UNIVERSIDAD EMPRESARIAL

SIGLO 21

CARRERA: LIC. EN INFORMATICA

Trabajo Práctico número 2

Materia: **Algoritmos concurrentes y paralelos**

Profesor: PIRAY, EDUARDO ENRIQUE

Modulo: 2

Alumno: Soler, Diego Francisco.

Fecha: 17/06/2024

Objetivo: El objetivo principal de esta actividad consiste en que el estudiante identifique los posibles usos de los *threads* para el paralelismo y la concurrencia, y las herramientas que brindan soporte a la sincronización y comunicación.

Consignas: a. Implementa un algoritmo en C que reciba como argumento dos números enteros, N y M. Tales números serán utilizados para crear N hilos (*threads*) de tipo 1 y M hilos (*threads*) de tipo 2. A continuación se describe la funcionalidad que deberás implementar cada uno de los hilos:

Hilo de tipo 1: Deberás mostrar por pantalla el mensaje: “Instancia x del hilo 1”, siendo X el número de hilo creado (entre 0 y N-1). Posteriormente, se suspenderá por un tiempo aleatorio entre 0 y 2 segundos (para ello utilizarás la llamada *usleep* o *sleep*). Finalmente, incrementará la variable global “Compartida”.

Hilo de tipo 2: Deberás mostrar por pantalla el siguiente mensaje: “Instancia y del hilo 2”, siendo Y el número de hilo creado (entre 0 y M-1). Posteriormente, se suspenderá por un tiempo aleatorio entre 0 y 2 segundos. Para finalizar, leerá y mostrará por pantalla el valor de la variable global “Compartida”.

Cuando los hilos finalicen su ejecución, el padre debe mostrar por pantalla “Se ha finalizado la ejecución”. Ver funcionamiento de pthread_create y pthread_join.

b. Continúa con lo realizado en la consigna anterior, pero esta vez utilizando procesos en vez de hilos. Para ello, deberás crear, por medio de la llamada *fork*, N procesos tipo 1 y M procesos de tipo 2.

Realiza un análisis del comportamiento de ambas implementaciones y detalla el comportamiento observado, diferencias, describir.

Implementación con hilos:

Los hilos a diferencia de los procesos comparten la misma memoria, lo que permite que todos lo hilos pueden acceder a la variable “compartida” y puedan modificar la misma. La sincronización la logramos utilizando una función atómica como lo es “__sync_fetch_and_add” que nos permite garantizar una operación compartida de manera segura y sin conflictos.

Implementación con procesos:

Dado a que los procesos no comparten la misma memoria directamente, utilizamos el mecanismo de la memoria compartida para que se comuniquen entre ellos, de esta forma, y con la utilización de la función atómica “__sync_fetch_and_add” los procesos de tipo1 y de tipo 2 pueden acceder a la variable “compartida” de manera sincronizada y segura.

Diferencias:

La diferencia entre las 2 implementaciones yace en que que los hilos comparten memoria directamente a diferencia de los procesos, por lo cual, cuando utilizamos la implementación con procesos nos vemos obligados a implementar la memoria compartida lo que agrega complejidad al código.

En conclusión, la implementación con hilos es más eficiente ya que, para modificar una variable compartida no requieren de una comunicación externa. Pero la implementación con procesos es mas segura, dado que no comparten la misma memoria y solo se comunica a través de la memoria compartida.