Contrôle terminal - Session 1 - 5/1/2022

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

Barème donné à titre indicatif : Ex.1 : 6 - Ex. 2 : 5 - Ex. 3 : 5 - Ex 4 : 4

Exercice 1. On considère les processus suivants, définis par leur durée d'exécution et leur date d'arrivée :

Processus	arrivée	durée
p_1	0	3
p_2	1	6
p_3	4	4
p_4	6	2
p_5	7	1

Dessiner un diagramme de Gantt et indiquer le temps d'attente moyen correspondant à l'application des algorithmes

1. FIFO
2. PCTE (*job le plus court*)
3. PCTER (*job avec temps restant le plus court*)
4. tourniquet avec un quantum de temps fixé à 2 unités de temps.

On prendra le soin de minimiser autant que possible le nombre de commutations de contexte. Si un choix est possible entre un processus arrivant et celui en tête de la file des processus prêts, on privilégiera le premier.

Exercice 2. Soit $T = \{a, b, c, d, e, f, g\}$ un ensemble de tâches soumises aux contraintes de précédence suivantes :

— a précède b, c et d
— b et c précèdent f
— d précède e
— f et e précèdent g.

1. Construire le graphe de précédence associé au problème.
2. Écrire un programme qui prend en compte le graphe de précédence et qui utilise les constructions `parbegin, parend`.
3. Implanter le graphe de précédence avec une exécution parallèle de toutes les tâches et au maximum quatre sémaphores.

Exercice 3. Considérons 5 processus p_i, $1 \leq i \leq 5$ arrivés en même temps et insérés dans cet ordre dans la file d'attente des processus prêts. Ces processus ne font pas d'E/S. Calculer le temps de traitement moyen dans le cas :

1. d'un ordonnanceur circulaire avec un quantum de temps égal à qt sachant que le temps d'exécution des 5 processus est estimé à $2 * qt + r$ avec $r < qt$;
2. d'un ordonnanceur sans préemption fonctionnant selon la discipline premier arrivé, premier servi. On suppose que le temps estimé vaut aussi $2 * qt + r$ avec $r < qt$.

Dans quel cas obtient-on un meilleur temps de traitement ?

Exercice 4. Dans un lavomatique, on cherche une solution pour permettre de répartir les machines à laver équitablement entre les clients. Considérons le programme suivant dans lequel,

pour obtenir une machine, chaque client doit utiliser la fonction `alloue()` et après usage de la machine, il doit utiliser la fonction `libère()` :

Conditions initiales

```
#define NMachines 5;

Semaphore nlibre = sem(NMachines);
int dispo[NMachines] = (1,1,1,1,1);
```

Client

```
alloue() {
   int i;
   P(nlibre);
   for (i=0; i < NMachines; i++)
      if (dispo[i] != 0) {
         dispo[i] = 0;
         return i;
      }
}

libère(int machine) {
   dispo[machine] = 1;
   V(nlibre);
}
```

1. Ce programme peut attribuer la même machine à deux clients. Expliquer pourquoi.
2. Proposer une version de la fonction `alloue()` qui corrige ce problème.