

QCM Moodle IF-3-SYS de contrôle continu, mars 2023

Donnez la forme longue de l'abréviation "DMA"

Réponse :

Le rôle du noyau est, entre autres, d'exécuter les instructions assembleur qui composent les applications userland

Veuillez choisir une réponse.

- Vrai
- Faux

Combien de fois ce programme affiche-t-il la lettre "X" ? (rappel: le signe `||` est l'opérateur logique "ou paresseux")

```
main()
{
    if (fork() || fork())
        fork();
    print('X');
}
```

Réponse :

Pour chaque affirmation ci-dessous, indiquez si elle est correcte.

- Chaque appel à `fork()` cause l'ajout d'un nouveau PCB dans la *Ready queue*.
- Un appel système `exec()` causera typiquement une *IO burst*.
- Dans un shell, chaque appel à `fork()` est typiquement suivi d'un appel à `exec()`.
- Appeler `wait()` permet de faire attendre un processus sans suspendre son exécution.

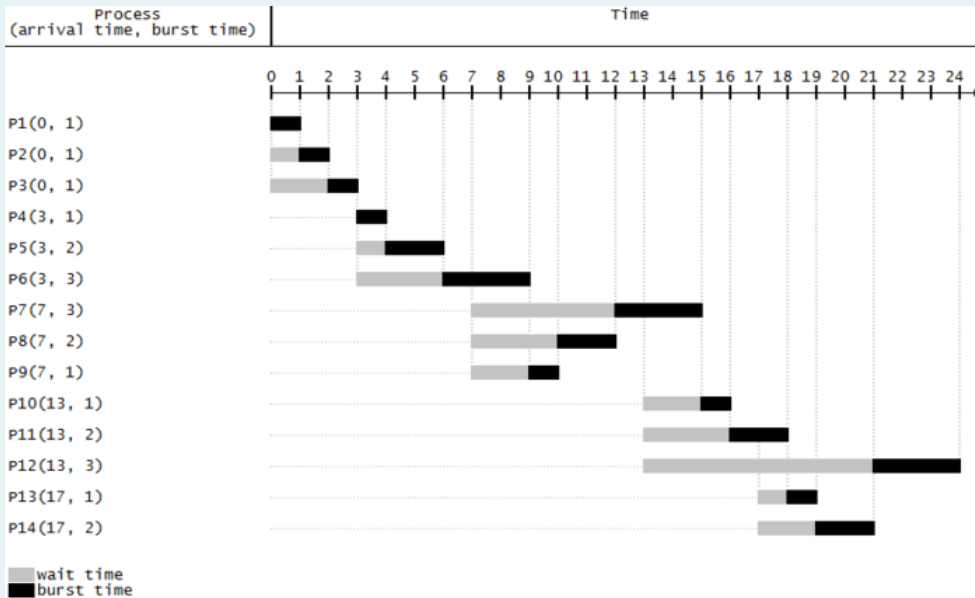
On suppose dans le programme ci-dessous l'existence de deux fonctions pour convertir des nombres d'une représentation à l'autre: `int atoi(char * str)` et `char * itoa(int val)`. On compile ce programme en un exécutable `prog` qu'on lance en tapant `./prog 5`. Combien de fois la lettre Z sera-t-elle affichée ?

```
int main(int argc, char * argv[])
{
    assert(argc==2);
    int N=atoi(argv[1]);
    if(N)
    {
        fork();
        char * param=itoa(N-1);
        exec(argv[0], param);
        print('Z');
    }
    else
    {
        print('Z');
    }
}
```

Réponse :

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 .

L'image ci-dessous représente l'exécution de 14 tâches P1 à P14 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 Remaining Time First
- First Come First Served
- Round Robin
- Shortest Job First

Pour chacune des propositions ci-dessous, indiquez s'il faut compter cette durée lorsqu'on calcule le "temps de séjour" d'un processus.

- Le temps passé à exécuter des instructions sur le CPU.
- Le temps passé dans la "Ready Queue"
- Le temps passé à attendre la terminaison des IO-burst.
- Le temps passé dans l'état "BLOCKED"

Pour chaque affirmation ci-dessous, indiquez si elle est correcte.

- Si l'ordonnanceur est de type "non-préemptif" alors les processus ne sont jamais suspendus.
- Lorsque qu'un processus s'exécute pendant un temps dépassant son "quantum", le noyau le passe dans l'état "READY" pour le faire attendre.
- Lorsqu'il quitte l'état "BLOCKED", un processus peut se retrouver dans le noyau.
- Si tous les processus sont CPU-bound, alors la "Ready Queue" sera presque tout le temps vide.

On s'intéresse dans cette question à un ordonnanceur Round-Robin avec un quantum de 3 unités de temps. Les tâches à exécuter sont indiquées dans le tableau ci-dessous:

nom	arrivée	durée
A	0	10
B	4	11
C	7	7

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: