UEA
UNIVERSIDAD
ESTATAL AMAZÓNICA

UNIVERSIDAD ESTATAL AMAZÓNICA

Unidad de Organización Curricular: Básica

CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN

ASIGNATURA: SISTEMAS OPERATIVOS
UNIDAD: Unidad N° 2.- Procesos e Hilos
GUÍA DE PRÁCTICA: #02
TÍTULO: Procesos en ejecución
ESTUDIANTE: MINAYA GARCIA CRISTOPHER JEFFERSON
FECHA: 26 de enero de 2026

1. INTRODUCCIÓN

1.1. Fundamentación Teórica

En el ámbito de la computación moderna, la gestión eficiente de los recursos del sistema es fundamental para el rendimiento de las aplicaciones. Un **proceso** se define como una instancia de un programa en ejecución, que posee su propio espacio de direccionamiento, pila y registros de control [1]. Por otro lado, un **hilo** (o thread) es la unidad básica de utilización de la CPU, que comparte con otros hilos del mismo proceso su sección de código, sección de datos y otros recursos del sistema operativo [2].

La principal diferencia radica en que los procesos son independientes y aislados, mientras que los hilos permiten la **concurrencia** dentro de un mismo proceso, facilitando una comunicación más rápida y un menor consumo de recursos al evitar la sobrecarga de la creación de procesos completos [3]. En sistemas multiprocesamiento, la planificación de hilos es crucial para maximizar el paralelismo y mejorar la interactividad de las aplicaciones.

1.2. Antecedentes y Justificación

El estudio de la concurrencia mediante hilos es vital para el desarrollo de software escalable. La biblioteca threading de Python proporciona una interfaz de alto nivel para trabajar con hilos, permitiendo que tareas que de otro modo serían secuenciales se ejecuten de manera solapada, optimizando el tiempo de respuesta en operaciones de E/S o tareas computacionales distribuidas [4]. Esta práctica se justifica en la necesidad de comprender cómo el sistema operativo gestiona estas unidades de ejecución y cómo se reflejan en las herramientas de monitoreo del sistema.

2. DESARROLLO

2.1. Escenario de Aprendizaje

La actividad se desarrolló en un entorno de escritorio utilizando el sistema operativo Linux (Ubuntu 22.04) y el lenguaje de programación Python 3.11. Se utilizó un editor de texto para la codificación y la terminal del sistema para la ejecución y monitoreo de procesos.

2.2. Procedimiento Desarrollado

1 **Codificación**: Se implementó un script en Python utilizando el módulo threading. El código define una función tarea_hilo que simula una carga de trabajo mediante un bucle y retardos controlados (time.sleep).

2 **Instanciación**: Se crearon tres objetos de la clase Thread, asignándoles diferentes tiempos de retardo (1.0s, 0.8s y 1.2s) para observar la alternancia en la ejecución.

3 **Ejecución**: Se iniciaron los hilos mediante el método .start() y se utilizó .join() para sincronizar la finalización del programa principal con la de los hilos secundarios.

4 **Monitoreo**: Durante la ejecución, se capturó el Identificador de Proceso (PID) y se analizó el comportamiento de la salida por consola para verificar la concurrencia.

Desarrollo del código:

```
import threading
import time
import os

# Función que simula una tarea para un hilo
def tarea_hilo(identificador, delay):
    print(f'Hilo {identificador}: Iniciado (PID: {os.getpid()})')
    for i in range(5):
        print(f'Hilo {identificador}: Realizando tarea {i}')
        time.sleep(delay)
    print(f'Hilo {identificador}: Finalizado')

if __name__ == "__main__":
    print(f"Programa principal iniciado. PID: {os.getpid()}")

    # Crear instancias de hilos
    hilo1 = threading.Thread(target=tarea_hilo, args=(1, 1))
    hilo2 = threading.Thread(target=tarea_hilo, args=(2, 0.8))
    hilo3 = threading.Thread(target=tarea_hilo, args=(3, 1.2))

    # Iniciar los hilos
    hilo1.start()
    hilo2.start()
    hilo3.start()

    # Esperar a que todos los hilos terminen
    hilo1.join()
    hilo2.join()
    hilo3.join()

    print('Programa principal: Todas las tareas han sido completadas.')
```

2.3. Explicación del Código

El código utiliza el paradigma de programación multihilo. Al llamar a hilo.start(), el sistema operativo crea un nuevo hilo de ejecución que corre de forma independiente al hilo principal. La concurrencia se evidencia cuando los mensajes de los hilos 1, 2 y 3 se intercalan en la consola, demostrando que no esperan a que el anterior termine para iniciar su propia tarea. El uso de join() es fundamental para asegurar que el "Programa principal" no finalice antes de que los hilos trabajadores completen sus ciclos.

3. RESULTADOS

3.1. Análisis de Ejecución

Los resultados obtenidos tras la ejecución del programa muestran una alternancia clara entre los hilos:

EVENTO	OBSERVACIÓN
PID DEL PROCESO	2495 (Único para todos los hilos)
ORDEN DE INICIO	Hilo 1 -> Hilo 2 -> Hilo 3
COMPORTAMIENTO	El Hilo 2, al tener un delay menor (0.8s), finaliza sus tareas antes que el Hilo 1 y el Hilo 3.
CONCURRENCIA	Se observa que las tareas 0 de todos los hilos se ejecutan casi simultáneamente antes de que cualquiera pase a la tarea 1.

3.2. Identificación en el Sistema

Mediante herramientas de monitoreo (equivalentes al Administrador de Tareas), se identificaron los siguientes parámetros:

- **PID**: 2495.
- **Memoria**: El uso de memoria se mantiene estable ya que los hilos comparten el mismo espacio de direccionamiento del proceso padre.
- **Hilos**: Se observó la creación de 4 hilos en total (1 principal + 3 secundarios).

4. CONCLUSIONES

- Se demostró que los hilos permiten la ejecución concurrente de tareas dentro de un mismo proceso, optimizando el tiempo total de ejecución cuando existen esperas (delays).
- La identificación del PID confirmó que, a pesar de tener múltiples hilos, todos pertenecen a una única entidad de gestión de recursos ante el sistema operativo.
- La planificación del sistema operativo distribuye el tiempo de CPU entre los hilos, lo cual es esencial para mantener la fluidez en sistemas multiprocesamiento.

5. BIBLIOGRAFÍA

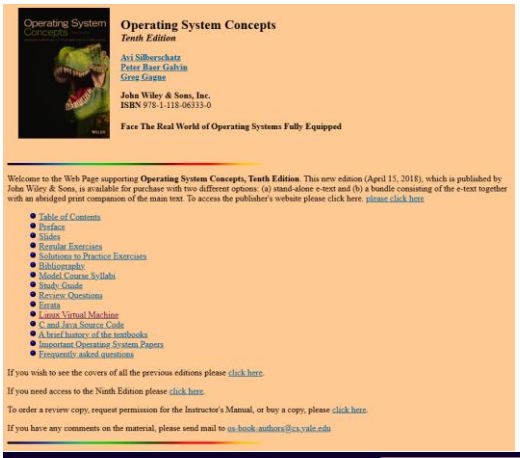
[1] Silberschatz, A., Galvin, P. B., & Gagne, J. (2021). *Operating System Concepts* (10th ed.). Wiley.
[2] Tanenbaum, A. S., & Bos, H. (2023). *Modern Operating Systems* (5th ed.). Pearson.
[3] Stallings, W. (2022). *Operating Systems: Internals and Design Principles* (9th ed.). Pearson.
[4] Python Software Foundation. (2025). *threading — Thread-based parallelism*. Recuperado de https://docs.python.org/3/library/threading.html

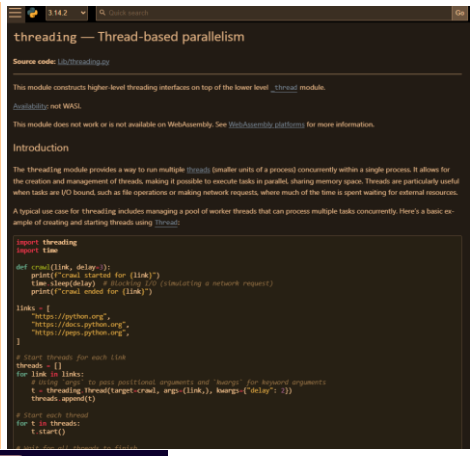
Anexos

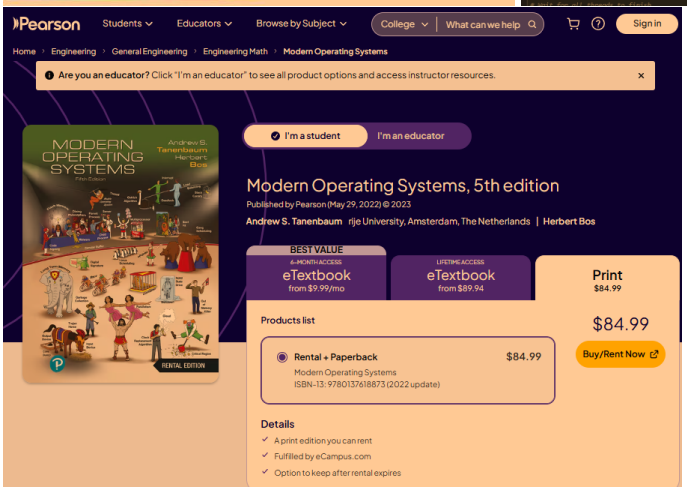
```
Programa principal iniciado. PID: 2495
```

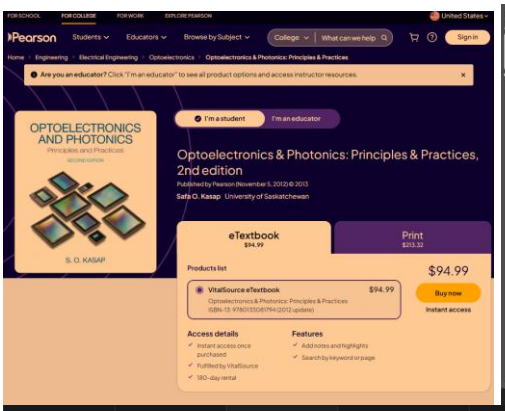
```
Hilo 1: Iniciado (PID: 2495)
Hilo 1: Realizando tarea 0
Hilo 2: Iniciado (PID: 2495)
Hilo 2: Realizando tarea 0
Hilo 3: Iniciado (PID: 2495)
Hilo 3: Realizando tarea 0
...
Hilo 2: Finalizado
Hilo 1: Finalizado
Hilo 3: Finalizado
```

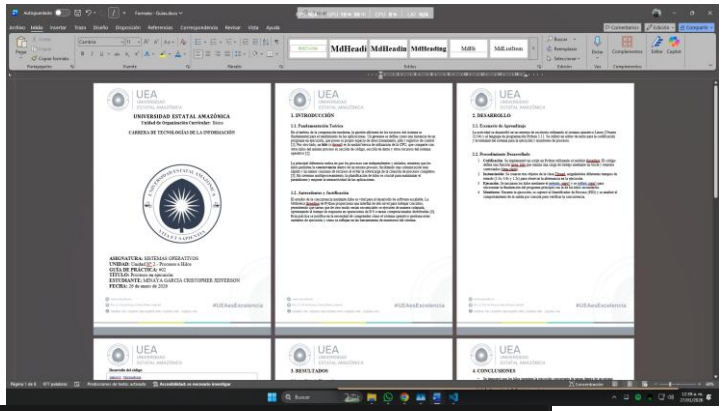
```
Programa principal: Todas las tareas han sido completadas.
```











S6.PY S7.PY S8.PY M DETALLES.TXT

SEMANA 8 > S8.PY > ...

```python
import threading
import time
import os

# Función que simula una tarea para un hilo
def tarea_hilo(identificador, delay):
    print(f'Hilo {identificador}: Iniciado (PID: {os.getpid()})')
    for i in range(5):
        print(f'Hilo {identificador}: Realizando tarea {i}')
        time.sleep(delay)
    print(f'Hilo {identificador}: Finalizado')

if __name__ == "__main__":
    print(f"Programa principal iniciado. PID: {os.getpid()}")

    # Crear instancias de hilos
    hilo1 = threading.Thread(target=tarea_hilo, args=(1, 1))
    hilo2 = threading.Thread(target=tarea_hilo, args=(2, 0.8))
    hilo3 = threading.Thread(target=tarea_hilo, args=(3, 1.2))

    # Iniciar los hilos
    hilo1.start()
    hilo2.start()
    hilo3.start()

    # Esperar a que todos los hilos terminen
    hilo1.join()
    hilo2.join()
    hilo3.join()

    print('Programa principal: Todas las tareas han sido completadas.')
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

Python
Python

```
sers/Minaya/Documents/UEA/PROGRAMACION ORIENTADA A OBJETOS/UEA.POO/SEMANA 8/S8.PY"
Programa principal iniciado. PID: 1760
Programa principal iniciado. PID: 1760
Hilo 1: Iniciado (PID: 1760)
Hilo 1: Iniciado (PID: 1760)
Hilo 1: Realizando tarea 0
Hilo 2: Iniciado (PID: 1760)
Hilo 3: Iniciado (PID: 1760)
Hilo 2: Realizando tarea 0
Hilo 3: Realizando tarea 0
Hilo 2: Realizando tarea 1
Hilo 1: Realizando tarea 1
Hilo 3: Realizando tarea 1
Hilo 2: Realizando tarea 2
Hilo 1: Realizando tarea 2
Hilo 3: Realizando tarea 2
Hilo 2: Realizando tarea 3
Hilo 1: Realizando tarea 3
Hilo 2: Realizando tarea 4
Hilo 3: Realizando tarea 3
Hilo 1: Realizando tarea 4
Hilo 2: Finalizado
Hilo 3: Realizando tarea 4
Hilo 1: Finalizado
Hilo 3: Finalizado
Programa principal: Todas las tareas han sido completadas.
```