3IN010- Système - Durée : 1 h30min

Mars 20223 - Les documents ne sont pas autorisés

Partiel - Barème indicatif

1. QUESTION DE COURS – INTERRUPTION (3 PTS)

1.1 (0,5 point)

Dans quel mode (utilisateur ou système) est exécuté le traitement d'interruptions ?

Système

1.2 (0,5 point)

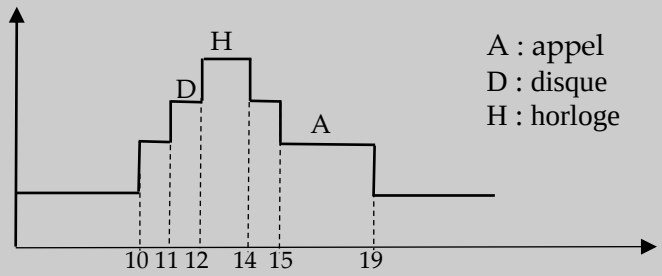
Quel composant matériel d'une machine déclenche l'interruption disque de fin d'entrée/sortie ?

Le contrôleur

1.3 (1 point)

Soit le scénario suivant : on suppose qu'à t=10ms un appel système est appelé, cet appel doit s'exécuter pendant 5 ms, à t=11ms une interruption disque a lieu dont le traitement dure 2ms, à t=12 ms une interruption horloge est déclenchée dont le traitement dure 2ms.

Dessinez un diagramme temporel où sont indiqués les traitements de l'appel système, l'interruption disque et de l'interruption horloge, aussi que les instants où ils se terminent.

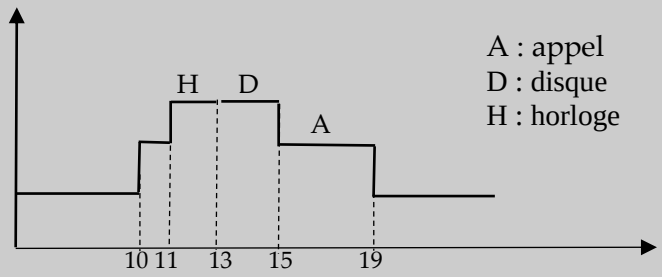


fin appel : 19ms
fin disque : 15 ms
fin horloge : 14 ms

1.4 (1 point)

On considère maintenant le scénario suivant : à t=10ms un appel système est appelé pour une durée de 5 ms, à t=11ms une interruption horloge est déclenchée dont le traitement dure 2ms et à t=12 ms une interruption disque a lieu dont le traitement dure 2ms.

Dessinez un diagramme temporel où sont indiqués les traitements de l'appel système, l'interruption disque et de l'interruption horloge, aussi que les instants où ils se terminent.



fin appel : 19ms
fin disque : 15 ms
fin horloge : 13 ms

2. ORDONNANCEMENT (6 PTS)

On considère un ordonnancement en temps partagé avec un **quantum de 100 ms**.

Soit l'algorithme d'ordonnancement suivant. Le processus élu est celui **ayant le plus grand quantum restant**, en cas d'égalité on choisit la tâche ayant de plus petit identifiant (par exemple T1 avant T2). A leur création le quantum restant des processus est égal à 100. Lorsque tous les processus prêts ont épuisé leur quantum (i.e., leur quantum restant est égal à 0), le système réinitialise les quantums restants de tous les processus prêts à 100.

Soit le scénario suivant :

Tâches	Instant création	Durée d'exécution sur le processeur	Entrées/Sorties (E/S)
T1	0 ms	60 ms	une seule E/S de 20ms après 10 ms d'exécution
T2	0 ms	60 ms	une seule E/S de 20 ms après 40 ms d'exécution
T3	40 ms	110ms	aucune E/S

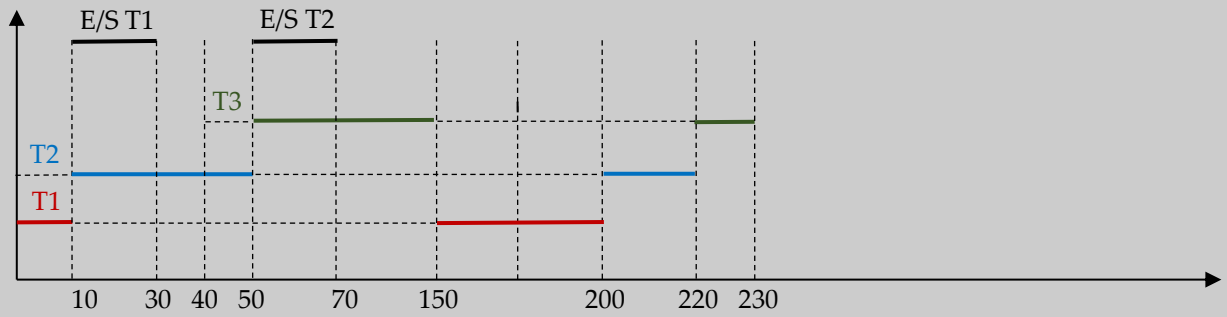
2.1 (3 points)

On considère une stratégie **sans réquisition**.

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

Vous indiquerez également les valeurs des quantums restants au moment où une nouvelle tâche est élue.

- Indiquez les temps de réponse des tâches.



t=10 ms
$QR_1 = 90$

t=50ms
$QR_2 = 60$
$QR_3 = 100$

t=150ms
$QR_3 = 0$

t=220ms
$QR_3 = 100$

$TR_1 = 200 - 0 = 200ms$
$TR_2 = 220 – 0 = 220ms$
$TR_3 = 230-40 = 190ms$

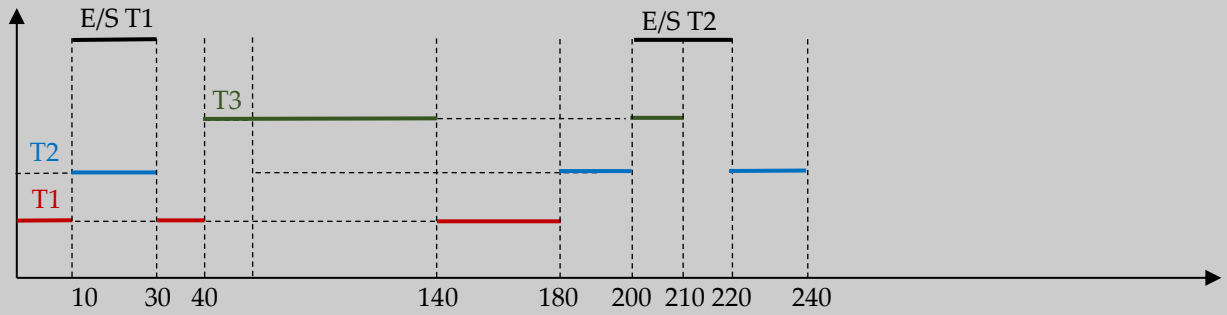
2.2 (3 points)

On considère maintenant qu'une **réquisition** de tâche est possible lors d'une création de tâche ou d'une fin d'E/S.

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

Vous indiquerez les valeurs des quantums restants au moment où une nouvelle tâche est élue ainsi que les instants où une tâche a réquisitionné le processeur.

- Indiquez les temps de réponse des tâches.



t=10 ms
$QR_1 = 90$

t=30ms
$QR_2 = 80$

t=40ms
$QR_3 = 100$
$QR_1 = 80$

t=140ms
$QR_3 = 0$

t=200ms
$QR_3 = 100$

Réquisition à 30 et 40 ms.

$TR_1 = 180 - 0 = 180ms$
$TR_2 = 240 - 0 = 240ms$
$TR_3 = 210 - 40 = 170ms$

3. PROCESSUS (3 PTS)

Soit le programme prog1 dont le code est le suivant :

```
1:   int main() {
2:      int e;
3:      int a = 5;
4:      if (fork() == 0) {
5:            a *= 2;
6:            printf("A  - %d\n", a);
7:            if (fork() == 0) {
```

```
8:              a++;
9:              printf("B  - %d\n", a);
10:             exit(1);
11:         }
12:         execl("./prog2", "prog2", NULL);
13:         printf("C  - %d\n", a);
14:         exit(2);
15:   }
16:   wait(&e);
17:   printf("D - %d, %d\n", a, WEXITSTATUS(e));
18:   return 0;
19: }
```

Le programme prog2.c (ayant pour exécutable ./prog2) est le suivant :

```
20: int main() {
21:   printf("F\n");
22:   return 3;
23: }
```

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

3.1 (1 point)

Donnez l'arbre des processus fait par prog1.

```
prog1 ── prog2 ── prog1'
```

3.2 (2 points)

Donnez les affichages fait par chacun des processus ainsi que les contraintes sur l'ordre dans lequel les affichages peuvent avoir lieu.

père
D – 5, 3

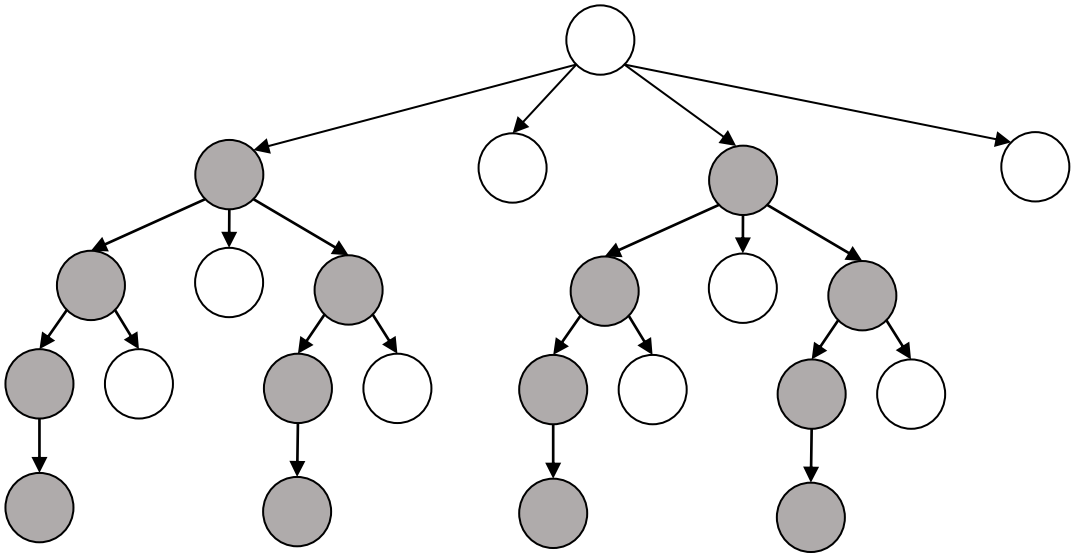
fils
A – 10
F

petit-fils
B - 11

Ordre : fils avant père.

4. PROGRAMMATION PROCESSUS (4 PTS)

On veut écrire une fonction void creer_arbre(int nb) qui crée un arbre de hauteur *nb* ayant la structure suivante. Le processus initial commence par créer *nb* fils. Uniquement chacun des fils **pair** (fils 0, 2, ..) crée ensuite *nb-1* fils et ainsi de suite. Par exemple, l'appel à **creer_arbre(4)** entraînera l'arbre suivant (les processus pair sont indiqués en gris) :



4.1 (4 points)

Donnez le code de la fonction void creer-arbre(int nb).

Version iterative

```
void creerarbre(int nb) {
 int i=0;
 while (i!=nb) {
  if (fork() == 0) {
   if (i%2)
    break;
   i=0;
   nb--;
  }
  else
   i++;
 }
}
```

Version recursive

```
void creerarbre(int nb) {
  int i=0;
  for (i=0;i<nb;i++) {
    if (fork() == 0) {
      if ((i%2) == 0)
        creerarbre(nb-1);
      return;
    }
  }
}
```

5. SYNCHRONISATION (4 PTS)

On considère les quatre tâches suivantes :

A	B	C	D
V(S1) ; P(S3); printf("a");	P(S1); printf("b"); V(S2);	P(S1); printf("c"); V(S2);	P(S2); printf("d"); V(S1); V(S3);

Les sémaphores *S1*, *S2* et *S3* sont initialisés à 0.

5.1 (1,5 points)

A la fin des 4 processus, quels sont les affichages possibles ?

bdac
bdca
cdba
cdab

5.2 (1,5 points)

Sans modifiez les affichages, modifiez ce programme pour systématiquement avoir l'affichage *bdac* (vous pouvez si nécessaire définir de nouveaux sémaphores, dans ce cas n'oubliez pas d'indiquer leurs valeurs initiales).

S4=CS(0)

A	B	C	D
V(S1)	P(S1)	**P(S4)**	P(S2)
P(S3)	...	P(S1)	...
...	V(S2)	.. .	V(S1)
V(S4)		V(S2)	V(S3)

5.3 (1 point)

On suppose, maintenant, que les 4 processus manipulent une variable partagée v. Dans la configuration initiale, on souhaite remplacer les printf par l'incrémentation de cette variable (*v++)*. Par quelles instructions faut-il remplacer les printf ?

Ajouter un sémaphore Mutex=CS(1);

Remplacer les printf par
P(Mutex);
v++;
V(Mutex);