3IN010- Système - Durée : 1 h45min

Mars 2021 - Les documents ne sont pas autorisés

Partiel - Barème indicatif

1. ORDONNANCEMENT (8 PTS)

Soit un système en mode batch (i.e., sans quantum).

On applique une stratégie d'ordonnancement qui consiste à choisir la tâche prête dont le **temps d'exécution passé est le plus petit** (c'est-à-dire la tâche s'étant exécuté le moins). En cas d'égalité, on applique un ordre FIFO.

1.1 (1 point)

Quel est l'avantage de choisir les tâches s'étant le moins exécutée ?

ATTENTION : question qui a « disparu » lors de l'impression

Cela va avantager les taches courtes

Barème : binaire

Soit le scénario suivant :

Tâches	Instant création	Durée d'exécution sur le processeur	Entrées/sorties (E/S)
T1	0 ms	40ms	1 E/S de 10 ms après 5 ms d'exécution
T2	0 ms	30ms	1 E/S de 10 ms après 20 ms d'exécution
T3	30 ms	30ms	Aucune

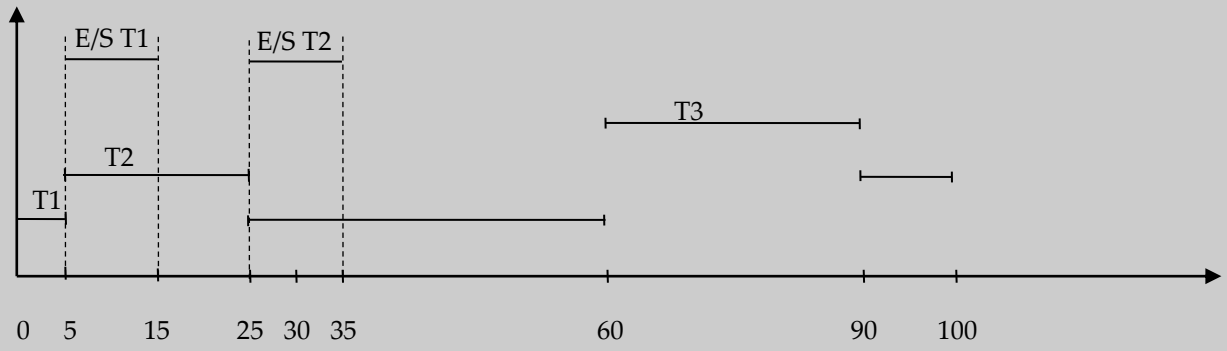
On suppose que la tâche T1 a été créée avant T2.

1.2 (3,5 points)

On considère une stratégie **sans réquisition**.

- Faites un diagramme temporel (Gantt) de l'évolution des tâches.

- Indiquez les temps de réponse des tâches.



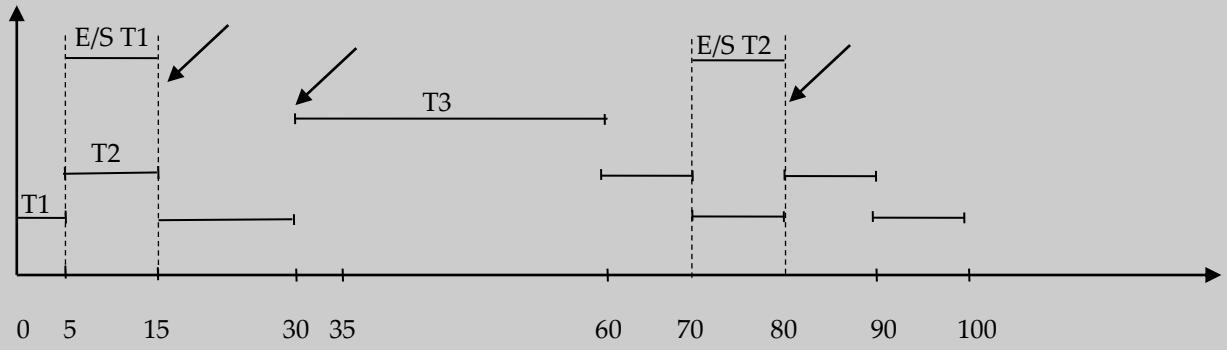
T_R T1 = 60 – 0 = 60 ms
T_R T2 = 100 – 0 = 100 ms
T_R T3 = 90 – 30 = 60 ms

Barème :
70 % Diagramme, 30 % Temps de réponses

On considère maintenant une stratégie **avec réquisition** possible en fin d'Entrée/Sortie ou à la création des tâches.

1.3 (4,5 points)

- Faites un diagramme temporel (Gantt) de l'évolution des tâches avec la réquisition.
- Combien y-a-t-il eu de réquisitions et à quels moments ?
- Indiquez les temps de réponse des tâches.



T_R T1 = 100 – 0 = 100 ms
T_R T2 = 90 – 0 = 90 ms
T_R T3 = 60 – 30 = 30 ms

Réquisition à 15, 30 et 80 ms
Barème :

70 % Diagramme, 15 % réquisition, 15 % Temps de réponses

2. PROCESSUS (6 PTS)

Soit le programme suivant :

```
1:    int main(int argc, char *argv[]) {
2:      int e=0;
3:      int a = 5;
4:      a++;
5:      if (fork() == 0) {
6:          a*=2;
7:          if (fork() !=  0) {
8:                  printf("1 - a : %d\n", a);
9:                   if (fork() == 0) {
10:                         a++;
11:                         printf("2 - a : %d\n", a);
12:                         exit(2);
13:                 }
14:                 printf("3 - a : %d\n", a);
15:                 wait(NULL);
16:                 exit(1);
17:         }
18:         else {
19:                 a+=3;
20:                 printf("4 - a : %d\n", a);
21:                  exit(3);
22:        }
23:     }
24:     else {
25:         wait(&e);
26:         printf("5 - a : %d, e : %d\n", a, WEXITSTATUS(e));
27:     }
28:     return 0;
29:   }
```

On suppose que les appels à fork n'échouent pas.

2.1 (2 points)

Donnez l'arbre des processus fait par ce programme.

```
prog1──prog2─┬─prog3
```

└─prog4

Barème : 33% par proc.

2.2 (2 points)

Donnez l'affichage fait par chacun des processus en précisant l'ordre dans lequel les affichages peuvent avoir lieu.

Affichage :

prog1 : 5 - a : 6, e : 1
prog2 : 1 - a : 12
3 – a : 12
prog3 : 2 - a : 13
prog4 : 4 - a : 15

Ordre affichage : a) prog3 avant prog2 b) prog1 à la fin

Barème :
60% Affichage
40 % Ordre

2.3 (2 points)

On remplace le printf de la ligne 8 par la ligne suivante :

8. execl("/usr/bin/echo", "echo", "1 – a", NULL);

La commande echo se trouve dans /usr/bin et retourne 0.

Donnez le nouvel arbre des processus ainsi que les affichages fait par chacun des processus dans cette nouvelle version du programme.

prog1——prog2——prog3

prog1 : 5 – a : : 6, e : 0
prog2 : 1 – a
prog3 : 4 - a : 15

Barème :
50% arbre
50 % affichage

3. OVERHEAD ET ORDONNANCEMENT (6 POINTS)

L'interruption horloge est la plus prioritaire.

3.1 (1 point)

Expliquez ce qui ce passe si une interruption disque a lieu lorsque le système (noyau) est en train d'exécuter la procédure de traitement d'une interruption horloge ou une partie de son code qui a masqué les interruptions horloges. Quand l'interruption disque sera traitée ?

L'interruption disque restera pendante et sera traiter lorsque l'interruption horloge est démasquée (fin interruption horloge ou démasquage explicite).

Barème :
50% IT pendants
50 % moment du traitement

3.2 (1 point)

Même question si l'inverse arrive : une interruption horloge a lieu lorsque le système (noyau) est en train d'exécuter la procédure de traitement d'une interruption disque ou une partie de son code qui masque les interruptions disque. Quand l'interruption horloge sera traité ?

Le contexte de la procédure de l'interruption disque sera sauvegardée, l'interruption horloge sera traitée immédiatement et lorsqu'elle se termine le noyau reprend l'exécution de l'interruption disque.

Barème :
70 % traitement immediat
30% explication

Nous considérons un système en temps partagé, où la durée du quantum (en ms) est notée ***q***, l'overhead lors de la commutation a une durée ***s*** et le traitement de l'interruption disque a une durée ***d*** (en ms). La commutation est exécutée avec les interruptions horloge masquées.

Soit le scénario suivant :

Tâches	Instant création	Durée d'exécution sur le processeur	E/S
T1	0 ms	50ms	Une seule après 20 ms. Durée E/S 205 ms.
T2	0 ms	200ms	Aucune
T3	0ms	100ms	Aucune

On dispose d'un ordinateur ayant une unité d'échange (UE) travaillant en parallèle avec le CPU. Le processeur est géré selon l'ordonnancement circulaire ou **round-robin**, comme vu en TD, où les tâches sont rangées dans une file unique. Le processeur est donné à la **première** tâche **prête** de la file. Lorsque la tâche élue demande une E/S, elle est insérée en queue de file. Si une tâche arrive au début de la file dans l'état "bloquée" (attente d'une E/S), elle reste en début de file.

On supposera que les tâches sont initialement prêtes et rangées dans l'ordre de leur numéro. On considère aussi : q=100 ms, s= 2ms et d=3ms. Lorsqu'une tâche est élue, le système lui attribue un quantum entier (100ms).

3.3 (4 points)

En considérant vos réponses aux questions précédentes, faites un tableau, en décrivant précisément l'évolution du système : *instant*, *nature de l'événement*, *tâche élue* et *état de la file.* Donnez pour chacune des tâches l'instant où elle termine son exécution.

NB. (1) Nous considérons les événements suivants: chargement tâche, fin quantum, fin commutation, demande d'E/S, interruption disque, fin traitement E/S.

(2) Pour la file indiqué l'état de la tâche : p=prêt ; b=bloqué.

Instant	Evénement	Tâche Elue	Etat de la file	Commentaire
0	Chargement T1	T1	1p 2p 3p	
20	E/S T1		1p 2p 3p	overhead: 2ms
22	Fin commutation	T2	2p 3p 1b	
122	Fin quantum		2p 3p 1b	overhead: 2ms
124	Fin commutation	T3	3p 1b 2p	
224	fin quantum		1b 2p	**fin T3**
225	Interr. disque		1b 2p	Pendante
226	Fin commutation	T2	1b 2p	overhead: 2ms
229	fin E/S	T2	1p 2p	traitement: 3ms
326	Fin quantum		1p	**fin T2**
328	Fin commutation	T1	1p	overhead: 2ms
358		__	vide	**fin T1**

Barème : pas de consigne très précise : ~- 30% si erreur dans prise en compte overhead. – 30 % si erreurs dans les files .