TP 3: Exclusion Mutuelle par Sémaphore

Écrire un programme qui initialise une variable globale entière ***valeur_globale*** à 1 et crée deux threads tache1 et tache2. Le thread tache1 incrémente ***valeur_globale*** 100000 fois et le thread tache2 décrémente value globale 100000 fois, en y intégrant un sémaphore pour résoudre le problème de l'accès concurrent à la variable value globale. .

•Les services Posix de manipulation des sémaphores se trouvent dans la librairie <semaphore.h>.
•Le type sémaphore est désigné par le type sem_t.

Initialisation d'un sémaphore

int sem_init(sem_t *sem, int pshared, unsigned int valeur)

- ***sem :*** est un pointeur sur le sémaphore à initialiser;
- **Valeur :** est la valeur initiale du sémaphore ;
- **pshared :** indique si le sémaphore est local au processus (pshared=0) ou partage entre le père et le fils pshared ≠0.

int sem_wait(sem_t *sem) : est équivalente à l'opération P (S) et retourne toujours 0.
int sem_post(sem t *sem): est équivalente a l'opération V(S) : retourne 0 en cas de succès ou -1 autrement.

```
int pthread_create ( pthread_t *thread , pthread_attr_t *attr, void *nomFonct, void *arg   );

void    pthread_join( pthread_t tid, void * *status);
```

La fonction **pthread_create** crée un nouveau thread.

Syntaxe:

```
int pthread_create ( pthread_t *thread , pthread_attr_t *attr, void *nomFonct, void *arg   );
```

- Attend la fin d'un thread.

```
void    pthread_join( pthread_t tid, void * *status);
```