Ordonnancement

Exercice 1. Questions de cours Les questions de cours sont à destinées à vous permettre de vérifier votre compréhension du cours. Elles sont à travailler à l'avance et ne seront pas traitées en TD ou TP.

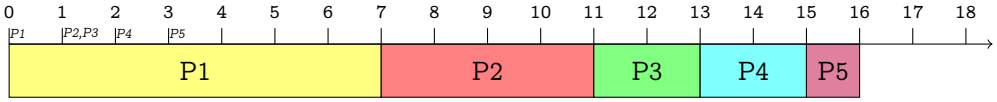
1. Quels sont les critères d'évaluation d'un ordonnanceur ? Indiquez pour chaque critère comment il doit être utilisé.
2. Citez deux inconvénients de l'algorithme d'ordonnancement FIFO.
3. Comment estimer le temps restant d'un processus pour l'algorithme plus court d'abord ?
4. Pourquoi cherche-t-on à réduire le nombre de commutation de processus ?
5. Dans quel cas l'algorithme Round-Robin devient-il particulièrement efficace ?

Exercice 2. Problème de base On considère les cinq processus suivants pour lesquels nous indiquons la date d'arrivée ainsi que leur durée d'exécution estimée.

Processus	Arrivée	Durée
P1	0	7
P2	1	4
P3	1	2
P4	2	2
P5	3	1

1. Dessinez le diagramme de Gantt du résultat d'un ordonnancement FIFO non préemptif et indiquez le temps d'attente moyen.

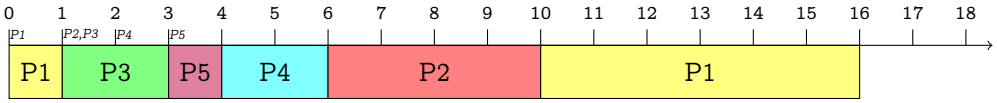
   Correction: *On prend juste les processus dans leur ordre d'arrivé, rien de spécial à noter ici.*



   Temps d'attente moyen : $(0+6+10+11+12)/5 = 39/5 = 7,8$

2. Dessinez le diagramme de Gantt du résultat d'un ordonnancement *plus court d'abord* préemptif et indiquez le temps d'attente moyen.

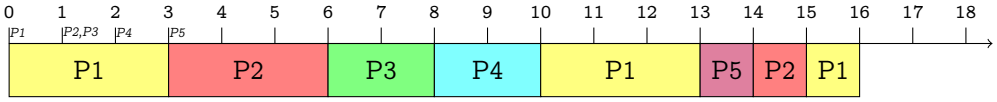
   Correction: *Au temps* t_1, *P3 prend la main sur P1 car sa durée restante estimée est plus courte. Au temps* t_2 *par contre, P4 ne prend pas la main car sa durée estimée n'est pas plus courte.*



   Temps d'attente moyen : $(9+5+0+2+0)/5 = 16/5 = 3,2$

3. Dessinez le diagramme de Gantt du résultat d'un ordonnancement *round-robin* non-préemptif avec un quantum de 3 et indiquez le temps d'attente moyen.

   Correction:



   Temps d'attente moyen : $(9+10+5+6+10)/5 = 40/5 = 8$

4. Quel est le meilleur algorithme suivant le critère du temps d'attente moyen ? Du temps d'attente min-max ?

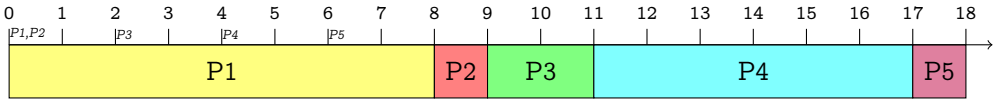
   Correction: *Pour le temps d'attente moyen, le meilleur est le plus court d'abord ; pour le temps d'attente min-max, on cherche l'algorithme qui minimise le temps d'attente maximum, c'est aussi le plus court d'abord.*

Exercice 3. Problème de base On considère les processus suivants, définis par leur durée d'exécution estimée, leur date d'arrivée et leur priorité initiale. Les plus grandes valeurs de priorité indiquent les processus les plus prioritaires.

Processus	Arrivée	Durée	Prioritée
P1	0	8	2
P2	0	1	1
P3	2	2	1
P4	4	6	4
P5	6	1	3

1. Dessinez le diagramme de Gantt correspondant au résultat d'un ordonnancement FIFO non préemptif et indiquez le temps d'attente moyen.

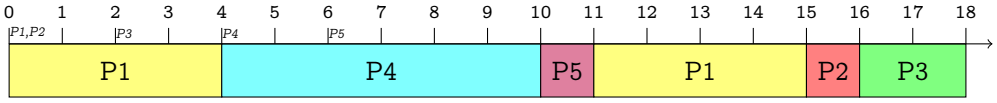
Correction: *Rien de particulier ici, on prend les processus dans l'ordre où ils arrivent en ignorant les priorités.*



Temps d'attente moyen : $(0+8+7+7+11)/5 = 33/5 = 6.6$

2. Dessinez le diagramme de Gantt correspondant au résultat d'un ordonnancement par priorité préemptif et indiquez le temps d'attente moyen.

Correction: *Ici, on choisi à chaque fois le processus ayant la plus grande priorité, il faut juste faire attention au fait qu'un processus peut en interrompre un autre s'il a une plus grande priorité ce qui se produit au temps 4.*

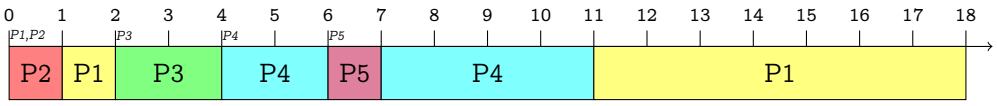


Temps d'attente moyen : $(7+15+14+0+4)/5 = 40/5 = 8$

On peut remarquer ici que si le temps d'attente moyen est plus important, les temps d'attente individuels des processus sont très fortement correlés aux prioritées des processus, ce qui est bien entendu une propriétés recherchée.

3. Dessinez le diagramme de Gantt correspondant au résultat d'un ordonnancement plus court d'abord préemptif et indiquez le temps d'attente moyen.

Correction: *Cette fois-ci, on choisit à chaque fois le processus ayant le temps restant le plus court. Là aussi il ne faut pas oublier qu'un processus court peut en interrompre un plus long au moment où il arrive.*

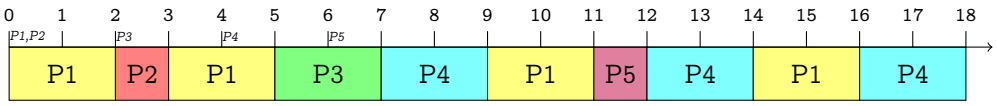


Temps d'attente moyen : $(10+0+0+1+0)/5 = 11/5 = 2.2$

Le temps d'attente moyen est ici exceptionellement bon ce qui est normal puisque cet algorithme garanti d'obtenir le meilleur temps d'attente moyen possible. Mais il faut relativiser ces performances qui sont ici illustrées sur un exemple trivial, dans la réalité l'écart avec les autres algorithme est beaucoup plus faible et ce fais aux prix d'un fort temps d'attente pour les processus les plus long.

4. Dessinez le diagramme de Gantt correspondant au résultat d'un ordonnancement round-robin non-préemptif, sans priorité, avec un quantum de temps fixé à 2 et indiquez le temps d'attente moyen.

Correction: *On reviens maintenant à une FIFO comme à la première question mais après deux unitées de temps les processus sont arrêtés et remis à la fin de la file.*



Temps d'attente moyen : $(8+2+3+8+5)/5 = 26/5 = 5.2$
Le temps d'attente moyen est ici bon mais le plus intéressant ici est de remarquer que tous les processus on régulièrement le processeur, aucun processus ne reste longtemps en attente. C'est ce qui garanti une bonne réactivité, une propriété essentielle des système modernes.

5. Quel est le meilleur algorithme suivant le critère du temps d'attente moyen ? Du temps d'attente min-max ?

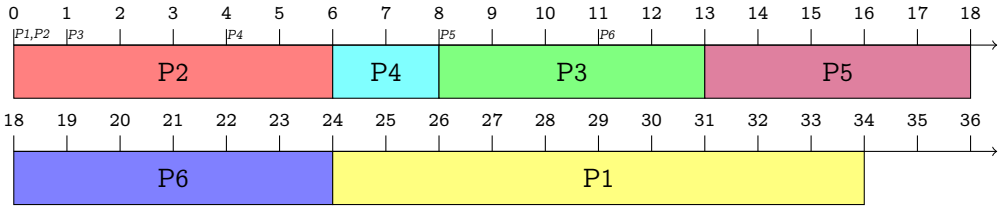
Correction: *Pour le temps d'attente moyen, le meilleur est le plus court d'abord ; pour le temps d'attente min-max, on cherche l'algorithme qui minimise le temps d'attente maximum, c'est donc le round-robin.*

Exercice 4. Priorités dynamiques et E/S On considère les processus suivants pour lesquels on indique la date d'arrivée, la durée estimée, la priorité et les temps où ils vont réaliser une demande d'E/S :

Processus	Arrivée	Durée	Priorité	E/S
P1	0	10	4	3, 5, 9
P2	0	6	2	
P3	1	5	3	
P4	4	2	4	1
P5	8	5	2	
P6	11	6	3	

1. Dessinez le diagramme de Gantt représentant l'ordonnancement de ces processus dans le cas de l'algorithme plus court-d'abord préemptif en ignorant les priorités et les demandes d'E/S. Vous indiquerez aussi le temps d'attente moyen.

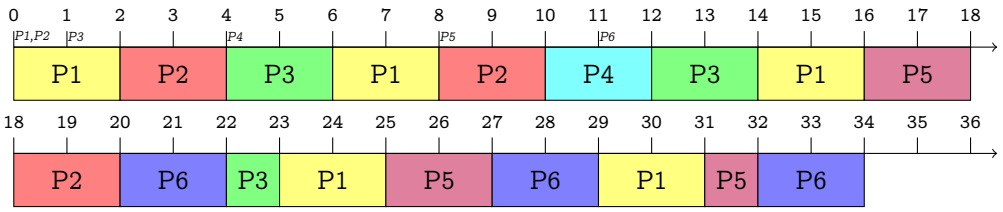
Correction: *L'algorithme est simple à appliquer ici. Il faut juste faire attention au fait qu'il peut y avoir préemption lorsqu'un nouveau processus arrive. Toutefois, le nouveau processus ne peut préempter que si son temps restant est strictement inférieur au temps restant du processus en cours. Aux temps 1 et 4 il n'y a donc pas de préemption.*



Attention, à T=8, P3 et P5 sont à égalité mais P3 est arrivé en premier et donc à la priorité.
Temps d'attente moyen : $(24+0+7+2+5+7)/6 = 45/6 = 7.5$

2. Toujours en ignorant les priorités et demandes d'E/S, dessinez le diagramme de Gantt dans le cas de l'algorithme *Round-Robin* avec un quantum de temps de 2. Vous indiquerez aussi le temps d'attente moyen.

Correction: *L'algorithme Round-Robin n'est pas très complexe mais demande de la rigueur. Il faut être très attentif à mettre correctement à jour la file et les temps restants des différents processus. La moindre erreur perturbe toute la suite de l'ordonnancement.*



Attention, au temps 4, les processus 2 et 4 arrivent en même temps dans la file, par convention, ils sont dans ce cas ajouter dans l'ordre des numéros de processus, c'est-à-dire P2 et premier puis P4. Dans la réalité, il est impossible que deux processus arrivent en même temps et le problème ne se pose pas, en cours, on choisi cette convention afin de rendre déterministe le résultat des problèmes.

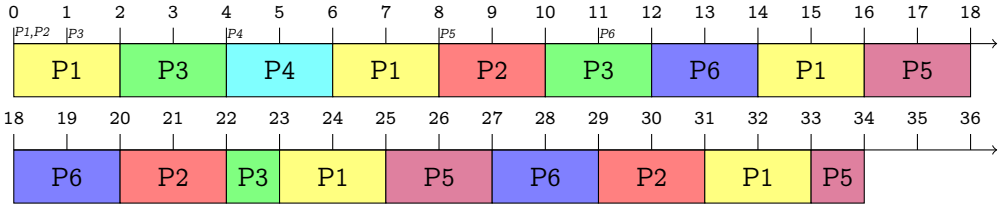
Temps d'attente moyen : $(21 + 14 + 17 + 6 + 19 + 17)/6 = 94/6 = 15.6$

On considère maintenant un système utilisant un ordonnancement par priorités dynamiques allant de 0 (la plus basse) à 4 (la plus haute). À chaque fois qu'un processus se voit attribuer le processeur, sa priorité baisse de 1 sans jamais descendre en dessous de 0.

L'algorithme d'ordonnancement choisit le processus dont la priorité est la plus élevée et pour un niveau de priorité il utilise l'algorithme *Round-Robin* avec un quantum de 2.

3. Dessinez le diagramme de Gantt pour ce nouveau système, toujours en ignorant les demandes d'E/S. Vous indiquerez aussi le temps d'attente moyen.

Correction: *L'algorithme se déroule de manière similaire à la question précédente mais avec la complexitée aditionelle de devoir gérer une FIFO par niveau de priorité.*

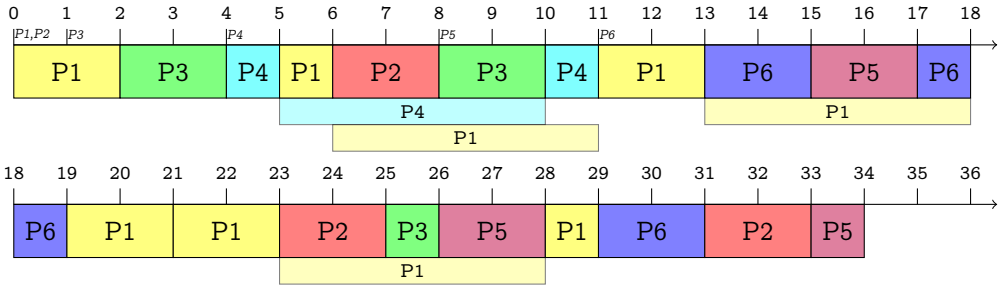


Il n'y a rien de particulier ici, on remarque qu'au temps 4 il n'y a plus de soucis, les processus 2 et 4 arrivent toujours en même temps dans l'état prêt mais avec des priorités différentes.
Temps d'attente moyen : $(24 + 16 + 21 + 0 + 19 + 14)/6 = 94/6 = 15.6$

Les processus peuvent faire des demandes d'E/S qui suspendent le processus pendant 5 unités de temps. Lorsqu'un processus repasse dans la file "Prêt" après une demande d'E/S, sa priorité est réinitialisée à sa valeur initiale mais il ne peut pas préempter le processus actuellement en cours d'exécution.

4. Dessinez le diagramme de Gantt représentant l'ordonnancement des processus dans le cas de ce système avec priorité et demandes d'E/S. Vous indiquerez aussi le temps d'attente moyen.

Correction: *Le principe reste très similaire à la question suivante mais la gestion de 5 files de priorités au lieu d'une et la prise en compte des E/S demande encore plus d'attention. N'essayer surtout pas d'aller trop vite ici.*



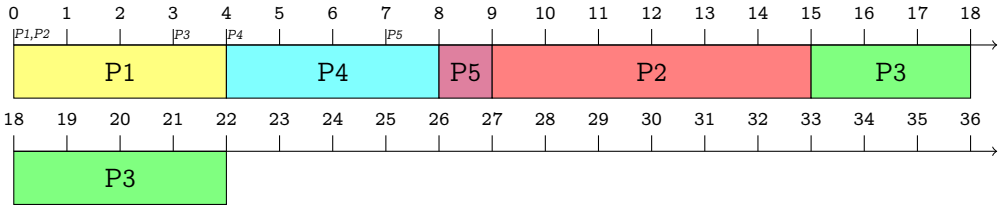
Temps d'attente moyen : $(4 + 27 + 20 + 0 + 21 + 14)/6 = 86/6 = 14.3$

Exercice 5. Priorités dynamiques et E/S On considère les processus suivants définis par leur date d'arrivée, leur durée estimée, leur priorité initiale ainsi que les entrées/sorties qu'ils vont effectuer. Les priorités sont comprises entre 0 et 3, les plus grandes valeurs indiquant les processus les plus prioritaires. Pour chaque E/S on indique après combien de temps d'exécution le processus réalise la demande et, entre parenthèses, combien de temps celle-ci va durer.

Processus	Arrivée	Durée	Priorité	E/S
P1	0	4	2	3(3)
P2	0	6	1	
P3	3	7	3	
P4	4	4	2	1(3), 2(4)
P5	7	1	2	

1. On ignore pour l'instant les priorités et les demandes d'E/S. Dessinez le diagramme de Gantt représentant l'ordonnancement des processus pour l'algorithme *plus court d'abord* préemptif. Vous indiquerez les temps d'attente moyen.

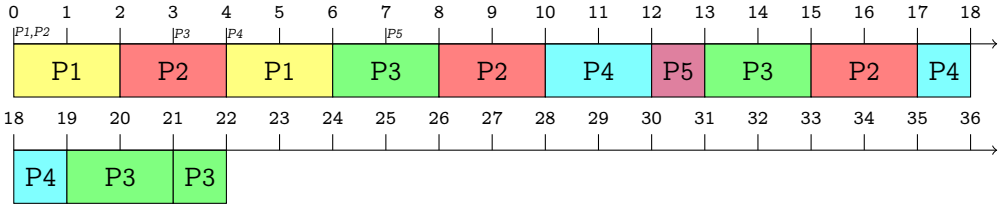
Correction: *Simple, juste ne pas oublier que P5 ne préempte pas P4 lorsqu'il arrive au temps* $\mathbf{t_7}$*, le deux processus on le même temps restant mais P4 est déjà en cours d'exécution, il garde donc le processeur.*



Temps d'attente moyen : $(0+9+12+0+1)/5 = 22/5 = 4,4$

2. Toujours en ignorant les priorités et les demandes d'E/S, dessinez le diagramme de Gantt représentant l'ordonnancement des processus par l'algorithme *round-robin* non-préemptif avec un quantum de 2. Vous indiquerez aussi le temps d'attente moyen.

Correction: *Simple aussi, juste faire attention au temps* $\mathbf{t_4}$ *car les processus P2 et P4 arrivent en même temps dans la file.*

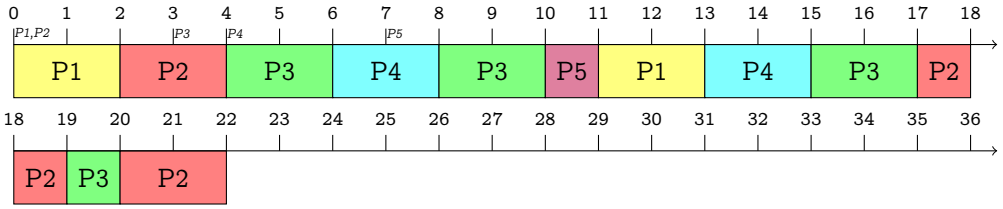


Temps d'attente moyen : $(2+11+12+11+5)/5 = 41/5 = 8,2$

On introduit maintennt les priorités dans le système. L'ordonnancement est de type *round-robin* non-préemptif par niveau de priorité avec un quantum de 2. Lorsqu'un processus utilise l'intégralité de son quantum, sa priorité est réduite de 1. Si au moment d'ordonnancer tous les processus ont une priorité de 0, ils sont tous remontés à une priorité de 2.

3. Dessinez le diagramme de Gantt représentat l'ordonnancement selon cet algorithme et indiquez le temps d'attente moyen.

Correction:

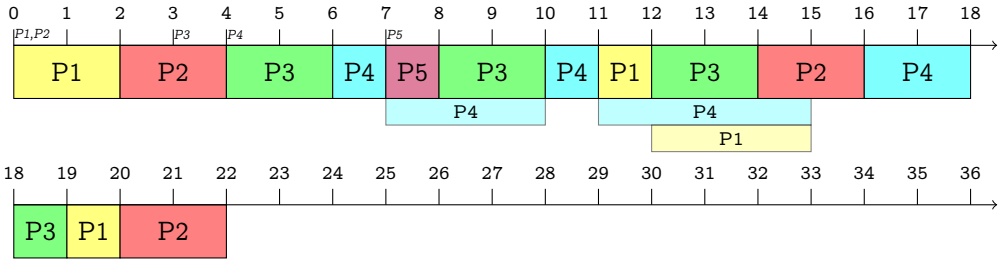


Temps d'attente moyen : $(9+16+10+7+3)/5 = 45/5 = 9$

Enfin, nous introduisons la prise en compte des E/S qui suspendent le processus pour une certaine durée le temps que la demande soit satisfaite. Lorsque l'E/S se termine, le processus reviens dans l'état près avec une priorité augmentée de 1 mais sans possibilité de préempter.

4. Dessinez le diagramme de Gantt représentant l'ordonnancement selon cet algorithme et indiquez le temps d'attente moyen et maximal.

Correction:



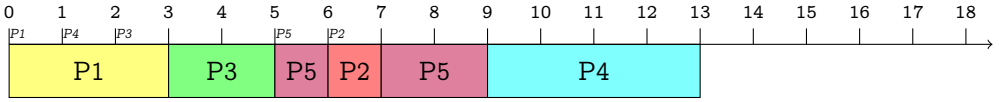
Temps d'attente moyen : $(13+16+9+3+0)/5 = 42/5 = 8.2$

Exercice 6. Système à deux processeurs On considère les processus suivants, définis par leur date d'arrivée et leur durée estimée :

Processus	Arrivée	Durée
P1	0	3
P2	6	1
P3	2	2
P4	1	4
P5	5	3

1. Dessinez le diagramme de Gantt representant l'ordonnancement selon l'algorithme *plus court d'abord* préemptif et indiquez le temps d'attente moyen.

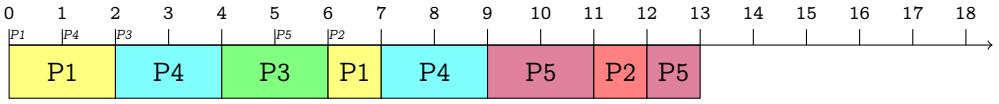
Correction:



Temps d'attente moyen : $(0+0+1+8+1)/5 = 10/5 = 2$

2. Dessinez le diagramme de Gantt representant l'ordonnancement selon l'algorithme *round-robin* avec un quantum de 2 et indiquez le temps d'attente moyen.

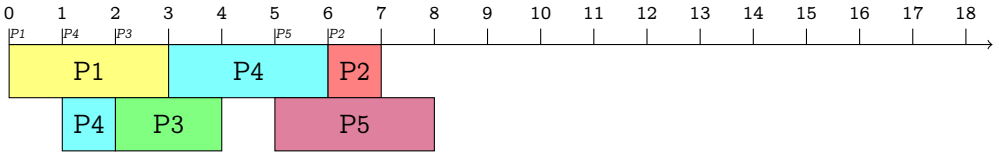
Correction:



Temps d'attente moyen : $(4+5+2+4+5)/5 = 20/5 = 4$

3. Dessinez le diagramme de Gantt reprenant la question 1 sur un système disposant de deux processeurs.

Correction: *Tout se passe comme avant à l'exception que l'on placer en exécution deux processus au lieu d'un seul. Lorsqu'il y a préemption, on compare le temps restant du processus arrivant à celui des deux processus actuellement en exécution et on remplace si possible. Si les deux processus en exécution on des temps plus long que celui qui arrive, on va préempter celui à qui il reste le plus de temps restant.*

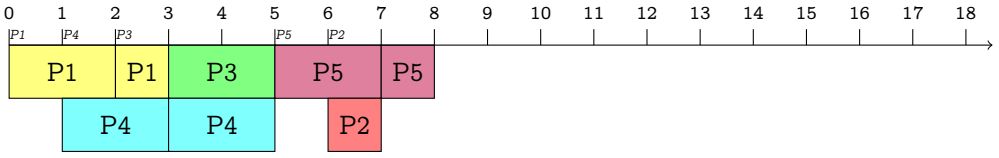


Temps d'attente moyen : $(0+0+0+1+0)/5 = 1/5 = 0.2$

Le deuxième processeur apporte ici beaucoup de flexibilité et permet de quasiment éliminer le temps d'attente des processus.

4. Dessinez le diagramme de Gantt reprenant la question 2 sur un système disposant de deux processeurs.

Correction:



Au temps t_2*, les processus P1 et P3 arrivent en même temps dans la file d'attente, p1 est choisi car on utilise le critère de l'ordre des numéros de processus. Au temps* t_3*, les deux processeurs sont disponibles pour les processus P3 et P4, le mieux est de placer P4 sur le deuxième processeur car il est déjà en place, la commutation de contexte sera donc moins coûteuse.*

Temps d'attente moyen : $(0+0+1+0+0)/5 = 1/5 = 0.2$

Exercice 7. Cas d'étude On considère maintenant les processus suivants. Les valeurs de priorités les plus grandes indiquent les processus les plus prioritaires, pour les E/S sont indiqué le temps (dans l'exécution du processus) où se produit l'E/S puis sa durée entre parenthèses.

Processus	Arrivée	Durée	Priorité	E/S
P1	0	10	3	4(2), 9(2)
P2	0	3	2	
P3	2	1	3	
P4	6	8	2	2(8)
P5	8	2	4	1(5)
P6	8	6	1	

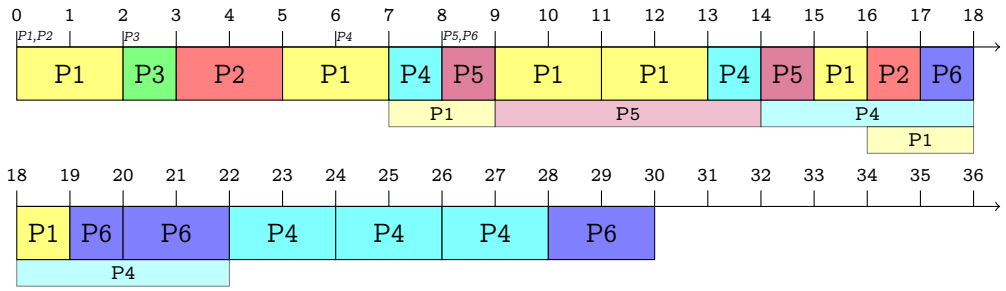
Première approche : On considère le principe d'ordonnancement suivant, inspiré de l'algorithme utilisé sous Windows dans lequel chaque processus est muni d'une valeur de priorité dynamique :

- lorsqu'un processus a consommé son quantum de temps, il est interompu et voit sa priorité diminuer de 1 (une priorité ne peut jamais être inferieure à zéro) ;
- lorsqu'un processus est interompu avant la fin de son quantum, sa priorité ne change pas et il terminera son quantum la prochaines fois qu'il sera sélectionné par l'ordonnanceur ;
- lorsqu'un processus retourne dans la file "prêt" après avoir réalisé une E/S, il voit sa priorité augementée de 2.

L'algorithme est préemptif : il choisit systématiquement le processus ayant la priorité la plus élevée. Au sein d'un même niveau de priorité, il utilise le principe du Round-Robin avec un quantum de 2.

1. Dessinez le diagramme de Gantt correspondant à l'exécution de ces 5 processus pour l'algorithme décris et indiquez le temps d'attente moyen.

Correction:



t_0 *Arrivée des processus P1 et P2. P1 a la priorité la plus élevée et donc est choisi.*

t_2 *Le quantum de P1 est épuisé, il est donc suspendu et sa priorité passe à 2, le processus P3 arrive avec une priorité de 3, c'est donc lui qui est choisi.*

t_3 *P3 se termine. P2 et P1 sont, dans cet ordre, dans la file de priorité 2. P2 est donc choisi.*

t_5 *Le quantum de P2 est épuisé, il est suspendu et sa priorité passe à 1. Le processus P1 est à nouveau choisi.*

t_6 *P4 arrive mais sa priorité est la même que celle de P1, il va donc dans la file de priorité 2.*

t_7 *P1 fait une E/S et est donc placé dans la file "en attente" pour les deux prochaines unité de temps. P4 est choisi car il a la plus grande priorité. On peut remarquer que P1 une E/S pile à la fin de son quantum, on suppose qu'il l'a fait en fait un epsilon de temps avant la fin.* **Attention :** *La priorité de P1 ne diminue donc pas puisqu'il n'a pas épuisé tout son quantum. Il reste à 2 et reviendra donc avec une priorité de 4.*

t_8 *P5 arrive, puisque sa priorité est supperieur à celle de P4, ce dernier va être suspendu pour lui laisser la place. P4 est replacé dans la file de priorité 2, sa priorité ne baisse pas car il n'a pas pus consomer ses deux unités de temps. P6 arrive aussi et se place dans la file de priorité 1 après P2.*

t_9 *P5 pars en E/S pour 5 unités de temps et P1 reviens dans la file de priorité 4. P1 est donc choisi car il est le plus prioritaire.*

t_{11} *P1 termine sont quantum sa priorité baisse donc à 3. À ce moment, nous avons donc P1 dans la file de priorité 3, P4 dans celle de priorité 2 et P2 suivi de P6 dans celle de priorité 1. P1 est donc à nouveau choisi.*

t_{13} *P1 termine sont quantum et sa priorité passe à 2. P4 est devant lui dans cette file et donc est choisi.*

t_{14} *P4 pars en E/S pour 8 unités de temps. P5 termine son E/S et reviens donc dans la file de priorité 6, il est donc choisi.*

t_{15} *P5 se termine. P1, avec une priorité de 2, est le processus le plus prioritaire et donc est choisi.*

t_{16} *P1 pars en E/S pour 2 unités de temps. P2 à enfin à nouveau choisi ! On peut bien sûr avoir plusieurs processus en E/S en même temps.*

t_{17} *P2 se termine. P6 est le seul processus près, P1 et P4 étant en attente, il est donc choisi pour la première fois !*

t_{18} *P1 termine son E/S et reviens avec une priorité de 4, P6 est donc interompu pour lui laisser la place.*

t_{19} *P1 se termine. P6 est choisi et il va pouvoir terminer son quantum qu'il n'avait pas pus finir.*

t_{20} *P6 termine son quantum et sa priorité tombe à 0. Il n'y a pas d'autres processus, il est donc à nouveau choisi.*

t_{22} *P4 reviens de son E/S avec une priorité de 4, il va donc pouvoir garder la main pendant 4 tours avant que P6 puisse être choisi à nouveau mais 3 tours sont suffisants pour qu'il termine son exécution.*

t_{28} *P4 se termine et P6 reprend la main.*

t_{30} *P6 se termine.*

Temps d'attente moyen : $(5 + 14 + 0 + 6 + 0 + 16)/6 = 41/6 = 6.83$

Deuxième approche : on utilise maintenant un autre algorithme d'ordonnancement, inspiré des algorithmes utilisées par les systèmes Linux et MacOS X, qui utilise aussi des priorités dynamiques pour les processus :

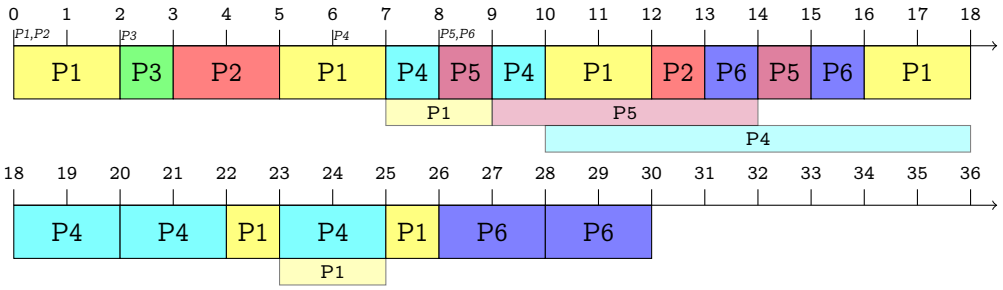
- lorsque un processus à consomé son quantum de temps, il est interrompu et voit sa priorité diminuer de 1 (une priorité ne peut jamais être inferieure à zéro) ;
- lorsque tous les processus *prêts* ont une priorité de 0, la priorité de chaque processus est modifiée :

$$\text{nouvelle priorité} = \frac{\text{priorité actuelle}}{2} + \text{priorité initiale}.$$

L'algorithme est préemptif : il choisit systématiquement le processus ayant priorité la plus élevée. Au sein d'un même niveau de priorité, il utilise le principe du Round-Robin avec un quantum de 2.

2. Dessiner le diagramme de Gantt correspondant à l'exécution de ces processus pour l'algorithme décris et indiquez le temps d'attente moyen.

Correction:



Tant que au moins un processus a une priorité > 0 et qu'aucun processus ne reviens d'une E/S, cette méthode est identique à la précédente. Le déroulement est donc le même jusqu'à $t = 9$ lorsque P1 reviens de sa première E/S. Les moments importants sont les suivants :

t_9 *P1 reviens avec une priorité de 2 et se place donc derrière P4 dans la file. P5 part en E/S. P2 et P6 sont dans la file de priorité 1, c'est donc P4 qui est choisi.*

t_{10} *P4 part en E/S, P1 est donc choisi.*

t_{12} *Le quantum de P1 se termine, il est donc suspendu et se place dans la file de priorité 1. P2 est choisi car il est en tête de cette file.*

t_{13} *P2 se termine et c'est maintenant au tour de P6 d'être choisi.*

t_{14} *P5 reviens avec une priorité de 4. P6 est donc interompu car sa priorité est inferieure. Attention : P6 à été interrompu avant d'avoir pus finir son quantum, il n'est donc pas replacé en fin de file mais conserve sa position et finira son quantum la prochaine fois qu'il sera choisi.*

t_{15} *P5 se termine. P6 est choisi pour terminer son quantum interrompu.*

t_{16} *P6 termine son quantum et a maintenant une priorité de 0. P1 est choisi.*

t_{18} *P1 termine son quantum et a maintenant lui aussi une priorité de 0. P4 reviens de son E/S avec une priorité de 2. Il est donc choisi et va garder la main pendant deux tours jusqu'à ce que sa priorité tombe à 0.*

t_{22} *Le quantum de P4 se termine et il est placé en fin de file 0 derrière P6 et P1. Tous les processus prêts ont une priorité de 0, il sont donc tous recrédités de leur priorité initiale : 3 pour P1, 2 pour P4 et 1 pour P6. C'est P1 qui est choisi.*

t_{23} *P1 fait une E/S. P4 est choisi.*

t_{25} *P4 se termine, P1 revient avec une priorité de 3 et prend la main.*

t_{26} *P1 se termine. P6 est maintenant tout seul.*

t_{30} *P6 se termine.*

Temps d'attente moyen : $(12 + 10 + 0 + 3 + 0 + 16)/6 = 41 = 6.83$