
# Instituto Politécnico Nacional
# Escuela Superior de Cómputo
# "ESCOM"


## Unidad de Aprendizaje:
## Cómputo Paralelo

## Práctica No. 6
## Programación de hilos en Python

**Integrantes:**
**Sánchez De Los Ríos Flavio Josué - 6BV1**
**Tinoco Videgaray Sergio Ernesto - 6BV1**
**Ibarra González Emilio Francisco- 6BV1**

**Maestro:**
**Jiménez Benítez José Alfredo**

**Fecha de Entrega: 16/06/23**

## Resumen

Dentro de los sistemas de cómputo de alto rendimiento se utilizan múltiples recursos y herramientas que facilitan la realización de tareas de forma simultánea, como ejemplos existen la memoria compartida que nos permite acceder a un mismo sector dentro de la memoria principal con el fin de leer o escribir un dato que pueda ser accedido por uno o más procesos dentro de la misma región de memoria, los semáforos que nos permite mantener la sincronicidad entre los distintos procesos que se ejecutan de tal forma que no se presente algún inconveniente a la hora de acceder al mismo elemento de la memoria compartida. Y uno de los recursos más utilizados actualmente y que se van a abordar en el desarrollo de esta práctica son los hilos o también llamados "threads", los cuales se definen como una secuencia de instrucciones o conjunto de instrucciones que puede ser ejecutada de forma independiente dentro de un proceso. Muchos sistemas operativos hacen uso de estos hilos, en algunos casos llamados como "subprocesos" cumplen la función de realizar múltiples operaciones "simples" para de esta forma dividir la carga del trabajo de cada proceso.

Actualmente hay varias formas de implementar estos hilos en los diferentes lenguajes de programación, algunas de ellas son muy similares ya que implementan paquetes o bibliotecas que permiten el uso de estos hilos dentro del sistema. En el caso del lenguaje de programación Python, el uso de hilos se realiza por medio del módulo "Threading" que integra una clase llamada "Thread" que gestiona el comportamiento de un subproceso que se ejecuta de forma asíncrona dentro de un proceso.

En el desarrollo de esta práctica se va a codificar un programa que, haciendo uso de hilos, genere una matriz de números y realice la suma de estos por fila para mostrar los resultados posteriormente.

# Índice

## CONTENIDO

Introducción ..... 1
Marco teórico ..... 2
Desarrollo ..... 5
Conclusiones ..... 7
Referencias ..... 11
Anexo. Códigos completos. ..... 12

# Introducción

Uno de los recursos más utilizados en el cómputo paralelo son los hilos o también llamados “threads”, los cuales se definen como una secuencia de instrucciones o conjunto de instrucciones que puede ser ejecutada de forma independiente dentro de un proceso. Muchos sistemas operativos hacen uso de estos hilos, en algunos casos llamados como “subprocesos” cumplen la función de realizar múltiples operaciones “simples” para de esta forma dividir la carga del trabajo de cada proceso.

Estos hilos son elementos fundamentales en el ámbito de la programación concurrente y paralela. Cada hilo puede entenderse como una secuencia de instrucciones o un flujo de ejecución que es independiente de un proceso. A medida que los sistemas informáticos han evolucionado y se han vuelto cada vez más complejos, la capacidad de ejecutar múltiples tareas simultáneamente se ha vuelto cada vez más importante.

Python es un lenguaje de programación de código abierto, creado en 1991 por Guido van Rossum. Se trata de un lenguaje bajo el paradigma orientado a objetos, aunque también se puede aplicar para otros paradigmas. Es fácil de interpretar y con una sintaxis que permite leerlo de manera semejante a como se lee el inglés. Es un lenguaje interpretado, esto significa que el código de programación se convierte en bytecode y luego se ejecuta por el intérprete, que, en este caso, es la máquina virtual de Python.

La implementación de hilos en el lenguaje de programación Python se realiza mediante el uso de bibliotecas o “paquetes” como el caso del paquete “Threading” que utiliza una clase llamada Thread la cual funciona como superclase para cada instancia de dicha clase en cualquier metodo que se quiera utilizar.

# Marco teórico

## Tipos de datos en Python

Existen diferentes tipos de datos en Python, ya que Python es utilizado en distintas áreas como la estadística, la ingeniería, el aprendizaje de máquina, el desarrollo de aplicaciones WEB, etc [1].

| | |
|---|---|
| Numéricos | Son tipos aritméticos y consisten de dos tipos: enteros, de punto flotante y los complejos. |
| Cadenas | Se caracterizan por ser una secuencia de caracteres. |
| Booleanos | Los valores booleanos en Python incluyen dos valores posibles: “True” y “False” |
| Objetos | Incluyen tipos compuestos por otros como listas, tuplas, diccionarios o cualquier tipo de dato definido por medio de una clase. |

**Tabla 1.** Clasificación de tipos de datos en Python [1].

## Procesos

En el cómputo paralelo, un proceso es una unidad de ejecución que realiza una tarea específica. Los procesos se relacionan con el cómputo paralelo porque se utilizan para dividir una tarea compleja en subtareas más pequeñas y ejecutarlas en paralelo en múltiples núcleos de procesamiento [2]. De esta manera, se puede lograr un aumento significativo en la velocidad de procesamiento y reducir el tiempo de respuesta.

El principal concepto en cualquier sistema operativo es el de proceso, un proceso es un programa en ejecución, incluyendo el valor de los registros y las variables [2]. Cuando se inicia un proceso en un sistema, el proceso utiliza una parte de los recursos disponibles en el sistema. Cuando está ejecutándose más de un proceso, un planificador que está incorporado al sistema operativo proporciona a cada proceso su parte de tiempo del sistema, basándose en las prioridades establecidas.

En la siguiente tabla se describen los tipos de procesos [2].

| | |
|---|---|
| **Procesos en primer plano y en segundo plano** | Son procesos que necesitan que un usuario los inicie o que interactúe con ellos se denominan procesos en primer plano. Los procesos que se ejecutan con independencia de un usuario se denominan procesos en segundo plano. |

| | |
|---|---|
| **Procesos daemon** | Son procesos que se ejecutan de forma desatendida. Están constantemente en segundo plano y siempre están disponibles. Los daemons suelen iniciarse cuando se arranca el sistema y se ejecutan hasta que se detiene el sistema. |
| **Procesos zombie** | Es un proceso finalizado que ya no se ejecuta pero que sigue reconociéndose en la tabla de procesos (en otras palabras, tiene un número PID). Ya no se asigna espacio del sistema a dicho proceso. |

**Tabla 2.** Tipos de procesos en un sistema operativo [2].

## Hilos

Los hilos de proceso, también conocidos como hilos o "threads", son elementos fundamentales en el ámbito de la programación concurrente y paralela. Un hilo puede entenderse como una secuencia de instrucciones o un flujo de ejecución independiente dentro de un proceso [3].

Un proceso puede contener uno o varios hilos, y cada uno de ellos representa una unidad de trabajo que puede realizar tareas en paralelo con otros hilos del mismo proceso. Aunque los hilos comparten los recursos del proceso, como la memoria y los archivos abiertos, cada hilo tiene su propio contador de programa, pila de ejecución y estado de registro. Esto les proporciona una ejecución independiente y les permite comunicarse y cooperar entre sí de manera eficiente [3].

Una de las ventajas de utilizar hilos de proceso es la capacidad de realizar múltiples tareas simultáneamente dentro de una aplicación. Por ejemplo, en una aplicación de procesamiento de imágenes, se puede utilizar un hilo para cargar imágenes desde el disco, otro hilo para procesar las imágenes y un tercer hilo para mostrar los resultados en la interfaz de usuario. Esto permite que la aplicación sea más receptiva y evita que una tarea prolongada bloquee por completo la ejecución de otras tareas [3].

Además, los hilos son especialmente útiles en sistemas con múltiples núcleos de procesamiento o también denominados "multicore". Cada núcleo puede ejecutar un hilo diferente, lo que permite una ejecución paralela real y un mejor aprovechamiento de los recursos del hardware. Esto puede resultar en un aumento significativo en el rendimiento de las aplicaciones que hacen un uso intensivo de la CPU [3].

Sin embargo, trabajar con hilos también presenta algunos desafíos. Los hilos comparten memoria y otros recursos, lo que puede llevar a condiciones de carrera y problemas de sincronización. Una condición de carrera ocurre cuando varios hilos intentan acceder y modificar los mismos datos simultáneamente, lo que puede llevar a resultados inesperados o incorrectos. Por lo tanto, es necesario utilizar mecanismos de sincronización, como semáforos, para garantizar que los hilos accedan a los recursos compartidos de manera segura y ordenada [4].

Los principales estados de los hilos son: Ejecución, Listo y Bloqueado. Si un proceso es expulsado de la memoria principal, todos sus hilos deberán estarlo ya que todos comparten el espacio de direcciones del proceso [4].

- *Creación:* Cuando se crea un proceso se crea un hilo para ese proceso. Luego, este hilo puede crear otros hilos dentro del mismo proceso, proporcionando un puntero de instrucción y los argumentos del nuevo hilo. El hilo tendrá su propio contexto y su propio espacio de la columna, y pasará al final de los Listos.
- *Bloqueo:* Cuando un hilo necesita esperar por un suceso, se bloquea (salvando sus registros de usuario, contador de programa y punteros de pila). Ahora el procesador podrá pasar a ejecutar otro hilo que esté al principio de los Listos mientras el anterior permanece bloqueado.
- *Desbloqueo:* Cuando el suceso por el que el hilo se bloquea se produce, el mismo pasa a la final de los Listos.
- *Terminación:* Cuando un hilo finaliza se liberan tanto su contexto como sus columnas.

## Desarrollo

Primeramente, se importan los paquetes necesarios para esta práctica figura 1.

```
import threading
import random
import numpy as np
```

*Figura 1. Vista de la carpeta Practica 5.*

Se define la función del hilo 1 que se va a encargar de crear la matriz junto con los numeros que va a servir como consulta para los hilos 2-4 (figura 2).

```
def crearMatriz(mat):
    print(f'{threading.current_thread().name} te saluda')
    mat[0,:]=np.arange(1,10)
    mat[1,:]=np.arange(1,18,2)
    mat[2,:]=np.arange(2,19,2)
    print(f'{threading.current_thread().name} se despide')
```

*Figura 2. Función que crea la matriz con los números.*

Se define la función que va a ejecutar cada hilo (hilo 2 al hilo 4) (figura 3).

Dentro de la función se recibe como parámetro la matriz de números, el número de la fila que le corresponde a cada hilo y una lista a manera de arreglo para almacenar la suma en el espacio qué le corresponde al hilo.

Dentro de la función se va a iterar por medio de un for desde el 0 al 8 ya que la matriz esta compuesta por 9 columnas e internamente se va a realizar la suma acumulada de cada elemento de la matriz en la fila correspondiente (figura 3).

```
def ejecutar(matriz,fila,suma):
    print(f'{threading.current_thread().name} te saluda')
    for i in range (9):
        suma[fila]+=matriz[fila,i]

    print(f'{threading.current_thread().name} se despide')
```

*Figura 3. Función de cada hilo para realizar la suma por fila.*

Posteriormente se definen los arreglos para almacenar los valores a manera de listas.

Primero se define la matriz donde se van a almacenar los números a sumar por medio de la función “zeros” del paquete de Numpy que nos va a permitir generar un “iterable” es decir una matriz llena de ceros.

De igual forma se define una lista para los resultados de cada suma y una lista para los hilos utilizando el operador asterisco (figura 4).

```
mat=np.zeros((3,9))
suma=[0]*3
hilos=[None]*4
```

*Figura 4. Creación de las listas y la matriz para almacenar los valores numéricos.*

Posteriormente se define el primer hilo y se ejecuta por medio del método Thread que va a instanciar un objeto de la clase threading y va a asociarle una función a dicho hilo, en este caso le asigna la función crearMatriz definida previamente pasando como argumento la matriz de números previamente creada. Posteriormente se ejecuta el hilo y se le indica al proceso principal que espere a que el hilo termine de ejecutarse por medio del método “join” (figura 5).

```
#Creacion hilo 1
hilos[0] = threading.Thread(target=crearMatriz, name='Hilo 1',args=(mat,))
hilos[0].start()
hilos[0].join()
```

*Figura 5. Creación y ejecución del hilo 1.*

De igual forma de definen y ejecutan los hilos 2 al 4 por medio de un bucle for que va a repetir el mismo proceso de creación de cada hilo asociando cada hilo con la función ejecutar y pasando como argumento la matriz de números, el número de la fila que le corresponde a cada hilo y el arreglo de respuestas para almacenar las sumas (figura 6).

```
#Creacion hilos 2 al 4
for i in range (1,4):
  hilos[i] = threading.Thread(target=ejecutar, name=f'Hilo {i+1}',args=(mat,i-2,suma))
  hilos[i].start()
  hilos[i].join()
```

*Figura 6. Creación y ejecución de los hilos 2 al 4.*

Finalmente se imprimen los valores de la matriz de números y la lista de resultados (figura 7).

```
print("Matriz de numeros: \n", mat)
print("Suma de las filas:",suma)
```

*Figura 7. Impresión de la matriz y lista de reusltados.*

## Resultados.

Se ejecuta el programa y se muestran los resultados como se aprecian en la figura 8.

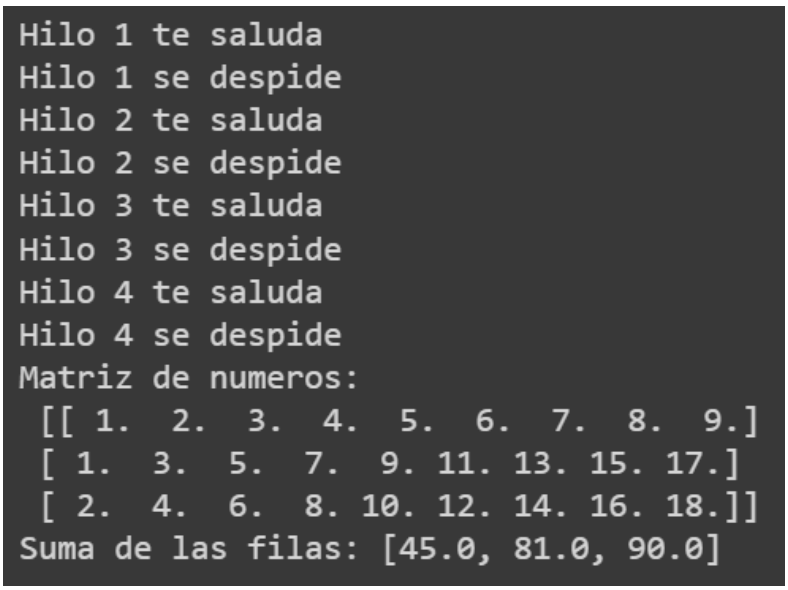


*Figura 8. Resultados de la ejecución del programa.*

# Conclusiones

**Sánchez De Los Ríos Flavio Josué**

Se abordó el tema del cómputo paralelo y la importancia de los hilos para calcular y acceder a la información en cada proceso. Realmente la implementación de hilos en Python es bastante más sencilla en comparación con otros lenguajes como C, principalmente con el manejo de los arreglos o en este caso de las listas.

Los hilos son una herramienta crucial en la programación concurrente que permiten dividir las tareas de un proceso. En el lenguaje de programación Python se implementan a través de paquetes como Threading y permiten a los hilos crearse y ejecutarse, así como permitir al proceso principal esperar a que la ejecución de dichos hilos termine similar a la implementación en C.

**Tinoco Videgaray Sergio Ernesto**

Los hilos resultan ser un recurso fundamental en el cómputo de alto rendimiento ya que permiten dividir la carga de trabajo a un nivel más profundo en el que ya no se hablan de procesos “padres” o procesos “no emparentados” sino de únicamente un solo proceso realizando múltiples operaciones de forma simultánea o “cuasi paralela”.

Como se pudo observar en esta práctica, el uso de hilos resulta clave en la paralelización de tareas ya que nos permite dividir el conjunto de operaciones aún más que si se utilizaran únicamente procesos no emparentados que necesiten comunicarse entre sí, no obstante el uso de hilos también tiene sus desventajas como el hecho de que los hilos dependan del proceso al que pertenecen y es que en el dado caso de que dicho proceso terminará su ejecución antes que los hilos dentro de él, por lo cual es necesario implementar el método join para que el proceso espere a la ejecución de cada hilo.

En Python los hilos se implementan utilizando el paquete de Threading y resulta bastante sencillo gracias a que Python es un lenguaje de alto nivel y multiparadigma como el paradigma orientado a objetos, con el cual se puede hacer una instancia de la clase Thread y gestionar sus atributos y métodos sin necesidad de hacer uso de un TDA como el caso de C.

Ibarra González Emilio Francisco

Los hilos de proceso son unidades de ejecución dentro de un proceso que permiten la ejecución simultánea de múltiples tareas. Su uso adecuado puede mejorar el rendimiento y la capacidad de respuesta de las aplicaciones, pero también requiere un manejo cuidadoso para evitar problemas de concurrencia. A medida que los sistemas informáticos continúan evolucionando, comprender y aprovechar los hilos de proceso se vuelve cada vez más importante para desarrollar aplicaciones eficientes y escalables.

La importancia de los hilos radica en su capacidad para aprovechar al máximo los recursos del sistema, como la memoria y la capacidad de procesamiento, y permitir que los programas realicen múltiples tareas simultáneamente. Sin embargo, también es importante tener en cuenta que el uso inadecuado de procesos e hilos puede conducir a problemas de rendimiento y seguridad, como cuellos de botella, errores de sincronización y problemas de acceso a datos compartidos.

La implementación de hilos en Python resulto bastante sencillo gracias a que Python cuenta con bibliotecas de procesamiento en paralelo como en este caso lo es Threading.

## Referencias


[1] J. Lozano. "Tipos de datos básicos de Python" 2019. https://j2logo.com/python/tutorial/tipos-de-datos-basicos-de-python/#tipos-datos-otros

[2 ]P. Bovet, M. Cesati, “Understanding The Linux Kernel Understanding The Linux Kernel” 3rd Edition 3rd Edition O'Reilly Media, Inc., 2005.

[3] M . Kerrisk. “The Linux Programming Interface”. San Francisco. 2022. No Starch Press. (617-631).

[4] Baeldung. (2022). “Linux Process vs. Thread”. Disponible en: https://www.baeldung.com/linux/process-vs-thread#:~:text=A%20thread%20is%20a%20lightweight%20process%20also%20called%20an%20LWP,slower%20due%20to%20isolated%20memory.

# Anexo. Códigos completos.

## Código del programa.

```
import threading
import random
import numpy as np

def crearMatriz(mat):
    print(f'{threading.current_thread().name} te saluda')
    mat[0,:]=np.arange(1,10)
    mat[1,:]=np.arange(1,18,2)
    mat[2,:]=np.arange(2,19,2)
    print(f'{threading.current_thread().name} se despide')


def ejecutar(matriz,fila,suma):
    print(f'{threading.current_thread().name} te saluda')
    for i in range (9):
        suma[fila]+=matriz[fila,i]

    print(f'{threading.current_thread().name} se despide')


mat=np.zeros((3,9))
suma=[0]*3
hilos=[None]*4

#Creacion hilo 1
hilos[0] = threading.Thread(target=crearMatriz, name='Hilo
1',args=(mat,))
hilos[0].start()
hilos[0].join()

#Creacion hilos 2 al 4
for i in range (1,4):
  hilos[i] = threading.Thread(target=ejecutar, name=f'Hilo
{i+1}',args=(mat,i-2,suma))
  hilos[i].start()
  hilos[i].join()

print("Matriz de numeros: \n", mat)
print("Suma de las filas:",suma)
```