Sistemas Operativos I - LCC
Práctica 1
Introducción a Procesos e Hilos

Introducción a POSIX Threads

POSIX Threads (o PThreads) es un estándar POSIX para la creación, manejo y destrucción de hilos. Todas las funciones se encuentran en la librería con *header* pthread.h.

Para compilar un programa C usando POSIX threads debe indicar al compilador gcc que *linkee* la librearía POSIX Threads con la opción −lpthread.

Manejo de Threads

La librería *PThread* define un nuevo tipo de datos en C llamado pthread_t para representar a los threads.

Creation de Threads

Para la creación de *Threads* utilizaremos la función pthread_create con la siguiente signatura:

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

Donde toma como argumentos

- un elemento pthread_t *thread que apunta al nuevo hilo creado,
- un puntero a una estructura de atributos const pthread_attr_t * attr utilizado al momento de crear el thread, en el caso que sea NULL se utilizan valores por defecto,
- la rutina a ejecutar,
- posibles argumentos que necesite la rutina para la ejecución.

En el caso que el llamado a pthread_create retorna 0, y en caso de fallar un numero de error dejando además el argumento thread indefinido. Los errores pueden encontrarlos en el manual de pthread_create.

Terminación de Hilos

Para la terminación de Hilos se utiliza la función pthread_join **bloqueante** que espera a que un hilo dado termine, y sí ya terminó continua inmediatamente.

```
int pthread_join(pthread_t thread
                 , void **retval);
```

Que toma dos argumentos: el hilo a esperar, pthread_t thread, y un puntero a un puntero a void, void **retval, que se utiliza para alojar el resultado de la ejecución del hilo thread.

En el caso que el llamado sea exitoso, se retorna 0, mientras que en el caso de fallar retorna un numero de error. Los números de error los pueden encontrar en el manunal de pthread_join.

Para obtener el valor a través de retval es necesario utilizar pthread_exit. Buscar en manual: man 3 pthread_exit

Pequeño Ejemplo de Creación de Hilos

Como ejemplo se muestra la creación de dos hilos, que simplemente muestran un número en pantalla, y se espera a que terminen. ¿Qué pasa cuando se lo ejecuta repetidas veces?(**while true**; **do** ./a.out; **done**).

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

void* printEcho(void *i){
  /* Interpretamos la entrada como un entero */
  int arg = *((int*)i);
  /* Mostramos un mensaje */
  printf("ECHO: %d!\n",arg);

  return NULL;
}

int main(int argc,char **argv){
  pthread_t thread_id[2];
  int arg1=1, arg2=2;
  void *res;

  printf("Creamos dos hilos!\n");

  /* Creamos dos hilos */
```

```
  assert(! pthread_create(&thread_id[0], NULL, printEcho, (void*)&arg1));
  assert(! pthread_create(&thread_id[1], NULL, printEcho, (void*)&arg2));

  printf("Esperamos a que terminen...\n");

  /* Esperamos a que terminen */
  assert(! pthread_join(thread_id[0], &res));
  assert(! pthread_join(thread_id[1], &res));

  printf("Terminamos!\n");

  return 0;
}
```

Ejercicios

El Jardín ornamental

En un jardín ornamental se organizan visitas guiadas y se desea contar cuánta gente entra por día. Hay dos molinetes, uno en cada una de las dos entradas y se ha implementado el sistema para contar los visitantes utilizando *PThreads*.

Se les provee un esqueleto de código que acompaña a éste ejercicio en la carpeta de *Ejemplos*.

- Por cada molinete entran NVisitantes personas, pero al ejecutar el programa verán que el resultado no es 2∗NVisitantes. ¿Por qué?.
- Ejecute el programa 5 veces con NVisitantes igual a 10. ¿El programa dio el resultado correcto siempre? Si esto es así, ¿por qué?
- ¿Cuál es el mínimo valor que podría imprimir el programa? Simular dicha situación.
- Implementar la solución vista en la clase de teoría.
- Implemente una solución utilizando un *mutex*.

Cena de los Filósofos (Dijkstra)

Cinco filósofos se sientan alrededor de una mesa redonda y pasan su vida comiendo y pensando. Cada filósofo tiene un plato de fideos y un tenedor a la izquierda de su plato. Para comer los fideos son necesarios dos tenedores y cada filósofo sólo puede tomar los que están a su izquierda y derecha. Primero toman el que está a su derecha y luego el que está a su izquierda. Si cualquier filósofo toma un tenedor y el otro está ocupado, se quedará esperando, con el tenedor en la mano, hasta que pueda tomar el otro tenedor, para luego empezar a comer.

Una vez que termina de comer deja los tenedores sobre la mesa y piensa por un momento hasta que luego empieza a comer nuevamente.

Figura 1: Filósofos a la mesa

Se les provee un esqueleto de código que acompaña a éste ejercicio en la carpeta de *Ejemplos*.

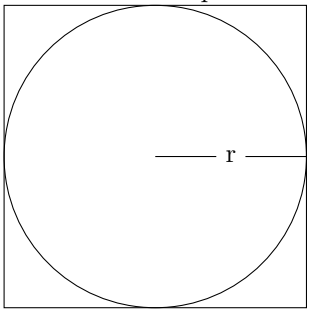
- Este programa puede terminar en **deadlock**. ¿En qué situación se puede dar?
- Cansados de no comer los filósofos deciden pensar una solución a su problema. Uno razona que esto no sucederá si alguno de ellos fuese zurdo y tome primero el tenedor de su izquierda.

  Implemente esta solución y explique por qué funciona.

Aproximación a π

La idea de éste ejercicio es producir un programa que aproxime el número π. Para esto se utilizará un método conocido como el Método de Monte-Carlo para la aproximación de π.

La idea es imaginarnos una circunferencia de radio r inscrita en un cuadrado de lado $2 * r$. Como podemos observar en la siguiente figura:



De esta manera el área de la circunferencia es $A_c = \pi * r^2$ mientras que la del cuadrado es $A_r = 4 * r^2$. Por lo que la razón de las áreas nos dan la siguiente relación $\frac{A_c}{A_r} = \frac{\pi * r^2}{4 * r^2} = \frac{\pi}{4}$ por lo que se concluye que $\pi = \frac{A_c * 4}{A_r}$.

Para aproximar el valor de π se generarán una cantidad arbitraria de puntos NPuntos al azar. Asumiendo que los puntos son generados aleatoriamente, estos corresponderán con la razón antes establecida. Es decir, que si contamos los puntos generados dentro de la circunferencia, circTotal, podemos aproximar a π de la siguiente forma: $\pi \approx \frac{4 * \text{circTotal}}{\text{NPuntos}}$.

Se les provee un esqueleto de código que acompaña a éste ejercicio en la carpeta de *Ejemplos*.

- Implementar de forma secuencial el algoritmo propuesto.
- Dar una implementación paralela donde se utilicen una cantidad de hilos NHilos con una carga balanceada entre ellos. Explique su solución.