SISTEMAS OPERATIVOS

TALLER No. 5: Hilos

OBJETIVOS

1. Familiarizar al estudiante con el concepto de hilo de ejecución.
2. Comprender cómo se crean y se lanzan las ejecuciones de varios hilos a la vez.
3. Comprender las motivaciones para escribir programas multihilo.
4. Convertir un programa de un solo hilo en un programa multihilo.
5. Comprender los problemas asociados al diseño y la ejecución de programas multihilo.

CONTENIDO DEL TALLER

1. Creación de hilos con C en modelo sincrónico**.**
2. Creación de hilos en C modelo asincrónico.
3. Hilos en Python.
4. Motivación para usar hilos: medida de tiempo de ejecución sin usar hilos.
5. Motivación para usar hilos: medida de tiempo de ejecución usando hilos.
6. Problemas de usar hilos: sincronización.

DESARROLLO DEL TALLER

1. Creación de hilos con C en modelo sincrónico.

   Escriba, compile y ejecute el siguiente programa en Linux. Tenga en cuenta que para compilar este programa debe hacerse como se indica a continuación.

```
#  gcc -pthread -o <archivo_ejecutable> <archivo_fuente>.c
```

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

void * funcion(void *arg) {
        int id = (int) pthread_self();
        printf("Hilo %d \n", id);
        pthread_exit(0);
}

int main(int argc, char *argv[]) {
        pthread_t h1, h2;

        pthread_create(&h1, NULL, funcion, NULL);
        pthread_create(&h2, NULL, funcion, NULL);

        printf("main() sigue su ejecución\n");
        printf("main() sigue su ejecución\n");
        printf("main() sigue su ejecución\n");

        pthread_join(h1, NULL);
        pthread_join(h2, NULL);

        printf("main() termina\n");
        return 0;
}
```

1.1 ¿Qué sucede si comenta las instrucciones de las líneas 22 y 23?

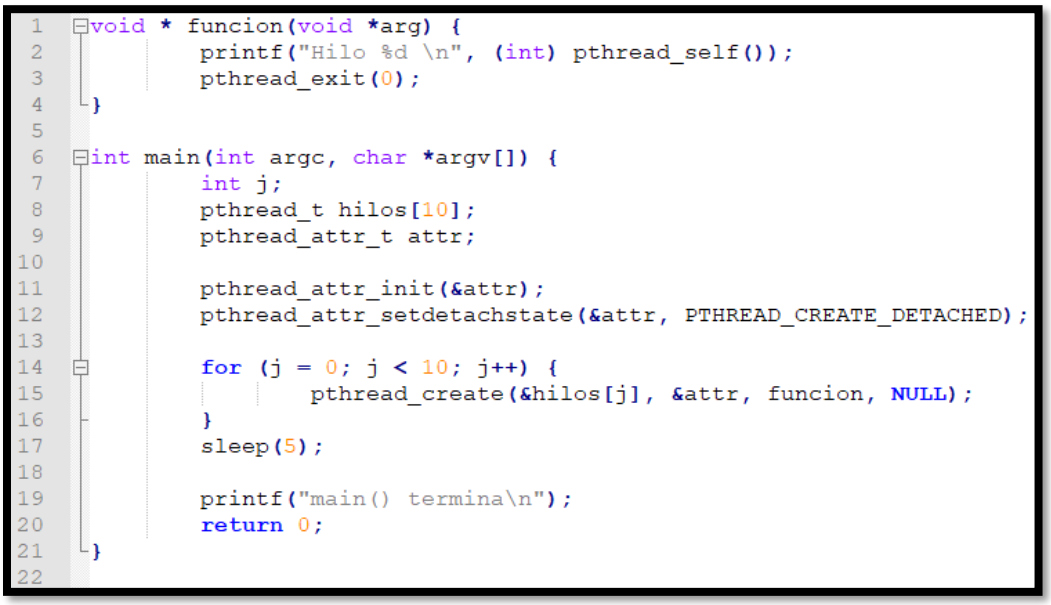
1.2 Modifique el programa anterior para que ocurra un retardo de un segundo en el hilo. Use la función `sleep(1);`.

1.3 Modifique el programa anterior para que imprima una cadena de texto cuando se crea el hilo y verifique en repetidas ocasiones cuál hilo se ejecuta primero.

1.4 Verifique si puede modificar el número de CPUs asignadas a la máquina virtual de Virtual Box e incremente en uno el número de CPUs. Ejecute de nuevo el programa en repetidas ocasiones y verifique si hay cambios en el resultado y orden de ejecución.

2. Creación de hilos con C en modelo asincrónico.

   Escriba, compile y ejecute el siguiente programa en Linux.



```c
void * funcion(void *arg) {
        printf("Hilo %d \n", (int) pthread_self());
        pthread_exit(0);
}

int main(int argc, char *argv[]) {
        int j;
        pthread_t hilos[10];
        pthread_attr_t attr;

        pthread_attr_init(&attr);
        pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_DETACHED);

        for (j = 0; j < 10; j++) {
            pthread_create(&hilos[j], &attr, funcion, NULL);
        }
        sleep(5);

        printf("main() termina\n");
        return 0;
}
```

2.1 ¿Qué sucede si disminuye a un segundo el tiempo de espera del hilo principal?

2.2 ¿Qué sucede si elimina el tiempo de espera del hilo principal?

2.3 Modifique el programa anterior para que imprima una cadena de texto cuando se crea el hilo y verifique en repetidas ocasiones cuál hilo se ejecuta primero.

2.4 Agregue un ciclo con retardo de un 1 segundo para cada hilo. Agregue nuevas instrucciones al hilo principal y verifique el orden de ejecución de su programa.

3. Hilos en Python

Con el objetivo de facilitar la comprensión del comportamiento de los hilos, a continuación, se explica muy brevemente el manejo de hilos en Python.

Para crear hilos en Python se debe importar la siguiente clase:

```
from threading import Thread
```

Para crear un hilo, se debe instanciar la clase `Thread` que tiene el siguiente constructor.

```
hilo = Thread(target=fn, args=tupla_args)
```

Donde,

- `hilo` corresponde al objeto/instancia que representa un hilo.
- `fn` corresponde a la función que define el flujo de trabajo del hilo.
- `tupla_args` corresponde a una tupla de parámetros que se pasan al hilo en caso de que se requiera pasar algún parámetro al hilo.

Lo anterior solo define un objeto de tipo hilo, sin embargo, para iniciar la ejecución del hilo, hay que usar el método `start()` del objeto creado.

```
hilo.start()
```

De manera análoga a como sucede en el programa del numeral uno de este documento, desde el hilo principal se puede invocar al método `join()` del hilo creado, para esperar a que termine su ejecución.

```
hilo.join()
```

El siguiente programa en Python es equivalente al programa del numeral uno de este documento. Use Google Colab para ejecutar el siguiente programa de Python y verifique los comportamientos en la ejecución.

```
1  from threading import Thread
2  from threading import get_ident
3
4  def funcion():
5      print("Soy un hilo ident = ", get_ident())
6      return
7
8  t1 = Thread(target=funcion)
9  t2 = Thread(target=funcion)
10 t1.start()
11 t2.start()
12 print("Hilo principal sigue su ejecución")
13 print("Hilo principal sigue su ejecución")
14 print("Hilo principal sigue su ejecución")
15 t1.join()
16 t2.join()
17 print("Termina principal")
```

3.1 Ejecute varias veces el programa anterior.

3.2 Agregue un retardo del tiempo que usted quiera en el hilo. Use la función sleep de la librería `time`: `from time import sleep`

3.3 Cambie de posición (en otro orden después de iniciar los hilos) o quite las instrucciones de las líneas 15 y 16 añadiendo retardos de ejecución a los hilos y corrobore el resultado.

4. Motivación para usar hilos: medida de tiempo de ejecución sin usar hilos.

   Escriba y ejecute el siguiente programa en Python (use Google Colab)

```
1  from time import sleep
2  from time import perf_counter
3
4  # Definición de la función
5  def mifuncion():
6      print("Estoy en mi función")
7      # Retardo de un segundo
8      sleep(1)
9      print("Terminé función")
10
11 # Se inicia temporizador
12 t1 = perf_counter()
13
14 # Se llama dos veces a la función
15 mifuncion()
16 mifuncion()
17
18 # Se termina temporizador
19 t2 = perf_counter()
20
21 # Se muestra el tiempo total de ejecución
22 print("Tiempo:",round((t2-t1),2),"segundos")
```

   4.1 ¿ De cuánto es el tiempo de ejecución del programa anterior?

5. Motivación para usar hilos: medida de tiempo de ejecución usando hilos.

   Basándose en el diseño del programa multihilo explicado en el numeral tres de este documento, convierta el programa anterior (del numeral cuatro) en un programa multihilo de Python. Los dos llamados a la función que se realizan en las líneas 15 y 16 se reemplazan por el inicio de la ejecución de cada hilo. Mida el tiempo de ejecución y verifique si hay alguna diferencia significativa con respecto al tiempo de ejecución obtenido en el programa del del numeral cuatro.

6. Problemas de usar hilos: sincronización.

   En este ejercicio se propone construir un programa multihilo en Python para calcular una función sumatoria desde un límite inferior (p.ej.: 1) hasta un límite superior (p. ej.: 100), teniendo en cuenta que, esta sumatoria la deben hacer dos hilos: un hilo debe calcular la sumatoria desde el límite inferior hasta la mitad del intervalo y el otro hilo debe calcular la sumatoria para la mitad del intervalo restante.

   6.1 Se debe escribir **una sola función** para el indicar el flujo de ejecución del hilo. Se crean dos hilos llamando a esta sola función.

6.2 La función de ejecución del hilo debe recibir dos parámetros: límite inferior del intervalo y límite superior del intervalo.

6.3 La variable que almacena la sumatoria debe ser global para todo el proceso. Por ejemplo, si su variable se llama **s**, dicha variable debe ser inicializada en cero en el hilo principal y declarada como global dentro de la función de ejecución del hilo: por ejemplo, `global s` iría al principio de la función.

6.4 Para pasar parámetros a la función de ejecución del hilo, debe hacerlo cuando crea la instancia del objeto hilo como se indica a continuación.

```
hilo = Thread(target=sumatoria, args=(li, ls))
```

Donde `li` y `ls` son los parámetros que está pasando a la función sumatoria (función de ejecución del hilo).

6.5 Muestre el resultado de la sumatoria al finalizar los hilos creados y disponga al hilo principal para esperar por la finalización de los dos hilos creados.

6.6 Ejecute varias veces su programa y verifique si obtiene el mismo resultado siempre.

6.7 Añada como última instrucción del hilo, un retardo de un segundo utilizando la función `sleep(1)`. Ejecute varias veces su programa y verifique resultados.

6.8 Elimine el retardo de un segundo e incremente los intervalos. Por ejemplo, el primer hilo que calcule la sumatoria desde 1 hasta 1000000 y el segundo hilo calcula la sumatoria desde 1000001 hasta 2000000. Ejecute el programa varias veces y verifique resultados. ¿Obtiene los mismos resultados? ¿Por qué obtiene resultados distintos? ¿Qué se le ocurre para resolver el problema en caso de que no siempre obtenga los mismos resultados?

6.9 Intente el último numeral escribiendo un programa convencional de un solo hilo principal: ¿tarda lo mismo que el multihilo? ¿siempre da el mismo resultado?