RAPPORT

Pour ce TP, on nous demandait de creer 2 programmes qui interragissent un peu comme le server-client mais grace a la memoire partagee. On cherche donc a creer une relation cuisinier-serveur dans un restaurant de pizza.

COOK

Dans ce programme, le cuisinier doit cuire des pizzas dans un rythme de 1 a 2 secondes par pizzas. Il peut cependant ne faire que 3 pizzas jusqu'a ce que l'etagere soit pleine, il va alors prendre une pause a ce moment la jusqu'il y est de la place libre sur l'etagere.

On va commencer par initialiser les semaphores:

```
// Seting up semaphores
sem_t *serve = sem_open(SERVE, O_CREAT, 0666, 0);
sem_t *cook = sem_open(COOK, O_CREAT, 0666, 3);
sem_t *mutex = sem_open(MUTEX, O_CREAT, 0666, 1);
sem_t *done = sem_open(DONE, O_CREAT, 0666, 0);
```

Ensuite on va pouvoir creer et initialiser un segment de memoire partagee tel que:

```
// Shared memory
int fd = shm_open(SMH, O_CREAT | O_RDWR, 0666);
```

> Creation de la shared memory, avec sont descipteur de fichier `fd`

```
// Memory size
if (ftruncate(fd, sizeof(int)) == -1)
```

> On gere la taille de la memoire partagee

```
// File mapping, init to 0
int *smh = mmap(NULL, sizeof(int), PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
```

> Enfin, on va mapper la memoire partagee pour qu'on puisse l'utiliser dans notre programme

On va donc pourvoir commencer notre boucle qui sera notre cuisinier. Cette boucle va **lock/unlock** les semaphores pour qu'on puisse faire notre programme. L'incrementation des pizzas sera faite entre les 2 `mutex` et entre chaque pizza on aura 1 a 2 secondes de pause.

On va bien evidemment afficher le plus de status possibles tel que le nombre de pizza faites, combien il en reste, etc...

Enfin, on aura la quantitee maximum qui va etre decremantee jusqu'a arriver a 0, ce qui terminera notre programme et fermera tout pour que l'on est aucun probleme de memoire a la suite de ce programme.

WAITER

On va proceder exactement de la meme facon sauf que on va utiliser une semaphore en plus:

```
sem_t *done = sem_open(DONE, O_CREAT, 0666, 0);
```

qui va permettre de savoir quand le cuisinier a fini ses pizzas sans avoir a hardcode encore une autredecrementation dans la boucle. On aura notre `condition` de sortie initialiser a 0 au debut.

Dans la boucle on commence de la meme facon que dans `cook.c`, on decremante le nombre de pizza surl'etagere (memoire) entre les 2 mutex puis on finit par recuperer la valeur de `done` et on la met dans`condition`. On continue ceci t'en que la condition est nulle et que le nombre de pizza n'est pas nul.

On ferme tout a la fin.

Utilisation

Sur un premier terminal:

```
make
./cook
```

Sur un deuxieme terminal:

```
./waiter
```