Práctica 2.- Introducción a la programación usando POSIX Threads

Resumen

En esta práctica el alumno aprenderá a crear, compilar y ejecutar un programa que usa POSIX-Threads. La librería POSIX-Threads nos permite la ejecución concurrente de funciones de nuestro programa sin necesidad de hacer enormes esfuerzos por parte del programador para adaptar sus programas. Si nuestro ordenador posee más de una CPU o más de un CORE por CPU, la concurrencia de la que hablábamos anteriormente se convierte en paralelismo. Las funciones ya no compiten por usar la misma CPU, sino que cada función se ejecutaría en una CPU o CORE distinto. En el caso de que hagamos la práctica en un ordenador con una única CPU, no podemos hacer paralelismo pero sí concurrencia. No podremos bajar la velocidad de ejecución pero sí podremos preparar nuestros programas para que funcionen de manera óptima cuando los portemos a otros ordenadores que posean múltiples CPU's o cores.

Objetivos

- Conocer la librería POSIX-Threads
- Crear, compilar y usar programas basados en la librería POSIX-Threads

Entregables

El alumno elaborará un documento de texto o pdf que podrá subir usando el "Aula Virtual". Fuera de plazo no se aceptarán la entrega de prácticas. El valor de esta práctica es de 10 puntos, 2'5 puntos cada una de las 4 actividades. Todas las actividades y capturas de pantalla que se piden en las actividades deben estar incluidas y claramente identificadas en el documento. Como complemento a la memoria, es posible que el profesor de prácticas haga preguntas sobre la misma presencialmente.

Material necesario

- Acceso a un equipo con sistema Linux

Enlaces de interés

- https://computing.llnl.gov/tutorials/pthreads/
- http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html

Desarrollo

Los threads son procesos ligeros del sistema. Un estándar de implementación de threads es POSIX-Threads (pthreads). Este estándar es el que usan los sistemas operativos Linux y Unix. La desventaja principal de la programación usando POSIX-Threads, es que no pueden ejecutarse en máquinas remotas o máquinas de memoria distribuida. Un ejemplo sencillo de programación usando threads es el siguiente:

```c
#include <pthread.h>
#include <stdio.h>
#define NUM_THREADS 5
void *PrintHello(void *threadid){
	long tid;
	tid = (long)threadid;
	printf("Hello World! It's me, thread #%ld!\n", tid);
	pthread_exit(NULL);
}

int main (int argc, char *argv[]){
	pthread_t threads[NUM_THREADS];
	int rc;
	long t;
	for(t=0; t<NUM_THREADS; t++){
		printf("In main: creating thread %ld\n", t);
		rc = pthread_create(&threads[t], NULL, PrintHello, (void *)t);
		if (rc){
			printf("ERROR; return code from pthread_create() is %d\n", rc);
			exit(-1);
		}
	}
	pthread_exit(NULL);
}
```

Para añadir las funciones de la librería pthreads, necesitamos añadir a nuestro programa el fichero cabecera "pthread.h" ("#include<pthread.h>"), además de compilar nuestro programa fuente pasando el parámetro "-lpthread" al compilador. Para generar un fichero binario ejecutable a partir del programa fuente anterior, que incluya la librería pthread, invocamos la siguiente orden desde la línea de comandos:

>> gcc thread_1.c -o thread_1 -lpthread.

De esta manera generaremos un fichero ejecutable denominado "thread_1" a partir de un programa fuente denominado "thread_1.c" que incluye la librería pthread.

Dado el siguiente programa en C que usa la libreria Posix-Threads:

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

void *print_message_function( void *ptr );

main(){
	pthread_t thread1, thread2;
	char *message1 = "Thread 1";
	char *message2 = "Thread 2";
	int iret1, iret2;
	/* Create independent threads each of which will execute function */
	iret1 = pthread_create( &thread1, NULL, print_message_function, (void*)
	message1);
	iret2 = pthread_create( &thread2, NULL, print_message_function, (void*)
	message2);
```

```
        /* Wait till threads are complete before main continues. Unless we */
        /* wait we run the risk of executing an exit which will terminate */
        /* the process and all threads before the threads have completed. */
        pthread_join( thread1, NULL);
        pthread_join( thread2, NULL);
        printf("Thread 1 returns: %d\n",iret1);
        printf("Thread 2 returns: %d\n",iret2);
        exit(0);
}

void *print_message_function( void *ptr ){
        char *message;
        message = (char *) ptr;
        printf("%s \n", message);
}
```

<u>Actividad 1</u>: Identificar las funciones Posix-Threads y explicar brevemente que realiza cada una y que parámetros y de qué tipo aceptan. ¿Qué realiza el programa?

Identificar la funcion "hebrada". La función hebrada es la que se ejecuta concurrentemente. Para poder compilar el programa necesitamos añadir la librería de threads, esto se hace añadiendo el parametro "-lpthread" a la orden de compilación.

<u>Actividad 2:</u> Modificar el programa para que acepte por línea de comandos el número de threads ("hilos") que se quieren ejecutar.

<u>Actividad 3</u>: Construir una función que realice la suma de un vector, de tipo *double y* de una longitud determinada pasada por parámetro.

<u>Actividad 4</u>: Construir un programa que incorpore la función de la actividad anterior para que ejecute de manera concurrente 4 threads de dicha función y realizando cada uno de los threads la suma parcial de 1/4 de la longitud total del vector. Al final el programa tendrá que obtener la suma total de los elementos del vector.