# QCM Moodle IFA-3-SYS du 12/02/2021

Donnez la forme complète de l'abréviation "PCB"

Réponse :

Dans la liste ci-dessous, indiquez quels appels systèmes sont dits "bloquants"

- [ ] `exec()`
- [ ] `sleep()`
- [ ] `fork()`
- [ ] `exit()`
- [ ] `wait()`

On suppose que les deux programmes ci-dessous ont été compilés en deux exécutables `first` et `second`. Si on tape `./first` sur la ligne de commande, combien de X sont affichés en tout ?

first.c:

```
main() {
   int x = fork();
   if( x ) {
       exec("./second");
       fork();
   }
   printf("X");
   exit(0);
}
```

second.c

```
main() {
    fork();
    printf("X");
    exit(0);
}
```

Réponse :

On s'intéresse à une machine avec P processeurs. Combien de processus peuvent être simultanément dans l'état "READY", au maximum ?

- 2P
- P
- P*P
- indépendant de P

Vrai ou faux: Le rôle de l'ordonnanceur est de choisir quel processus exécuter, alors que le rôle du dispatcher est de charger le contexte d'exécution de ce processus sur le matériel

Sélectionnez une réponse :

- Vrai
- Faux

On s'intéresse à un système avec un seul processeur ordonnancé selon la stratégie "Shortest Remaining Time First". On considère les trois tâches de calcul (i.e. CPU-bursts) décrites par le tableau ci-dessous:

| | A | B | C |
|---|---|---|---|
| **arrivée** | 0 | 4 | 7 |
| **durée** | 10 | 5 | 3 |

Au brouillon, dessinez un diagramme représentant l'exécution de ces tâches, puis calculez le temps d'attente total (Waiting Time) et indiquez-le ci-dessous:

Réponse :

Parmi les stratégies d'ordonnancement ci-dessous, indiquez lesquelles sont, par définition, sans risque de famine.

- [ ] Shortest Job First
- [ ] Round Robin
- [ ] First Come First Serve
- [ ] Shortest Remaining Time First

On s'intéresse à la table de pages d'un certain processus. Parmi les évènements ci-dessous, indiquez lequel/lesquels vont typiquement amener le noyau à modifier le contenu de cette table de pages:

- [ ] Notre processus demande de nouvelles données via l'appel système `read()`
- [ ] Notre processus demande à s'agrandir via l'appel système `mmap()`
- [ ] Notre processus fait un accès mémoire qui cause une "faute de page"
- [ ] Notre processus demande à se dupliquer via l'appel système `fork()`

vrai ou faux: dans un système à mémoire virtuelle, un numéro de page physique et un numéro de page virtuelle occupent forcément le même nombre de bits.

Sélectionnez une réponse :

- Vrai
- Faux

On s'intéresse à un système avec des pages mémoire de 2Kio (2048 octets) et des adresses virtuelles codées sur 32 bits. Combien de ces bits d'adresse servent à coder le *page offset* ?

Réponse :

Cette question se rapporte au TP sur la mémoire virtuelle. Pour chaque affirmation ci-dessous, indiquez si elle est correcte.

- [ ] Grâce à l'appel système `mmap()` on peut consulter le contenu d'un fichier sans avoir besoin de faire appel à `open()`
- [ ] Dans l'exercice sur les nombres premiers, on a utilisé `fork()` et le *copy-on-write* pour obtenir plusieurs exemplaires de notre crible d'Ératosthène.
- [ ] Dans l'exercice sur les nombres premiers, la tableau représentant le crible d'Ératosthène est situé à la même adresse virtuelle dans chacun des processus.
- [ ] Grâce à l'appel système <tt>mmap()</tt> on peut consulter le contenu d'un fichier sans avoir besoin de faire appel à <tt>read()</tt><br>