Contrôle terminal Session1 - 15/12/2022

Durée : 2h
Une feuille A4 R/V manuscrite (cours et/ou td) autorisée.

Exercice 1. (6 points)
On considère un système d'exploitation dans lequel les processus sont classés selon deux niveaux de priorité distincts : `high` et `low`. L'algorithme d'ordonnancement utilise une file d'attente par niveau de priorité, la file `HIGH` gère les processus de priorité haute selon l'algorithme `PCTE` (le plus court temps d'exécution) et la file `LOW` gère les processus de basse priorité selon l'algorithme du tourniquet avec un quantum de temps égal à 5 unités de temps.

On considère les processus suivants, définis par leur durée d'exécution, leur date d'arrivée et leur priorité :

Processus	arrivée	durée	priorité
p_1	0	18	`low`
p_2	8	25	`low`
p_3	12	10	`high`
p_4	18	2	`high`
p_5	20	3	`low`
p_6	21	3	`high`

Compte tenu des règles suivantes :

a) un processus de haute priorité interrompt un processus de basse priorité y compris si celui-ci n'a pas épuisé son quantum de temps.

b) la file `LOW` qui gère les processus par tourniquet est mise à jour selon l'algorithme FIFO.

dessiner un diagramme de Gantt correspondant à l'exécution de ces processus et indiquer l'état des listes `HIGH` et `LOW` chaque fois que nécessaire.

Exercice 2. (8 points)
Dans le parc national Kruger en Afrique du Sud, il existe un profond canyon au-dessus duquel une corde a été tendue de façon à ce que les babouins puissent le traverser à bout de bras. Plusieurs babouins peuvent traverser en même temps, pourvu qu'ils aillent tous dans la même direction. Si des babouins qui se dirigent vers l'est et d'autres vers l'ouest se trouvent sur la corde au même moment, cela conduit à un interblocage (les babouins sont bloqués au point de rencontre sur la corde) : en effet, ils n'ont pas la possibilité de passer les uns par-dessus les autres alors qu'ils sont suspendus au-dessus du canyon. Si un babouin souhaite traverser le canyon, il doit vérifier qu'aucun autre babouin ne traverse en sens inverse.

1. Considérons dans un premier temps le cas simple où un seul babouin peut être sur la corde à un moment donné. En prenant la corde comme la ressource partagée, proposer un modèle de code unique pour tout babouin qui souhaite traverser le canyon à l'aide d'un sémaphore. on notera `traverser()` la fonction qui représente la traversée du canyon.
2. À présent, on veut une solution qui permette que plusieurs babouins avancent dans la même direction. La difficulté est que seul le premier babouin de la file doit prendre la ressource et seul le dernier, quand il a fini de traverser, doit rendre la ressource. Au moyen de sémaphores et de variables entières, écrire un programme pour éviter l'interblocage. On ne traitera pas le cas d'un groupe infini de babouins se déplaçant d'un côté et interdisant tout passage à ceux qui se déplacent vers l'autre côté.
   Donner le code `Babouins_de_l_Est` et le code `Babouins_de_l_Ouest`.
3. Montrer que votre solution peut provoquer un phénomène de famine.

4. Modifier le code donné à la question (2) pour éviter la famine en respectant la règle suivante : lorsqu'un babouin qui souhaite traverser le canyon vers l'est arrive à la corde et trouve un babouin qui traverse vers l'ouest, il attend jusqu'à ce que la corde soit vide, mais aucun babouin se déplaçant vers l'ouest n'est autorisé à démarrer jusqu'à ce qu'au moins un babouin ait traversé dans l'autre sens.
   La solution demandée devra utiliser un sémaphore supplémentaire pour gérer l'ordre d'arrivée des babouins, dans une file maintenue par le système.

Exercice 3. (6 points)
Considérons un disque géré à l'aide d'une FAT (*File Allocation Table*). Ce tableau d'entiers possède autant de cases que de blocs sur le disque. Un bloc i est libre si $fat[i] = 0$. Dans les autres cas, il est utilisé.

1. Écrire une fonction qui cherche un espace libre de n blocs consécutifs et renvoie l'indice du premier bloc de cet espace s'il existe ou -1 en cas d'échec en se basant sur la stratégie de premier trouvé (*First Fit*).

   ```
   int alloc_contigue(int fat[], int taille_fat, int n)
   ```

2. Même question pour la stratégie du meilleur ajustement (*Best Fit*).
3. Un descripteur est un enregistrement qui permet l'accès aux données d'un fichier. Pour un fichier donné, si les blocs i et j se suivent, alors $fat[i] = j$. Si le bloc k est le dernier, alors $fat[k] = -1$.

   ```
   typedef struct {
       int taille_en_blocs
       int adr_premier_bloc
       int adr_dernier_bloc
   } descripteur
   ```

   Écrire la fonction `verifier` qui vérifie le codage d'un fichier (descripteur, taille et chaînage) et renvoie 1 en cas de succès et 0 sinon.

   ```
   int verifier(descripteur d, int fat[], int taille_fat)
   ```