Examen - Systeme d'Exploitation Avance 2 (Simulation)

Exercice 1 : (6 pts)

Objectif : Assurer la synchronisation entre trois processus A, B et C pour garantir un ordre d'exécution précis.

Contrainte :

1. a1 doit être exécutée avant b1.
2. b2 doit être exécutée avant c1.
3. c2 doit être exécutée avant a2.

Utilisez des sémaphores pour réaliser la synchronisation.

Déclarez :

sem_a1_done = 0 ; sem_b2_done = 0 ; sem_c2_done = 0

Algorithme attendu :

- Processus A : a1 -> up(sem_a1_done) -> down(sem_c2_done) -> a2
- Processus B : down(sem_a1_done) -> b1 -> b2 -> up(sem_b2_done)
- Processus C : down(sem_b2_done) -> c1 -> c2 -> up(sem_c2_done)

Explication : Chaque semaphore agit comme un signal de synchronisation entre les processus, assurant l'ordre imposé.

Exercice 2 : (7 pts)

Objectif : Communication via mémoire partagée entre un processus père et un processus fils.

Scénario :

Le père écrit 15 entiers dans une mémoire partagée. Le fils lit et affiche uniquement les entiers impairs.

Contraintes :

- Le père doit écrire un entier puis attendre que le fils le lise avant d'écrire le suivant.
- Utiliser deux sémaphores pour synchroniser les opérations d'écriture et de lecture.

Examen - Systeme d'Exploitation Avance 2 (Simulation)

Explication :

Un sémaphore (sem_write) initialise à 1 pour permettre au père d'écrire. Un autre (sem_read) à 0 pour obliger le fils à attendre l'écriture.

Exercice 3 : (7 pts)

Objectif : Gérer une ressource limitée entre plusieurs threads via sémaphores.

Scénario :

Un parking dispose de 3 places. 10 voitures (threads) arrivent à des moments différents. Une voiture ne peut entrer que si une place est libre.

Contraintes :

- Utiliser un sémaphore initialisé à 3 pour représenter les 3 places disponibles.
- Afficher les étapes : arrivée, entrée dans le parking, sortie.

Explication :

Chaque thread utilise sem_wait pour réserver une place, et sem_post lorsqu'il quitte le parking, libérant la place.