Module : SE2 **TP N°1**

Le but ce ce TP est l'introduction aux threads POSIX. On utilisera les fonctions suivantes :

#include <pthread.h>

int **pthread_create** (pthread_t **thread,* const pthread_attr_t **attr,*
void **(*routine)(void*),* void **arg*);

int **pthread_exit** (void *value_ptr)
int **pthread_join** (pthread_t **thread,* void **value_ptr)

- pthread_create : crée un thread, et renvoie 0 si la création s'est bien déroulée, ou le code de l'erreur sinon. Elle reçoit en argument :

* le TID (Thread Identifier) du thread,
* une constante (dans notre TP elle prendra la valeur NULL)
* un pointeur vers la fonction exécutée par le thread
* un argument de la fonction

- pthread_exit : permet de quitter un thread, elle ne renvoie aucune valeur, et attend un seul argument qui est l'adresse de la variable à renvoyer au programme qui a appelé le thread.

- pthread_join : bloque l'appelant en attente de la fin du thread passé en 1er argument, alors que le 2nd argument est un pointeur de pointeur qui servira à récupérer l'adresse de la valeur renvoyée par pthread_exit du thread, cette fonction renvoie 0 si le thread se termine correctement, sinon elle renvoie le code de l'erreur.

Exercice 1 : Editer, compiler puis exécuter le fichier exo1.c. Que fait ce programme ?
Pour compiler : **gcc exo1.c -pthread**

Exercice 2 : Editer, compiler puis exécuter le fichier exo2.c. Que fait ce programme ?

Exercice 3 : Modifier le programme exo2.c pour qu'il créé N threads (N à lire), affichant chacun son numéro (1 à N)

Exercice 4 : Editer, compiler puis exécuter le fichier exo3.c. Que fait ce programme ?

Exercice 5 : Modifier le programme exo3.c pour qu'il créé deux autres threads calculant respectivement le triple et la factorielle de nb.

```
/* Exo1.c Programme qui crée un thread sans passage de paramètres */
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h> //pour les fonctions de thread
void *Affiche(void *arg); //prototype de fonction
pthread_t tid; //déclare un identifiant de thread
int main () {
  pthread_t tid; //déclare un identifiant de thread
  int ret;
  printf("Ceci est le main: pid = %d \n\n", getpid());
       /* création d'un thread avec l'identifiant trd, qui va exécuter la
     fonction Affiche sans passage de paramètres */
  ret = pthread_create(&tid, NULL, Affiche, NULL);
  if (ret < 0) {
     perror("Erreur de pthread_create \n");
     exit(1);
     }
  /* attente de la fin du thread sans retour de résultat */
  pthread_join(tid, NULL);
return 0;
} //fin du main

void *Affiche(void *arg)
{
printf("Je suis le thread fils , mon pid = %d\n\n", getpid());
pthread_exit(NULL); //termine l'exécution du thread sans retour de résultat
} //fin de MyThread


**************************************************************************
/* Exo2.c Programme qui crée un thread avec passage de paramètre */
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h> //fonctions Posix pour les threads
#define N 3 //déclaration d'une constante N=3
void *Affiche(void *arg);
int main () {
   pthread_t tid[N];
   int ret, i;
   /* création de thread */
   for(i=0;i<N;i++) {
   ret = pthread_create(&tid[i], NULL, Affiche, (void*)(long int)i);
   if (ret < 0) {
      perror("Erreur de pthread_create \n");
      exit(1);
   }
}
   /* attente de la fin du thread */
   for(i=0;i<N;i++)
   pthread_join(tid[i], NULL);
return 0;
} //fin du main

void *Affiche(void *arg)
{
```

```
  long int i = (long int) arg;
   printf("Je suis le thread %ld \n\n\n", i);
   pthread_exit(NULL); //termine l'exécution du thread sans retour de
    résultat
} //fin de MyThread
```

```
/*Exo3.c Programme qui crée un thread avec passage de paramètre */
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

void *ledouble(void *arg);

int main(){
   void *doubl;
   int nb;
   pthread_t tid;

/* lecture de nb au clavier */
   printf("\n\nDonnez un entier : ");
   scanf("%d", &nb);
/*crÉer un thread */
   pthread_create(&tid, NULL, ledouble, (void *)(long int)nb);
/* attendre la fin du thread */
   pthread_join(tid, &doubl);
   printf("\nLe double de %d est %ld \n\n\n",nb, (long int) doubl);
return 0;
}
void * ledouble(void * arg) {
   long int d=(long int) arg;
   d=2*d;
return (void*)d;
}
```