INSA RENNES

Département informatique

TP Parallélisme no 1
Pthreads

But du TP

gcc -o main -lpthread (...).c
./main

Utilisation des threads POSIX, mise en oeuvre d'un algorithme producteur/consommateur simple.

Mon premier programme

Le programme suivant lance un thread POSIX qui s'exécute en "parallèle" avec le main du programme. Compiler et exécuter ce programme. Peut-on observer du parallélisme ? Pourquoi ?

```
#include <stdio.h>   /* standard I/O routines */
#include <pthread.h> /* pthread functions and data structures */
/* function to be executed by the new thread */
void* do_loop(void* data)
{
  ... (a vous d'y mettre qqchose)
  /* terminate the thread */
  pthread_exit(NULL);
}
int main(int argc, char* argv[])
{
  int thr_id;          /* thread ID                    */
  pthread_t  p_thread; /* thread's structure           */
  int a = 1;           /* thread 1 identifying number */
  int b = 2;           /* thread 2 identifying number */
  pthread_attr_t attr; /* thread attributes            */

  /* create a new thread that will execute 'do_loop()' */
  pthread_attr_init(&attr); /* initialize attr */
  thr_id = pthread_create(&p_thread, &attr, do_loop, (void*)&a);

  /* run 'do_loop()' in the main thread as well */
  do_loop((void*)&b);

  return 0;
}
```

Modifier ce programme pour ajouter un délai d'une seconde entre chaque affichage.
Modifier ensuite le programme initial (sans les délais) pour avoir exactement 1 entrelacement d'une action de chaque thread.

Producteur/consommateur

Ecrire les fonctions `put(int v)` et `int get()` de gestion d'un tampon circulaire à N cases permettant de réaliser un programme producteur/consommateur. Ces fonctions doivent gérer la synchronisation (exclusion mutuelle et attente si le tampon est plein (resp. vide). On utilisera les sémaphores d'exclusion mutuelle et les conditions (opérations wait/notify/notifyall).

Ecrire un programme permettant de tester ces fonctions. Ce programme sera un mini interpréteur de commande qui lancera à la demande des threads producteurs ou des threads consommateurs et qui permettra également de visualiser l'état du système (contenu du tampon, nombre et identité des threads bloqués). Un thread producteur (resp. consommateur) produira (resp. consommera) N/2 éléments du tampon 1 par 1 en faisant appel aux fonctions put et get.