**UNIVERSIDAD DE GUADALAJARA**

**CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERIAS**

**Materia:** Computación tolerante a fallas

**Clave:** I7036

**Sección:** D06


**NRC:** 179961

**Hilos y Procesos**

**Código estudiante:** 221350567

**Alumno:** Rauf Alfonso Hamden Estrada

**Carrera:** Ingeniería en Computación

**Fecha:** 11/09/2023

**Docente:** Michel Emanuel Lopez Franco

2023B

## Introducción

Los hilos son unidades de ejecución más pequeñas dentro de un proceso. Puedes ser subprogramas independientes que pueden ejecutarse de manera simultánea dentro de un programa principal. Los hilos comparten la misma memoria y recursos dentro de un proceso, lo que los hace eficientes en términos de uso de recursos, pero también introduce desafíos en la sincronización y la comunicación entre hilos.

Por otro lado, los procesos son programas en ejecución que pueden contener uno o varios hilos. Cada proceso tiene su propio espacio de memoria y recursos, lo que los hace independientes entre sí. Los procesos son ideales para la separación de tareas y la protección de datos, pero pueden consumir más recursos que los hilos debido a la duplicación de recursos.

## Desarrollo

Importamos las librerías para la creación de hilos y procesos  y definimos las funciones con un for de un rango de 5 y un tiempo de espera de un segundo

```python
import threading
from threading import *
import multiprocessing
import time

def proceso_hilo(nombre):
    for i in range(5):
        print(f"{nombre}: Ejecución {i + 1}")
        time.sleep(1)

def procesamiento(nombre):
    for i in range(5):
        print(f"Proceso: Ejecución {i + 1}")
        time.sleep(1)
```

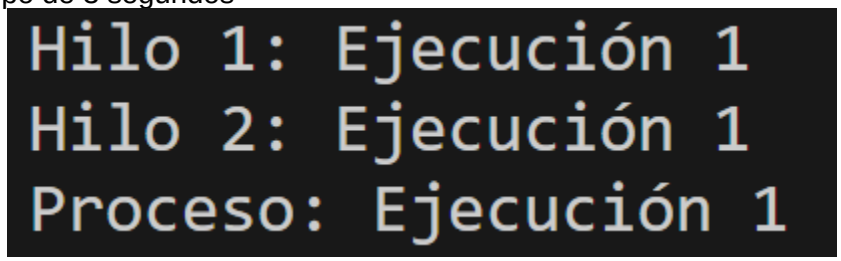
Creamos los hilos y el tercer hilo con una espera de 3 segundos para que pueda ejecutarse

```
def hola():
  print("Hola mundo")

if __name__ == "__main__":
    # Creamos los hilos
    hilo = threading.Thread(target=proceso_hilo, args=("Hilo 1",))
    hilo2=threading.Thread(target=proceso_hilo,args=("Hilo 2",))
    hilo3 = Timer(3,hola,)

    # Crear un proceso
    proceso = multiprocessing.Process(target=procesamiento, args=("Proceso 1",))#creamos el proceso

    # Configuro el proceso como demonio
    proceso.daemon = True
```

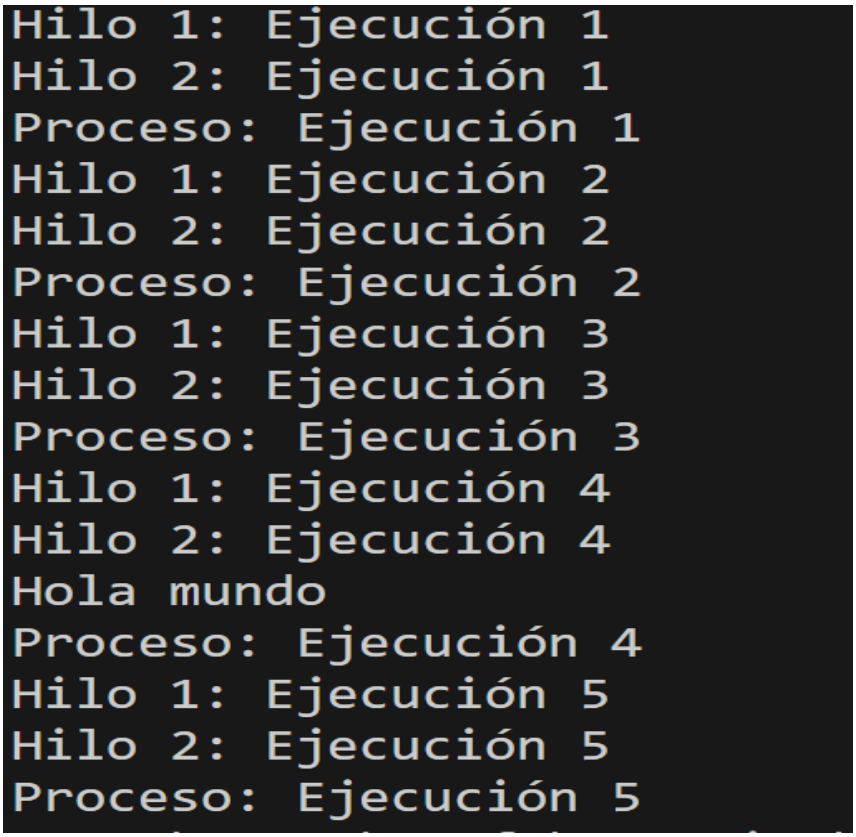
Aquí creamos un proceso e inicializamos tanto los hilos como el proceso y con un join hacemos que espere a que termine el hilo

```
# Crear un proceso
proceso = multiprocessing.Process(target=procesamiento, args=("Proceso 1",))#creamos el proceso

# Configuro el proceso como demonio
proceso.daemon = True

# Iniciamos los hilos y el proceso
hilo.start()
hilo2.start()

proceso.start()
hilo3.start()

# Esperar a que el hilo termine
hilo.join()
hilo2.join()
```

Esta es la primera ejecución de los procesos  como podemos observar tenemos el hilo 1 y el 2 y un proceso ,embargo el hilo numero 3 no se va ha ejecutar hasta que no haya pasado el periodo de tiempo de 3 segundos



Aquí tenemos los resultados de la ejecución de nuestro programa concurrente ya que ejecuta varios a la vez y a al espera de unos 3 segundos aparece nuestro tercer hilo ejecutando una función con el hola mundo y el proceso




## Conclusión

La utilización tanto de hilos como de procesos nos pueden servir mucho de utilidad depende de lo que el programador o la persona tenga planeado hacer con ello. La elección entre hilos y procesos depende de los requisitos específicos de la aplicación y de cómo se gestionen los recursos y la comunicación entre las tareas concurrentes. En general es importante tener en cuenta los recurso que podemos nosotros ahorrar con la utilización tanto de hilos como procesos

## Enlaces

Multiprocessing — paralelismo basado en procesos. (s. f.). Python documentation. https://docs.python.org/es/3/library/multiprocessing.html

Fernandez, R. (2019, 28 mayo). Hilos en Python - Garantizados. https://unipython.com/hilos-python/