité

Attention

On ne s'intéresse pas à la durée totale du processus mais au temps pendant lequel il va garder le processeur.

Temps avant :

- qu'il se termine
- qu'il fasse une E/S

Attention

On ne s'intéresse pas à la durée totale du processus mais au temps pendant lequel il va garder le processeur.

Temps avant :

- qu'il se termine
- qu'il fasse une E/S

C'est-à-dire, le temps avant qu'il quitte les états *prêt* ou *en exécution*.

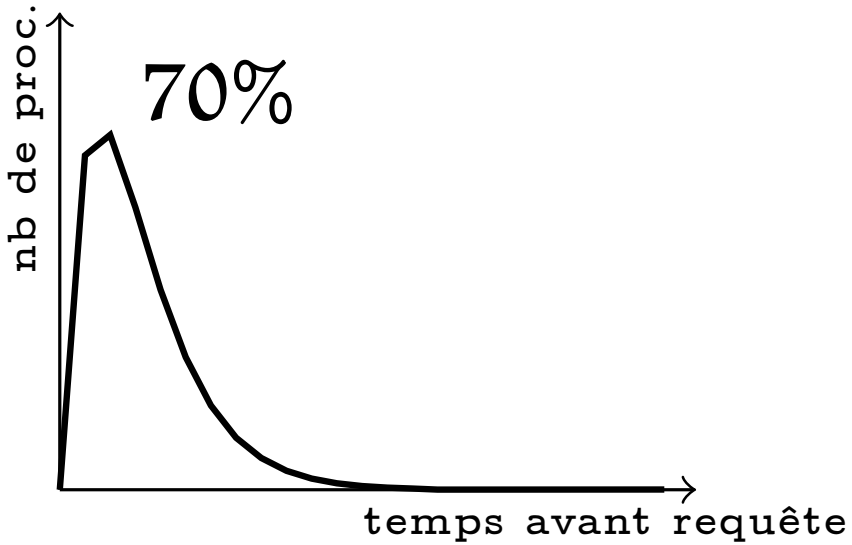
Difficultés I

Les E/S causent beaucoup d'interruptions

$\longrightarrow$ cycles processeurs très cours ($2\mu s$)

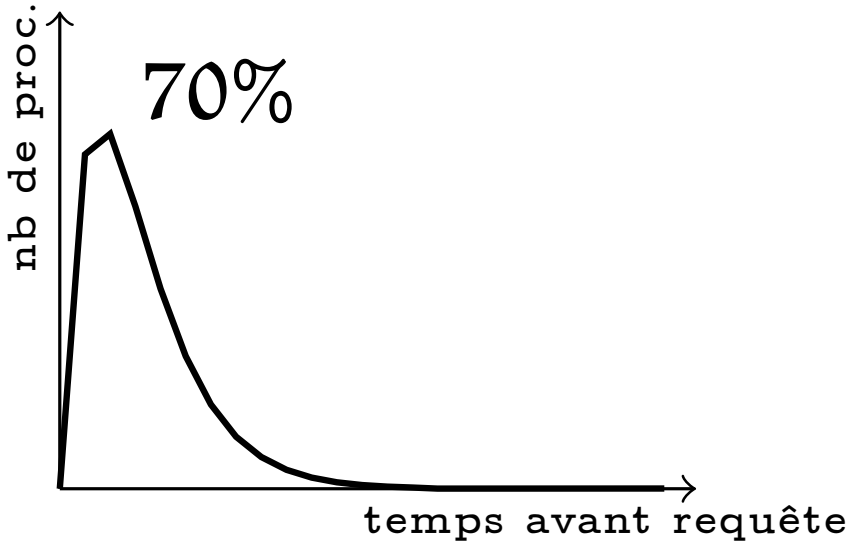
Les E/S causent beaucoup d'interruptions

$\longrightarrow$ cycles processeurs très cours ($2\mu s$)



Les E/S causent beaucoup d'interruptions

$\longrightarrow$ cycles processeurs très cours ($2\mu s$)



Réactivité

Permettre aux petits processus (E/S) d'accéder rapidement au processeur.

Commutation de contexte

Le changement de processus à un coût :

- Sauvegarde de l'ancien contexte
- Exécution de l'ordonnanceur
- Restauration du nouveau contexte

Commutation de contexte

Le changement de processus à un coût :

- Sauvegarde de l'ancien contexte
- Exécution de l'ordonnanceur
- Restauration du nouveau contexte

Il faut profiter des commutations naturelles

Commutation de contexte

Le changement de processus à un coût :

- Sauvegarde de l'ancien contexte
- Exécution de l'ordonnanceur
- Restauration du nouveau contexte

Il faut profiter des commutations naturelles

Ordonnancer lorsque

- Le processus se termine

Commutation de contexte

Le changement de processus à un coût :

- Sauvegarde de l'ancien contexte
- Exécution de l'ordonnanceur
- Restauration du nouveau contexte

Il faut profiter des commutations naturelles

Ordonnancer lorsque

- Le processus se termine
- Un processus passe dans l'état prêt

Algorithme préemtif

Comment rendre le système plus réactif ?

Algorithme préemtif

Comment rendre le système plus réactif ?

Préemption

Remplacement du processus actuellement en exécution par un processus arrivant dans la file des processus prêts.

Algorithme préemtif

Comment rendre le système plus réactif ?

Préemption

Remplacement du processus actuellement en exécution par un processus arrivant dans la file des processus prêts.

Ordonnancement lorsque :

Algorithme préemtif

Comment rendre le système plus réactif ?

Préemption

Remplacement du processus actuellement en exécution par un processus arrivant dans la file des processus prêts.

Ordonnancement lorsque :

- Le processus passe en attente ou se termine
  - Pas de problèmes de préemtion

Algorithme préemtif

Comment rendre le système plus réactif ?

Préemption

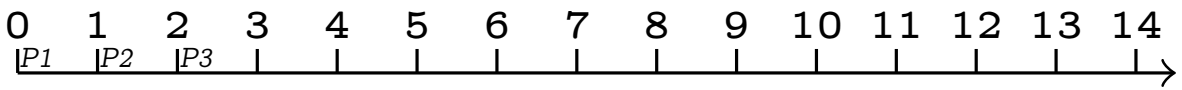
Remplacement du processus actuellement en exécution par un processus arrivant dans la file des processus prêts.

Ordonnancement lorsque :

- Le processus passe en attente ou se termine
  - Pas de problèmes de préemtion
- Un processus entre dans l'état prêt
  - Préempter ?

Principe

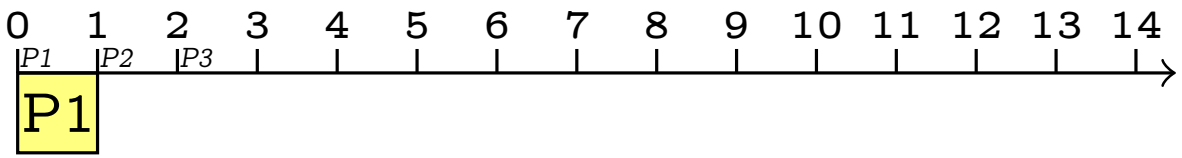
Priorité en fonction du temps restant. (estimé)



Proc.	date	Durée
P1	0	10
P2	1	2
P3	2	2

Principe

Priorité en fonction du temps restant. (estimé)



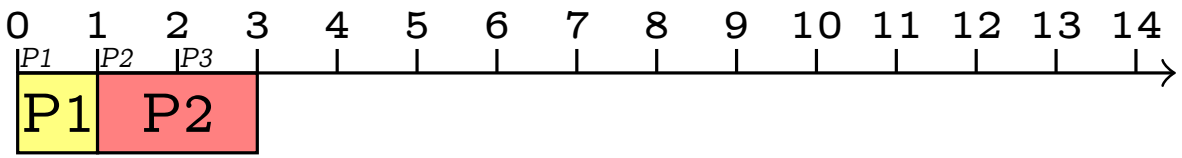
Proc.	date	Durée
P1	0	10
P2	1	2
P3	2	2

Plus court d'abord

Principe

Priorité en fonction du temps restant. (estimé)

Toujours préemptif.



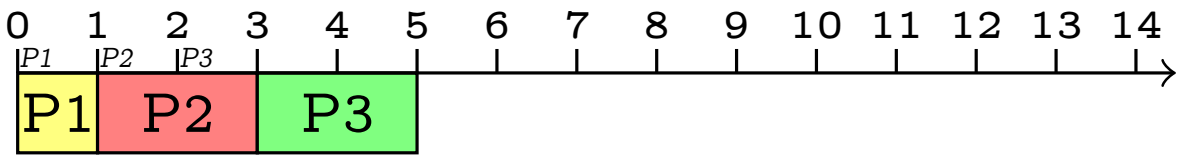
Proc.	date	Durée
P1	0	10
P2	1	2
P3	2	2

Plus court d'abord

Principe

Priorité en fonction du temps restant. (estimé)

Toujours préemptif.



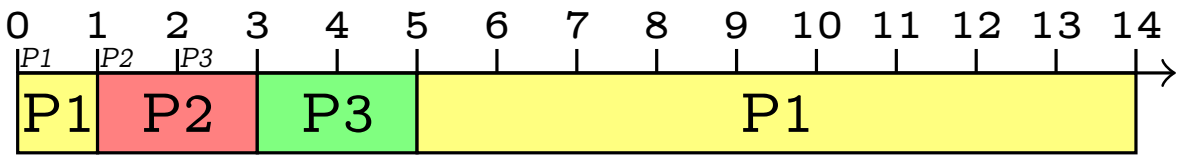
Proc.	date	Durée
P1	0	10
P2	1	2
P3	2	2

Plus court d'abord

Principe

Priorité en fonction du temps restant. (estimé)

Toujours préemptif.

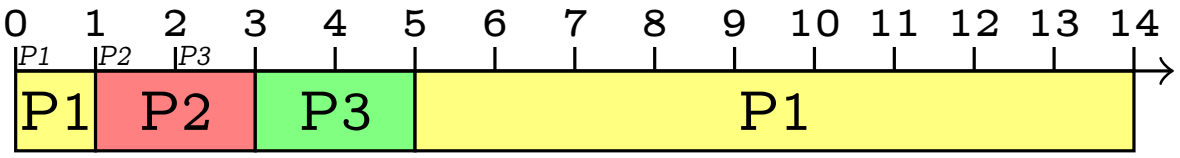


Proc.	date	Durée
P1	0	10
P2	1	2
P3	2	2

Principe

Priorité en fonction du temps restant. (estimé)

Toujours préemptif.

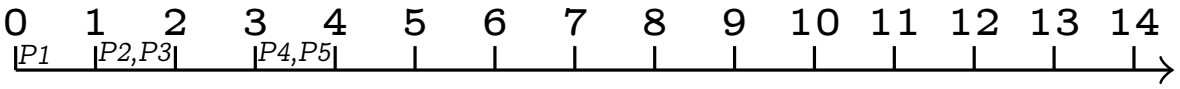


Proc.	date	Durée
P1	0	10
P2	1	2
P3	2	2

Temps d'attente

- Moyen $\frac{4+0+1}{3} \approx 1.7$
- Maximum 4

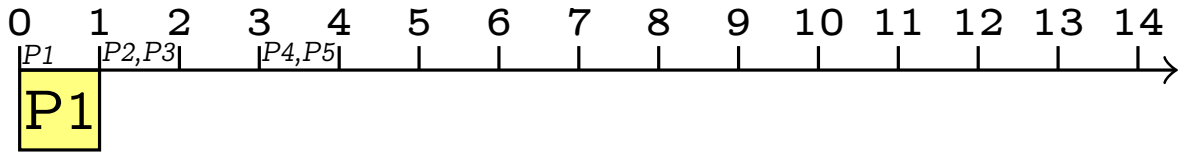
En cas d'égalitées ?

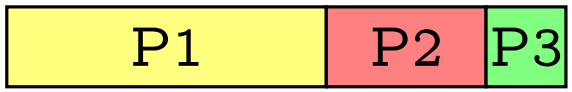


File : P1

Proc.	date	Durée
P1	0	5
P2	1	2
P3	1	1
P4	3	1
P5	3	1

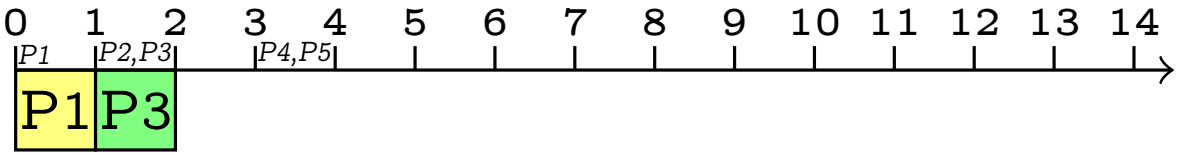
En cas d'égalitées ?



File : 

Proc.	date	Durée
P1	0	5
P2	1	2
P3	1	1
P4	3	1
P5	3	1

En cas d'égalitées ?

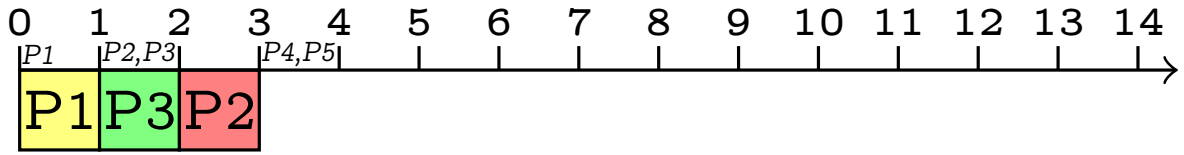


File : P1 P2

Proc.	date	Durée
P1	0	5
P2	1	2
P3	1	1
P4	3	1
P5	3	1

En cas d'égalitées ?

- Priorité au processus en cours d'exécution

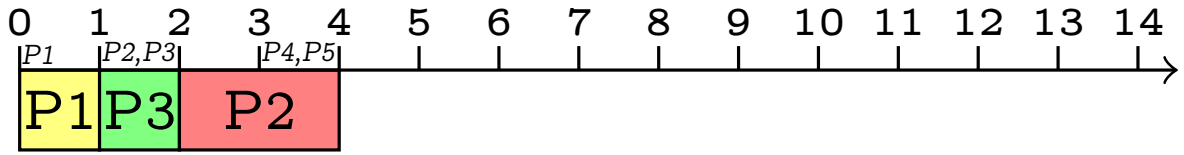


File : P1 P2 P4 P5

Proc.	date	Durée
P1	0	5
P2	1	2
P3	1	1
P4	3	1
P5	3	1

En cas d'égalitées ?

- Priorité au processus en cours d'exécution
- Puis en fonction de l'ordre d'arrivée

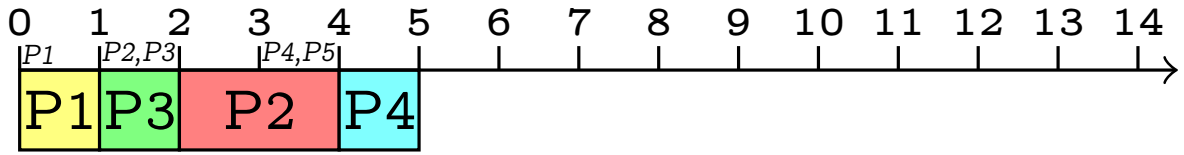


File : P1 P4 P5

Proc.	date	Durée
P1	0	5
P2	1	2
P3	1	1
P4	3	1
P5	3	1

En cas d'égalitées ?

- Priorité au processus en cours d'exécution
- Puis en fonction de l'ordre d'arrivée

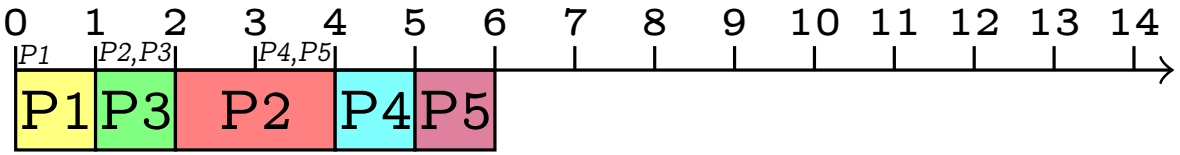


File : P1 P5

Proc.	date	Durée
P1	0	5
P2	1	2
P3	1	1
P4	3	1
P5	3	1

En cas d'égalitées ?

- Priorité au processus en cours d'exécution
- Puis en fonction de l'ordre d'arrivée

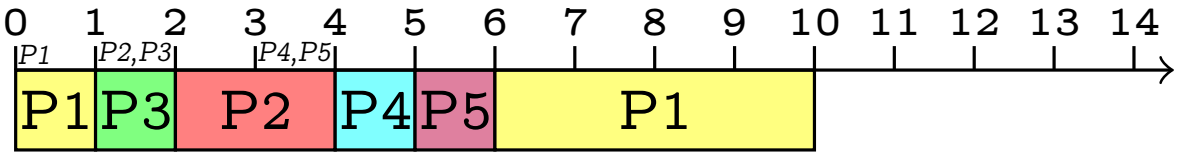


File : P1

Proc.	date	Durée
P1	0	5
P2	1	2
P3	1	1
P4	3	1
P5	3	1

En cas d'égalitées ?

- Priorité au processus en cours d'exécution
- Puis en fonction de l'ordre d'arrivée

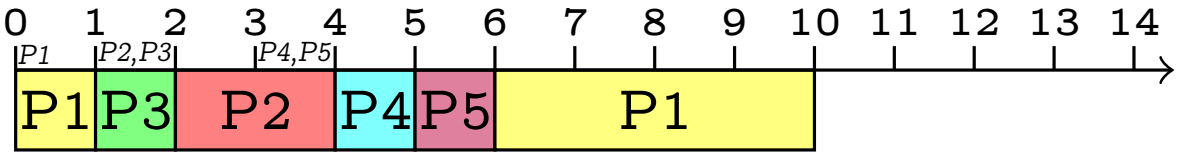


File :

Proc.	date	Durée
P1	0	5
P2	1	2
P3	1	1
P4	3	1
P5	3	1

En cas d'égalitées ?

- Priorité au processus en cours d'exécution
- Puis en fonction de l'ordre d'arrivée



File :

Proc.	date	Durée
P1	0	5
P2	1	2
P3	1	1
P4	3	1
P5	3	1

Temps d'attente

- Moyenne $\frac{5+1+0+1+2}{5} = 1.8$
- Maximum 5

Plus court d'abord

Problème

Comment connaître le temps restant ?

(avant requête ou interuption)

Problème

Comment connaître le temps restant ?

(avant requête ou interuption)

Estimation :

Moyenne exponentielles des cycles précédents :

Problème

Comment connaître le temps restant ?

(avant requête ou interuption)

Estimation :

Moyenne exponentielles des cycles précédents :

- P_t la durée estimée au temps t
- R_t la durée réelle au temps t

Problème

Comment connaître le temps restant ?

(avant requête ou interuption)

Estimation :

Moyenne exponentielles des cycles précédents :

- P_t la durée estimée au temps t
- R_t la durée réelle au temps t

$$P_{t+1} = \alpha P_t + (1 - \alpha) R_t$$

Généralement $\alpha = 0.5$

Plus court d'abord

Avantages

- ✓ Réactif : les petits processus passent en premier
- ✓ Optimal sur le temps d'attente moyen

Plus court d'abord

Avantages

- ✓ Réactif : les petits processus passent en premier
- ✓ Optimal sur le temps d'attente moyen

Inconvénients

- X Les processus de calcul on peu le processeur
- X Non équitable : famine pour les gros processus

Plus court d'abord

Avantages

- ✓ Réactif : les petits processus passent en premier
- ✓ Optimal sur le temps d'attente moyen

Inconvénients

- X Les processus de calcul on peu le processeur
- X Non équitable : famine pour les gros processus

Définition

On parle de famine lorsqu'un processus en attente n'a jamais accès à une ressource.

Équité

Ordonnancement équitable

FIFO

Un processus long peut monopoliser l'UC.

Ordonnancement équitable

FIFO

Un processus long peut monopoliser l'UC.

Plus court d'abord

Des processus courts peuvent monopoliser l'UC.

FIFO

Un processus long peut monopoliser l'UC.

Plus court d'abord

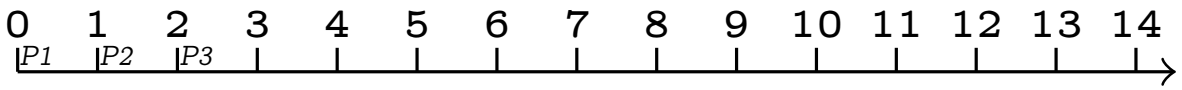
Des processus courts peuvent monopoliser l'UC.

Solution équitable

- Paramètres de l'algorithme non-manipulables
- Temps limite par processus avant préemption

Principe

FIFO avec quantum de temps.



Proc.	date	Durée
P1	0	10
P2	1	2
P3	2	2

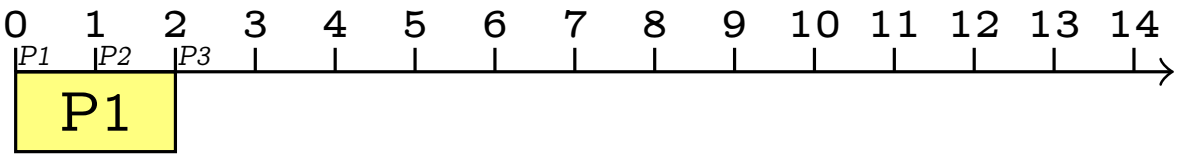
Quantum = 2

File : P1

Principe

FIFO avec quantum de temps.

Rarement préemptif.



Quantum = 2

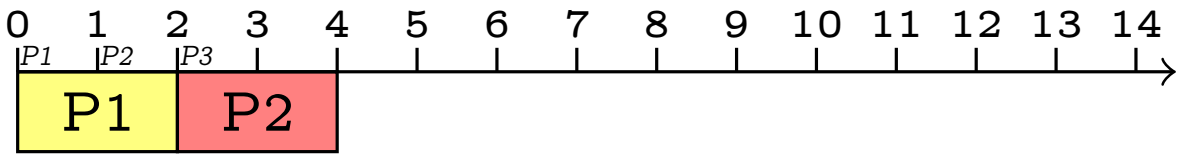
Proc.	date	Durée
P1	0	10
P2	1	2
P3	2	2

File : P2 | P1 | P3

Principe

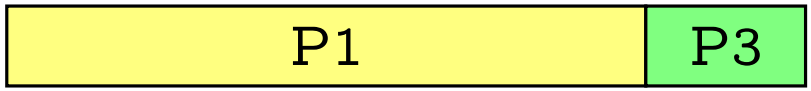
FIFO avec quantum de temps.

Rarement préemptif.



Quantum = 2

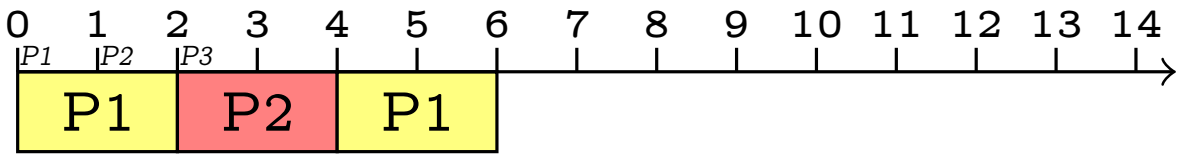
Proc.	date	Durée
P1	0	10
P2	1	2
P3	2	2

File : 

Principe

FIFO avec quantum de temps.

Rarement préemptif.



Quantum = 2

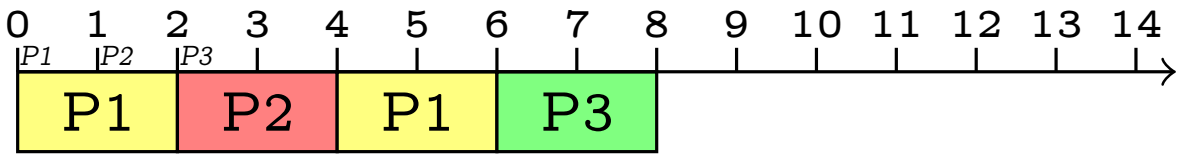
Proc.	date	Durée
P1	0	10
P2	1	2
P3	2	2

File : P3 | P1

Principe

FIFO avec quantum de temps.

Rarement préemptif.



Quantum = 2

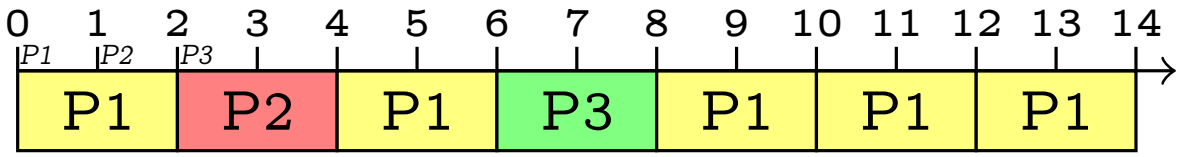
Proc.	date	Durée
P1	0	10
P2	1	2
P3	2	2

File : P1

Principe

FIFO avec quantum de temps.

Rarement préemptif.



Quantum $= 2$

Proc.	date	Durée
P1	0	10
P2	1	2
P3	2	2

Temps d'attente

- Moyenne $\frac{4+1+4}{3} = 3$
- Maximum 4

Round robin

Avantages

- ✓ Équitable : tous ont l'UC aussi rapidement
- ✓ Réactif : les petits processus attendent peu

Round robin

Avantages

- ✓ Équitable : tous ont l'UC aussi rapidement
- ✓ Réactif : les petits processus attendent peu

Inconvénients

- X Temps d'attente moyen plus élevé
- X Beaucoup de commutation (surcoût)

Round robin

Avantages

- ✓ Équitable : tous ont l'UC aussi rapidement
- ✓ Réactif : les petits processus attendent peu

Inconvénients

- X Temps d'attente moyen plus élevé
- X Beaucoup de commutation (surcoût)

En pratique

Efficace dès que la durée réelle de 80% des processus est inférieure au quantum.

Exemples concrets

Solaris (1993–2000)

Principe

- Priorités dynamique
- Quantum selon priorité

Principe

- Priorités dynamique
- Quantum selon priorité

P.	Q.	PN	PES
0	200	0	50
5	200	0	50
10	160	0	51
15	160	5	51
20	120	10	52
25	120	15	52
30	80	20	53
35	80	25	54
40	40	30	55
45	40	35	56
50	40	40	57
55	40	45	58
60	20	49	59

Principe

- Priorités dynamique
- Quantum selon priorité

P.	Q.	PN	PES
0	200	0	50
5	200	0	50
10	160	0	51
15	160	5	51
20	120	10	52
25	120	15	52
30	80	20	53
35	80	25	54
40	40	30	55
45	40	35	56
50	40	40	57
55	40	45	58
60	20	49	59

- Prioritaire = petit quantum
- Calcul = moins prioritaire
- E/S = plus prioritaire

Windows (depuis XP)

Multi-level Feedback Queue

32 niveaux de priorités + Round-Robin

Multi-level Feedback Queue

32 niveaux de priorités + Round-Robin

Priorités dynamiques

- Quantum consommé entièrement :
  $\rightarrow$ Priorité diminuée
- Retour d'entrée/sortie
  $\rightarrow$ Priorité augmenté

Windows (depuis XP)

Multi-level Feedback Queue

32 niveaux de priorités + Round-Robin

Priorités dynamiques

- Quantum consommé entièrement :
  $\rightarrow$ Priorité diminuée
- Retour d'entrée/sortie
  $\rightarrow$ Priorité augmenté

Favorise l'interface graphique

Système de crédit

- Chaque processus à des crédits (*priorité*)
- Le plus riche gagne (avec préemption)
- Perte d'1 crédit à la fin du quantum

Système de crédit

- Chaque processus à des crédits (*priorité*)
- Le plus riche gagne (avec préemption)
- Perte d'1 crédit à la fin du quantum

Tous les processus prêts à 0

Tous les processus sont recrédités

$$\text{cred} = \frac{\text{cred}}{2} + \text{prio}$$

Système de crédit

- Chaque processus à des crédits (*priorité*)
- Le plus riche gagne (avec préemption)
- Perte d'1 crédit à la fin du quantum

Tous les processus prêts à 0

Tous les processus sont recrédités

$$\text{cred} = \frac{\text{cred}}{2} + \text{prio}$$

Crédits accumulés pendant les E/S

Système de crédit

- Chaque processus à des crédits (*priorité*)
- Le plus riche gagne (avec préemption)
- Perte d'1 crédit à la fin du quantum

Tous les processus prêts à 0

Tous les processus sont recrédités

$$\text{cred} = \frac{\text{cred}}{2} + \text{prio}$$

Crédits accumulés pendant les E/S

Favorise l'interface graphique