# TP 6 (3h): Partage de la mémoire, threads

*Remarque : Utiliser les chapitres 4 &5 de* Advanced Linux Programming*, ainsi que les pages de man comme support*.

**Programme fourni :** Dans *matrix.[ch]* vous trouverez les éléments suivants :

* Type SquareMatrix permettant de stocker une matrice carrée de double
* Des fonctions pour saisir/afficher une matrice, calculer son carrée et pour libérer la mémoire
* Un exemple de programme principal

**Assurez-vous de bien comprendre le fonctionnement de ces fonctions avant de passer à la suite !**

1. **Partage de mémoire entre deux processus**

L’objectif est d’écrire deux programmes qui communiquent le contenu d’une matrice par mémoire partagée.

Les deux programmes commencent par créer/accéder la mémoire partagée (**shmget**) à partir d’une même clé, générée à partir du chemin du répertoire de travail (fonctions **ftok** et **getcwd**). *Le répertoire de travail à l’exécution devra donc être le même pour les deux processus.*

*Attention* : ne pas oublier d’attacher le segment dans l’espace mémoire avec **shmat**.

Le fonctionnement par la suite est le suivant :

* P1 :
  + Exécute la saisie de la matrice et la place dans la mémoire partagée
  + Se met en attente pour 15s (**sleep**)
  + Détache la mémoire partagée (**shmdt)**
* P2 :
  + Affiche le contenu de la matrice à partir de la mémoire partagée
  + Calcule et affiche le carré de la matrice
  + Détache et libère la mémoire partagée (**shmdt**, puis **shmctl** avec commande IPC\_RMID)

Questions:

1. Entre le moment où P2 se termine et la fin de P1, dans un autre terminal, lancez la commande **ipcs** (vous avez 15s !). A la fin de P1 lancez-la de nouveau. Que pouvez-vous conclure ?
2. Dans P1, inversez l’ordre de **sleep** et **shmdt**. Répétez le point a). Quelle conclusion maintenant ?
3. Toujours dans P1, ajoutez à la fin (après le **sleep**) un nouvel appel à **shmat** pour essayer d’attacher la mémoire à nouveau, avec traitement d’erreurs (perror). Que se passe-t-il ?

A la fin de l’exercice, vérifiez avec **ipcs** s’il reste des segments non-libérés et les libérer avec la commande **ipcrm**.

1. **Création de threads**

Reprendre les deux programmes de l’exercice 1 et les réécrire sous la forme de threads d’un seul et même programme. Plus précisément, le programme crée deux threads, en plus du thread principal (avec **pthread\_create**) :

* Le thread 1 exécute la saisie de la matrice
* Le thread 2 calcule le carré
* Le thread principal affiche matrice au carré et libère les deux matrices

A noter :

* Utiliser l’option **-lpthread** de **gcc** pour l’édition de liens.
* Le comportement de chacun de ces deux nouveaux threads doit être encapsulé dans une fonction à part.
* Pour que les actions se fassent dans le bon ordre, vous pouvez vous servir de **pthread\_join** (et supprimer les appels à **sleep**). Ainsi :
  + les 2 threads sont lancés au début du programme principal,
  + le thread 2 attend la terminaison du thread 1,
  + le thread principal attend la terminaison du thread 2

*Attention : Pour cela vous aurez besoin de passer l’identifiant du thread 1 en paramètre au thread 2.*