MicroProcesadores

Nicolás Concua - 23197
Esteban Cárcamo - 23016

Laboratorio 6

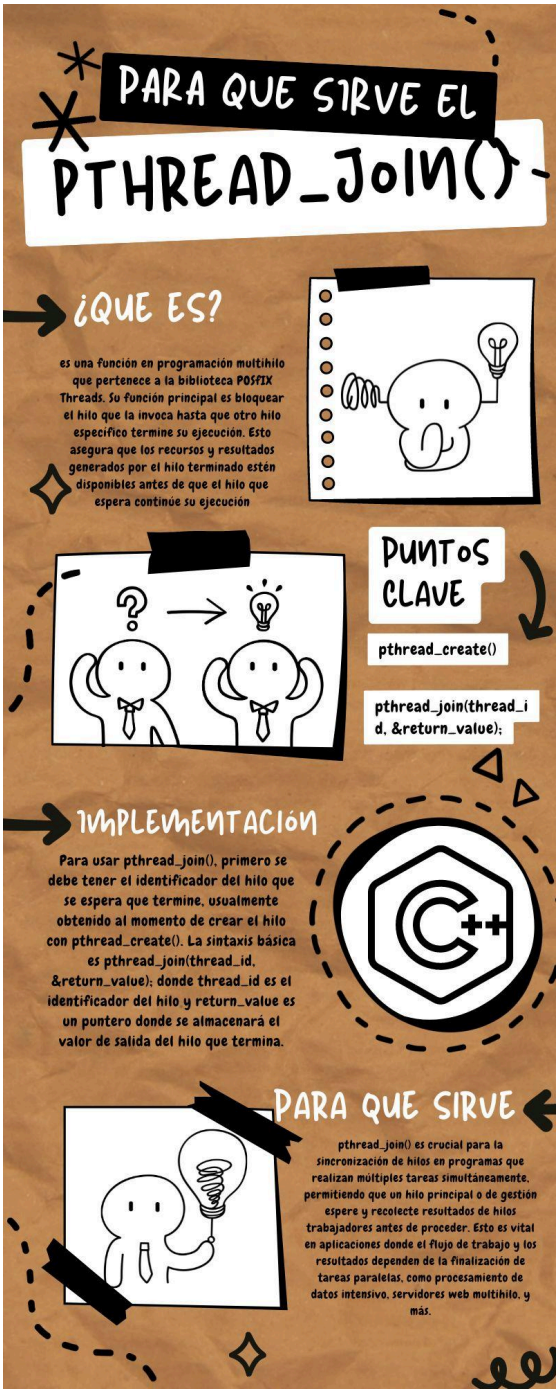
17 de septiembre del 2024

Link del repo: https://github.com/CarEsteban/microprocesadores.git

Ejercicio 2-Investigación

---

a. Investigue qué es, cómo usar y para qué sirve pthread_join(). Deje evidencia de su investigación creando una infografía.



Utilice el ejemplo de hello world con Pthreads y:

b. Modifique el programa para realizar un ciclo for para crear hilos y un ciclo for (separado) para hacer su respectivo join. Cada hilo debe de imprimir “Hello world thread No. X” colocando el número del hilo en lugar de la “X”.

helloworld.cpp

```cpp
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define NUM_THREADS 10

void *PrintHello(void *paramID) {
    int *id = (int *)paramID;  // Conversión explícita de void* a int*
    printf("hello world thread no. #%d\n" ,  *id);
    pthread_exit(NULL);
    return NULL;
}

int main(int argc, char *argv[]) {
    pthread_t threadsID[NUM_THREADS];
    pthread_attr_t attr;


    //Inicializador
    pthread_attr_init(&attr);

    // Modificar los atributos del thread
    pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_JOINABLE);


    int rc, t, param[NUM_THREADS];

    for (t = 0; t < NUM_THREADS; t++) {
        param[t] = t;
        rc = pthread_create(&threadsID[t], &attr, PrintHello, (void *)&param[t]);

        if (rc) {
            printf("ERROR: código retornado desde pthread_create() es %d\n", rc);
            exit(-1);
        }
    }

    //Join de hilos
    for (t = 0; t < NUM_THREADS; t++) {
        pthread_join(threadsID[t], NULL);
    }

    pthread_attr_destroy(&attr);
    pthread_exit(NULL);
}

```

```
nico@DESKTOP-0FC02F8:~/proyectosmp$ ./helloworld
hello world thread no. #0
hello world thread no. #1
hello world thread no. #2
hello world thread no. #3
hello world thread no. #4
hello world thread no. #5
hello world thread no. #6
hello world thread no. #7
hello world thread no. #8
hello world thread no. #9
```

c. Basándose en el mismo ejemplo, realice un ciclo for que cree el hilo y haga su respectivo join dentro del mismo ciclo. Cada hilo debe de imprimir “Hello world thread No. X” colocando el número del hilo en lugar de la “X”.

```c
#define NUM_THREADS 10

void *PrintHello(void *paramID) {
    int *id = (int *)paramID;  // Conversión explícita de void* a int*
    printf("hello world thread no. #%d\n" ,  *id);
    pthread_exit(NULL);
    return NULL;
}

int main(int argc, char *argv[]) {
    pthread_t threadsID;
    pthread_attr_t attr;


    //Inicializador
    pthread_attr_init(&attr);

    // Modificar los atributos del thread
    pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_JOINABLE);


    int rc, t, param;

    for (t = 0; t < NUM_THREADS; t++) {
        param = t;
        rc = pthread_create(&threadsID, &attr, PrintHello, (void *)&param);

        if (rc) {
            printf("ERROR: código retornado desde pthread_create() es %d\n", rc);
            exit(-1);
        }

        pthread_join(threadsID, NULL);
    }


    pthread_attr_destroy(&attr);
    pthread_exit(NULL);
```

```
nico@DESKTOP-0FC02F8:~/proyectosmp$ ./helloworld
hello world thread no. #0
hello world thread no. #1
hello world thread no. #2
hello world thread no. #3
hello world thread no. #4
hello world thread no. #5
hello world thread no. #6
hello world thread no. #7
hello world thread no. #8
hello world thread no. #9
```

Responder las siguientes preguntas:

1. **¿Cuál fue la diferencia entre la impresión del primer programa y el segundo?**
    a. La principal diferencia entre la impresión de los resultados del primer programa y del segundo radica en el orden de los mensajes de salida de los hilos:

b. Primer programa: Los hilos fueron lanzados en un ciclo for y luego se les hizo join en un ciclo for separado. Esto permitió que los hilos se ejecutaran en paralelo, resultando en una salida desordenada porque diferentes hilos podrían completarse en momentos distintos dependiendo del planificador de hilos del sistema.
c. Segundo programa: Cada hilo fue creado y se le hizo join en la misma iteración del ciclo for, lo que significó que cada hilo tuvo que completarse antes de iniciar el siguiente. Esto aseguró que los hilos se ejecutaran y terminaran en un orden secuencial, resultando en una salida ordenada.

2. ¿A qué se debió el comportamiento descrito en la respuesta anterior?

a. El comportamiento observado se debió a la forma en que se gestionaron los hilos en cada programa. En el primer programa, la creación de los hilos en un ciclo y su espera en otro permitió que se ejecutaran simultáneamente, resultando en una salida desordenada debido a la independencia en la terminación de los hilos. En cambio, en el segundo programa, al crear y esperar cada hilo dentro del mismo ciclo, se aseguró una ejecución secuencial, manteniendo el orden de salida tal como fueron iniciados los hilos.

   Si el orden de salida y la secuencialidad son importantes, como en procesos que dependen de la salida de un hilo anterior para continuar, el segundo enfoque es claramente superior. Asegura que cada hilo se complete antes de que el siguiente comience, manteniendo un flujo ordenado y predecible.

   Si el rendimiento y la eficiencia son prioritarios, especialmente en sistemas con múltiples núcleos donde la ejecución paralela puede aprovecharse al máximo, el primer enfoque es mejor. Permite que varios hilos se ejecuten simultáneamente, reduciendo potencialmente el tiempo total de ejecución del programa al aprovechar la concurrencia.

   Las 2 son muy eficientes dependiendo de la finalidad de ellas.

Ejercicio 3-Preguntas

Explique, ¿cómo decidió en qué punto del programa colocar la sumatoria para que la operación se realizará correctamente y que se aprovechara el paralelismo?

Durante la elaboración del programa, la mejor elección fue declarar una variable global, en la cual se fueron sumando los resultados después de haber calculado el resultado con cada hilo, se observó que se puede devolver el resultado y cuando se unen lo hilos obtener el valor para agregarlo a la sumatoria, entonces cuando los hijos están en el ciclo del "joinable" mediante el segundo parámetro se obtiene el resultado de cada hilo para sumarlo a la variable global, y así evitar que hayan colisiones en la misma variable.

De esta forma seguimos aprovechando el paralelismo cuando se obtiene el valor para cada resultado y se suma de una forma global para evitar alguna sobreescritura.