

QCM1: Contrôle continu IF-3-SYS de mars 2024

Pour chacun des mécanismes proposés dans la liste ci-dessous, indiquez si ce mécanisme est implémenté à l'intérieur du noyau.

Veuillez choisir au moins une réponse.

- ☐ Interrupt Service Routine
- ☐ Shell
- ☐ Context Switch
- ☐ Émulateur de terminal
- ☐ Fonction `printf()`
- ☐ Cycle de Von Neumann
- ☐ Fonction `exec()`
- ☐ Ordonnanceur

Pour chaque syscall de la liste ci-dessous, indiquez s'il s'agit d'un appel "bloquant":

- ☐ `getpid()`
- ☐ `exec()`
- ☐ `wait()`
- ☐ `read()`
- ☐ `fork()`
- ☐ `sleep()`
- ☐ `exit()`
- ☐ `write()`

Complétez avec les mots manquants: Un processeur moderne dispose de deux modes d'exécution. Le mode le plus privilégié, dit mode

, est typiquement utile pour exécuter le code des . À l'inverse, le mode dit est moins privilégié, et utile pour exécuter le code des . D'ailleurs, le code ne peut pas accéder directement aux , pour cela il doit invoquer explicitement le en faisant un .

Combien de fois le programme ci-dessous affiche-t-il la lettre A ?

```
void main(void)
{
    for(i=0 ; i<5 ; i++)
    {
        print('A');
        fork();
    }
}
```

Réponse :

Que peut-il arriver si un processus invoque l'appel système `wait()` alors qu'il n'a jamais appelé `fork()` auparavant ? Pour chaque affirmation ci-dessous, indiquez si elle décrit un scénario possible.

Veuillez choisir au moins une réponse.

- ☐ Le noyau suspend l'exécution de notre processus jusqu'à ce qu'un autre processus se termine.
- ☐ C'est un cas d'erreur, donc le noyau tue définitivement notre processus.
- ☐ Le noyau suspend indéfiniment l'exécution de notre processus, il faudra le réveiller manuellement.
- ☐ C'est un cas d'erreur, donc le noyau nous rend la main immédiatement.

Complétez avec les mots manquants: Un système est capable d'exécuter plusieurs même sans disposer d'autant de . L'idée est de partager le temps entre les , selon les décisions d'un composant appelé .

On s'intéresse dans cette question à un ordonnanceur appliquant la stratégie SRTF. Les tâches à exécuter sont indiquées dans le tableau ci-dessous:

	Arrivée	Délai
A	0	12
B	4	9
C	7	5

Au brouillon, dessinez un chronogramme pour représenter l'exécution de ces tâches sur le CPU. Indiquez ensuite l'instant de terminaison de chaque tâche:

A: B: C:

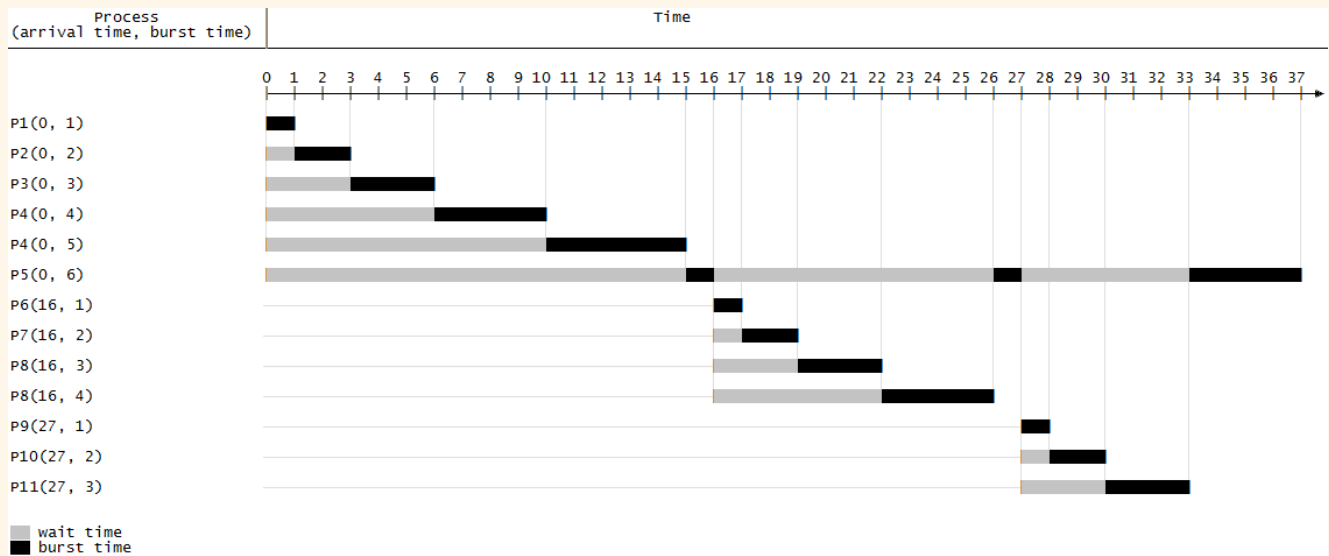
Pour chaque stratégie d'ordonnancement ci-dessous, indiquez si elle est susceptible de causer un problème de famine:

- ☐ SJF
- ☐ SRTF
- ☐ RR
- ☐ MLFQ

Cette question porte sur la notion "d'état" des processus, au sens de l'ordonnanceur. Pour chaque affirmation ci-dessous, indiquez si elle est correcte.

- ☐ Lorsqu'il quitte l'état "BLOCKED", un processus peut se retrouver dans le CPU
- ☐ Lorsqu'il quitte l'état "BLOCKED", un processus peut se retrouver dans le noyau
- ☐ Lorsqu'il quitte l'état "BLOCKED", un processus peut se retrouver dans l'état "RUNNING"
- ☐ Lorsqu'il quitte l'état "BLOCKED", un processus peut se retrouver dans l'état "READY"

L'image ci-dessous représente l'exécution de 11 tâches P1 à P11 sur une machine monoprocesseur: le temps d'attente est représenté en gris, et le temps passé sur le CPU est représenté en noir. Chaque tâche est caractérisée par son instant d'arrivée et par sa durée d'exécution (indiqués entre les parenthèses). Mais quelle stratégie l'ordonnanceur applique-t-il dans cet exemple ? Pour chaque proposition ci-dessous, indiquez si elle vous paraît compatible avec ce chronogramme.



- ☐ Shortest Job First
- ☐ Shortest Remaining Time First
- ☐ Round Robin
- ☐ First Come First Served