EXERCICE #1 Problème d'accès concurrent

Deux agences d'une banque veulent mettre à jour le même compte bancaire. Pour cela, l'agence de Nancy effectue :

1. courant = get_account(1867A)
2. nouveau = courant + 1000
3. update_account(1867A, nouveau)

et l'agence de Karlsruhe :

A. aktuelles = get_account(1867A)
B. neue = aktuelles -1000
C. update_account(1867A, neue)

En supposant que l'agence de Nancy commence en premier, quel sera le montant à l'issue des transactions?

Solution. Ça n'a aucun rapport réel avec qui commence la transaction, puisqu'elles ne sont pas atomiques (la partie « En supposant ... en premier » de la question est volontairement piégeuse). Des variables locales + exécutions parallèles entremêlées donnent des résultats différents, par exemple :

— 1 ; 2 ; 3 ; A ; B ; C
⇒ compte inchangé

— 1 ; A ; 2 ; B ; 3 ; C
⇒ compte – 1000

— 1 ; A ; B ; 2 ; C ; 3
⇒ compte + 1000

On a une condition de compétition (*race condition*).
→ Comment garantir le même montant?
⇒ en forçant l'ordre d'exécution.
→ Comment forcer l'ordre d'exécution?
⇒ en rendant les opérations atomiques.
Cette opération est une **section critique**. Elle doit s'exécuter en **exclusion mutuelle**.

Attention! Pour résoudre un problème de mutex, il faut trouver une solution vérifiant les propriétés de **sûreté** (il n'y a qu'un processus en section critique à un instant donné) et de vivacité (un processus souhaitant entrer en section critique le pourra dans un temps fini). Ces propriétés garantissent qu'il n'y a pas de problèmes d'interblocage ou de famine.

Ordonnancement

Définition (Ordonnancement)

La sélection dans le temps des processus ou threads qui peuvent accéder à un processeur est *l'ordonnancement*. Le but est de maximiser :

- Le débit (nombre de processus traités par unité de temps)
- Le taux utile (le taux du processeur effectivement utilisé pour les tâches utilisateurs)

Il y a plusieurs types d'ordonnancement en fonction de la possibilité d'interrompre une tâche :

- *Ordonnancement collaboratif* : les tâches ne sont pas interruptibles.
- *Ordonnancement préemptif* : le système peut interrompre une tâche à tout moment.

Ordonnancement

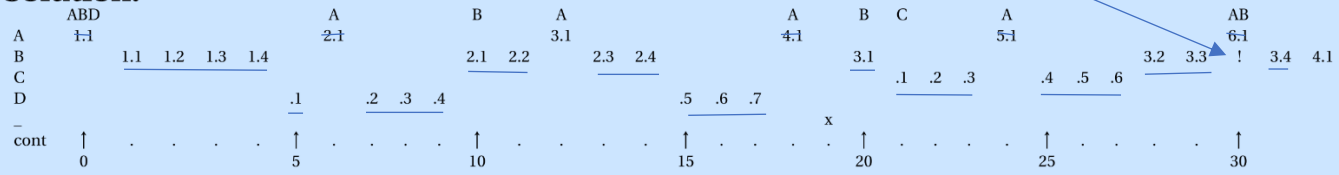
EXERCICE #2 Préemptif avec priorités

Nous utilisons un ordonnancement préemptif avec priorité
Nous allons utiliser un jeu de tâches qui mélange des tâches périodiques et des tâches ponctuelles.

Tâche	Date(s) d'arrivée(s)	Priorité	Durée	Remarque
A	0, 6, 12, 18, 24, 30 ...	10	1	Périodique
B	0, 10, 20, 30 ...	8	4	Périodique
C	21	9	6	Ponctuelle
D	0	1	7	Ponctuelle

1. Faire l'ordonnancement de ces tâches sur 32 unités de temps.
2. Quel est le temps de réponse de chaque tâche ?

Solution.



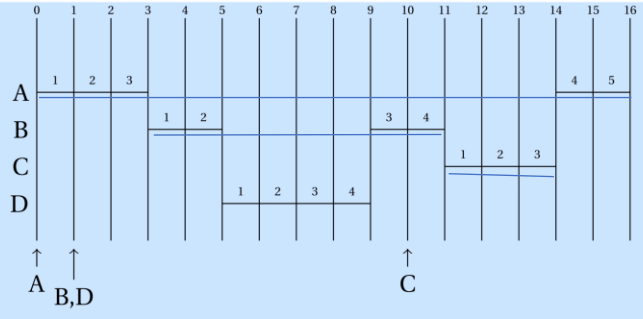
— A : 1
— B : 12 (le pire est la 3e fois). Une échéance est loupée.
— C : 7
— D : 18

EXERCICE # 3 Ordonnancement collaboratif

Le système n'interrompt jamais une tâche, il ne reprend la main que si la tâche se termine ou fait une opération bloquante. Supposons le jeu de tâches suivant :

Tâche	Date(s) d'arrivée(s)	Priorité	Durée	Remarque
A	0	4	5	au bout de 3 unités de calculs, la tâche laisse la main (`yield`) elle est immédiatement prête
B	1	8	4	au bout de 2 unités de calculs, la tâche fait un read bloquant qui dure au moins 3 UC
C	10	9	3	
D	1	5	4	

Faire l'ordonnancement sur 16 unités de temps.



EXERCICE #4 Ordonnancement avec liste de priorité

— Le système gère des listes de priorités.
— On se place en mode préemptif c'est à dire que lorsqu'une tâche plus prioritaire arrive, le système lui donne immédiatement accès au processeur.
— Pour des tâches de même priorité, le système utilise le Round Robin avec un quantum de 2.

Tâche	Date(s) d'arrivée(s)	Priorité	Durée	Remarque
A	0	4	5	
B	1	4	4	
C	3	8	6	

Faire l'ordonnancement sur 15 unités de temps.

Le tourniquet est un algorithme d'ordonnancement courant dans les systèmes d'exploitation et est adapté aux systèmes travaillant en temps partagés.

Une petite unité de temps, appelé quantum de temps, est définie. La file d'attente est gérée comme une file circulaire. L'ordonnanceur parcourt cette file et alloue un temps processeur à chacun des processus pour un intervalle de temps de l'ordre d'un quantum au maximum.

La performance de round-robin dépend fortement du choix du quantum de base.

Solution.