TP 7

SYSTEMES D'EXPLOITATION

INFORMATIONS GÉNÉRALES

- Travail en groupe est possible .
- Délivrables:
  - Manuel de votre programme en PDF
  - Code source commenté et fonctionnel (avec Makefile!)
  - Dossier à rendre :
    <Prénom1>.<Nom1>.<Prénom2>.<Nom2>.TP7.zip(ou tar.gz)

INFORMATIONS GÉNÉRALES

- Références au cours :
  - 10. Mémoires partagées
- Objectifs :

L'objectif général du TP est de créer deux programmes qui utilisent la mémoire partagée pour accomplir une tâche coopérative

  - Initialiser et utiliser mémoire partagée
  - Utiliser des sémaphores pour éviter les conditions de course

LES PROGRAMMES

- Contexte:
  - Une pizzeria automatisée avec 2 robots: un cuisinier et un serveur
  - Une commande de 10 pizzas
- Programme: serveur.c
  - Quand il n'y a pas de pizza sur l'étagère, le serveur attendra.
  - Lorsque les pizzas sont disponibles, le serveur commence à servir, et de décrémenter le nombre de pizzas sur l'étagère.
  - Le serveur s'arrêtera quand il aura servi toutes les pizzas (10).
- Programme: cuisinier.c
  - Le cuisinier a besoin d'un temps aléatoire pour cuire une pizza et la mettre sur l'étagère (et incrémenter le nombre de pizzas sur l'étagère)
  - Lorsqu'il y a trois pizzas sur l'étagère, le cuisinier se repose.
  - Le cuisinier s'arrêtera après 10 pizzas

MÉMOIRE PARTAGÉE

- Dans notre cas, l'étagère est la mémoire partagée entre les programmes.
- Le segment de mémoire partagée sera créé par le cuisinier
- Lorsque le travail est terminé, le cuisinier et le serveur dissocient la mémoire partagée. Avant que le serveur se ferme, il supprime le segment de mémoire partagée.

SÉMAPHORES

- Pour réaliser le synchronisation de serveur et cuisinier on peut utiliser des sémaphores, par exemple:
  - Un pour l'exclusion mutuelle d'étagère (pour éviter les conditions de course!)
  - Un pour informer le serveur de commencer à servir
  - Un pour informer le cuisinier de se reposer
- Vous pouvez utiliser moins de 3 en vérifiant directement la valeur de la mémoire partagée mais pour ce TP utilisez au moins un

CONSEILS

- Exemple memoire partagée:
  - http://cui.unige.ch/~chanel/prez/presentations/sys-info/10.sharedmem/#/8
- Exemple sémaphore:

```
#include <semaphore.h>
int main(int argc, char *argv[]){
    const char * smph_name= "mutex";
    // mutex for mutual exclusion of shared memory.
    // What should the initial value be?
    sem_t mutex = sem_open(smph_name,O_CREAT,0666, ???);
    while(condition){
        // decrement (lock) the semaphore pointed to by mutex.
        sem_wait(mutex);
        // do operation here like manipulate the data in shared memory
        // then  increment  (unlock) the semaphore pointed to by mutex.
        sem_post(mutex);
    }
    sem_close(mutex);
}
```

  - Regardez les pages manuel: sem_open, sem_wait, sem_post !

CONSEILS

- Vos deux programmes (cuisinier et serveur) devraient imprimer des informations utiles:
  - chaque fois que le cuisinier fait une pizza
  - Les pizzas restantes à faire
  - chaque fois que le cuisinier se repose
  - quand le cuisinier a fini
  - chaque fois que le serveur retire une pizza de l'étagère
  - chaque fois que le nombre de pizzas sur l'étagère change
  - Quand l'étagère est pleine

CONSEILS

- Pour votre implémentation le cuisinier peut faire une pizza en nombre entier de secondes entre 2 et 5
- Exemple: supposer que le cuisinier peut faire une pizza en 1 ou 2 secondes
  - `sleep(rand()%2+1); // The cook is making a pizza`