Manipulation des threads (*utilisation de la norme POSIX*)

① int pthread_create(

pthread_t *tid, // sert à récupérer le TID du thread créé

const pthread_attr_t *attr, // sert à préciser les attributs du thread (taille de la pile, priorité….)

//attr = NULL pour les attributs par défaut

void * (*func) (void*), // est la fonction à exécuter par le thread

void *arg); //le paramètre de la fonction.

Cette fonction permet de créer un nouveau thread, elle renvoie 0 s'elle réussit, sinon elle renvoie une valeur non nulle identifiant l'erreur qui s'est produite.

② void pthread_join(pthread_t tid, void *status); // sert à récupérer la valeur de retour et l'état de terminaison.

Cette fonction faire un attente d'un thread. L'équivalent de waitpid des processus sauf qu'on doit spécifier le TID du thread à attendre.

③ void pthread_exit(void * status); Termine l'exécution d'un thread.

④ void pthread_self(void); retourne l'identifiant, TID, du Thread courant.

Remarque: la ligne de commande suivante permet de compiler un programme multi-thread :

gcc -o nomprog nomprog.c –lpthread

Note (Description des sémaphores):

```
int sem_init(sem_t *semaphore, int pshared, unsigned int valeur)
```
Création d'un sémaphore et préparation d'une valeur initiale.
```
int sem_wait(sem_t * semaphore);
```
Opération P sur un sémaphore.
```
int sem_post(sem_t * semaphore);
```
Opération V sur un sémaphore.