Descripción Práctica 6
Programación Concurrente
15-I
Comunicación y sincronización de tareas usando POSIX

La librería POSIX (Portable Operating System Interface y X viene de UNIX) incluye los mecanismos para el manejo de hilos y su sincronización. Incluir en un programa C lo siguiente:

#include <pthread.h>

La **declaración** de un candado **mutex** en POSIX se hace de la manera siguiente:

pthread_mutex_t mutex;

La **inicialización** de un candado en POSIX se realiza con la función pthread_mutex_init, de la manera siguiente:

pthread_mutex_init (&mutex , (void *) attr);

esta función Inicializa el candado **mutex** con valor 0 (candado abierto). Por medio de **attr** se le pueden pasar atributos al hilo los cuales pueden ser NULL en caso de no necesitarlos.

La operación lock para manipular un candado se invoca de la siguiente manera:

pthread_mutex_lock (&mutex);

Con esta operación un hilo puede solicitar acceso exclusivo para ejecutar una sección crítica protegida por el candado mutex.

La operación unlock para manipular un candado se invoca de la siguiente manera:

pthread_mutex_unlock (&mutex);

Con esta operación se libera (abre) el candado **mutex** que protege al código de la sección crítica.

1. A) Escribe el siguiente programa “hola_hilo.c” que utiliza las operaciones vistas en clase para crear un hilo con la librería pthread.h

```
#include <stdio.h>
#include <pthread.h>

void mensaje(void *ptr)
{
  int id,dato;
  id = (int )ptr;
  printf("Hola soy  = %d\n",id);
```

```
   printf("salgo hilo\n");
   pthread_exit((void *)NULL);
}

main()
{
  pthread_t hilo1;
  int idh;

  idh=1;
  pthread_create(&hilo1,NULL,(void *)&mensaje,(void *)idh);
  pthread_join(hilo1,(void *)NULL);
  return 0;
}
```

B) compila y corre el programa de la siguiente manera

$gcc hola_hilo.c -o holah -lpthread

ignora los AVISOS (warnings) referentes a las conversiones de apuntadores

$./holah

2. Escribe un programa “hola_hilo2.c” en donde se generen N hilos y cada hilo se identifique con un número entre 0 y N-1

3. Escribe un programa “hola_hilo3.c” en donde se generen N hilos y cada hilo se identifique con un número entre 0 y N-1. Cada hilo va a generar un número aleatorio entre 0 y 50, luego se sincronizarán utilizando un candado para que una variable compartida **maximo** sea actualizada por cada hilo y solo guarde el máximo de los valores generados. Al final el hilo main desplegará el valor de **maximo** obtenido.

Nota: Las variables compartidas y los candados se tienen que declarar como variables globales, fuera de todo módulo, colóquenlas abajo de los #include para visualizarlas con facilidad. NO usar arreglos.

4. Escribe un programa “prod_cons.c”, Productor-Consumidor en donde el productor genera infinitamente números consecutivos módulo 100 y el consumidor los despliega en pantalla.

Nota: al igual que en OpenMP, aquí los candados deben ser inicializados de manera apropiada por el hilo main, antes de crear a los hilos productor y consumidor.