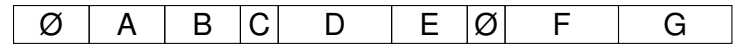
TD2 : Ordonnancement de processus

1 Ordonnancement de tâches

On s'intéresse dans cette partie à des tâches purement calculatoires (autrement dit, des CPU-bursts) A, B, C... dont les instants d'arrivée et les durées d'exécution sont indiquées par le tableau suivant :

tâche	A	B	C	D	E	F	G
arrivée	2	4	5	7	9	15	16
durée	3	2	1	4	2	6	8

Exemple Le chronogramme ci-dessous représente l'exécution de ces différentes tâches, ordonnancées selon la politique FCFS (First Come First Served). Le tableau indique les temps de séjour (TT pour *Turnaround Time*) et temps d'attente (WT pour *Waiting Time*) :



tâche	A	B	C	D	E	F	G
TT	3	3	3	5	5	6	13
WT	0	1	2	1	3	0	5

Ce scénario génère un temps d'attente total de 12 unités de temps.

Exercice Faites le même travail pour la politique d'ordonnancement SJF (Shortest Job First) : dessinez le chronogramme et calculez le temps d'attente total.

Exercice Même consigne pour SRTF (Shortest Remaining Time First) puis RR (Round-Robin) avec un quantum de 1 puis RR avec un quantum de 5.
Remarque : il faut lire les dates d'arrivée comme «juste avant» l'instant indiqué. Par exemple la tâche A arrive «juste avant 2» et donc à l'instant t=2 elle est déjà prête.

Exercice Quel(s) seraient le/les avantages de choisir une politique Round Robin avec un quantum encore plus long, par exemple 20 ? Respectivement, quels seraient les inconvénients ? Répondez à chaque fois par une phrase rédigée.

2 Ordonnancement de processus avec entrées-sorties

On s'intéresse dans cette partie à des processus faisant successivement des calculs et des entrées-sorties. Plus précisément, chaque processus fait une CPU-burst (de la durée indiquée ci-dessous) puis une IO-burst durant 10 unités de temps, puis une seconde CPU-burst identique à la première, puis il se termine.

Processus	1ière CPU-burst, puis	IO-burst, puis	2nde CPU-burst
A	2	10	2
B	3	10	3
C	7	10	7
D	18	10	18

À l'instant initial (t=0) tous les processus sont prêts à s'exécuter. On suppose que la machine ne dispose que d'un unique processeur, mais qu'elle peut traiter plusieurs requêtes d'entrées-sorties simultanément. [1]

Exercice Dessinez quatre chronogrammes représentant l'ordonnancement de ces processus selon les stratégies suivantes : SJF, SRTF, RR avec quantum=6, et RR avec quantum=20. Discutez avec votre voisin de table des avantages et inconvénients de chacune de ces stratégies sur ce scénario.

1. Si ça vous chagrine trop, imaginez plutôt que ce sont des `sleep(N)`, l'exercice reste le même.

3 Prédiction des durées de CPU-burst

Dans plusieurs politiques d'ordonnancement, le choix du processus à exécuter dépend des durées des prochaines *CPU-burst* des différents candidats. Dans le cas général, cette information n'est pas connue à l'avance. Mais pour chaque processus il est possible d'en faire une *estimation* en se basant sur les durées effectivement observées dans le passé.
Typiquement, la durée de la prochaine CPU-burst d'un processus P est prédite à l'aide d'une *moyenne mobile exponentielle* de ses précédentes durées. Notons $d(n)$ la durée de la n^e CPU-burst de P et notons $e(n+1)$ la durée estimée de sa prochaine CPU-burst. Pour un certain paramètre α fixé entre 0 et 1, l'estimation se fait par la formule suivante :

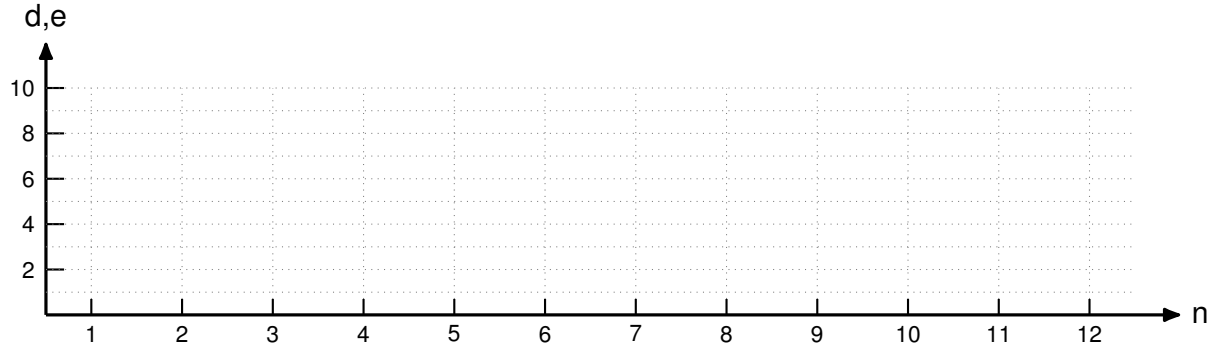
$$e(n+1) = \alpha \cdot d(n) + (1 - \alpha) \cdot e(n)$$

La mesure $d(n)$ est notre information la plus récente sur le comportement de P. La valeur $e(n)$ représente l'historique de nos prédictions. Les paramètres de notre estimateur sont α et $e(1)$. Le choix de α permet de pondérer la «faculté de mémoire» de l'ordonnanceur, et la valeur de $e(1)$ sert d'estimation par défaut pour les processus encore inconnus.

Exercice On s'intéresse à un ordonnanceur avec les paramètres suivants : $\alpha = 0.5$ et $e(1) = 5$.
Soit un processus P dont les durées successives de CPU-burst sont 10, 10, 10, 10, 10, 10, 1, 1, 1, 1, 1, 1 (on rappelle que l'ordonnanceur ne connait *pas* ces durées à l'avance.)
Dans le tableau ci-dessous, indiquez les prévisions successives de l'ordonnanceur (ne gardez pas plus d'un ou deux chiffres après la virgule) au fur et à mesure de l'exécution.

n	1	2	3	4	5	6	7	8	9	10	11	12
e(n)	5											
d(n)	10	10	10	10	10	10	1	1	1	1	1	1

Exercice Représentez toutes ces valeurs sur le graphe ci-dessous. Utilisez des couleurs distinctes pour $d(n)$ et $e(n)$.



Note : Remarquez au passage que ça ne voudrait rien dire de relier entre eux les points successifs sur ce graphe. Pourquoi ?

Exercice Supposons maintenant que l'ordonnanceur est paramétré avec une valeur de α plus petite, par exemple 0.25 voire 0.1 ? Quel impact cela aurait-il sur les prévisions ? Illustrez ce changement sur votre graphique (pas la peine de refaire les calculs). Mêmes questions avec une valeur de α plus proche de 1, par exemple 0.75 ou 0.9.

Exercice Proposez un scénario (sous forme d'une séquence de durées de CPU-burst) pour lequel une trop petite valeur de α conduirait l'ordonnanceur à faire de «mauvaises» prédictions. Justifiez votre réponse par une phrase rédigée et illustrez ce scénario sur un graphique.
Respectivement, proposez un scénario pour lequel une valeur de α trop proche de 1 conduirait l'ordonnanceur à faire de «mauvaises» prédictions. Justifiez votre réponse par une phrase rédigée et illustrez ce scénario sur un graphique.

4 Taux d'utilisation du processeur

Un des critères d'évaluation en ordonnancement est le taux d'utilisation du processeur (en VO *CPU utilization rate*) c'est à dire la fraction du temps effectivement passée à exécuter du code applicatif. Notons cette valeur U.
On s'intéresse à un système dans lequel les processus s'exécutent en moyenne pendant une durée T entre deux requêtes d'entrées-sorties. Lorsqu'un processus se bloque, l'ordonnanceur fait une commutation de contexte vers un processus prêt (on suppose qu'il y en a toujours). Notons S ce temps de *context-switch*, pendant lequel aucun processus ne fait de travail utile. L'ordonnanceur applique une politique round-robin avec un quantum de durée Q.

Exercice Donnez une formule pour U dans chacun des cas suivants :

a) $Q \approx +\infty$
b) $Q > T$
c) $S < Q < T$
d) $Q = S$
e) $Q \approx 0$

5 Multi-Level Feedback Queues Scheduling

Dans cet exercice, on s'intéresse à un ordonnanceur multi-niveaux avec trois *ready queues*. De la plus prioritaire à la moins prioritaire :

— la file Q0 est ordonnancée en round-robin avec un quantum de 8,
— la file Q1 est ordonnancée en round-robin avec un quantum de 16,
— la file Q2 est ordonnancée en premier arrivé premier servi (FCFS).

Les nouvelles tâches sont ajoutées à la file la plus prioritaire. À chaque changement de contexte (i.e. en cas de préemption, ou lorsqu'une nouvelle tâche devient prête) l'ordonnanceur examine chaque file successivement par ordre de priorité et donne la main au premier processus recontré. Chaque fois qu'un processus épuise son quantum il est «préempté» et rétrogradé vers la file suivante.

Exercice Dessinez un chronogramme représentant l'exécution des tâches ci-dessous par notre ordonnanceur.

tâche	A	B	C	D	E
arrivée	0	12	28	36	46
durée	17	25	8	32	18