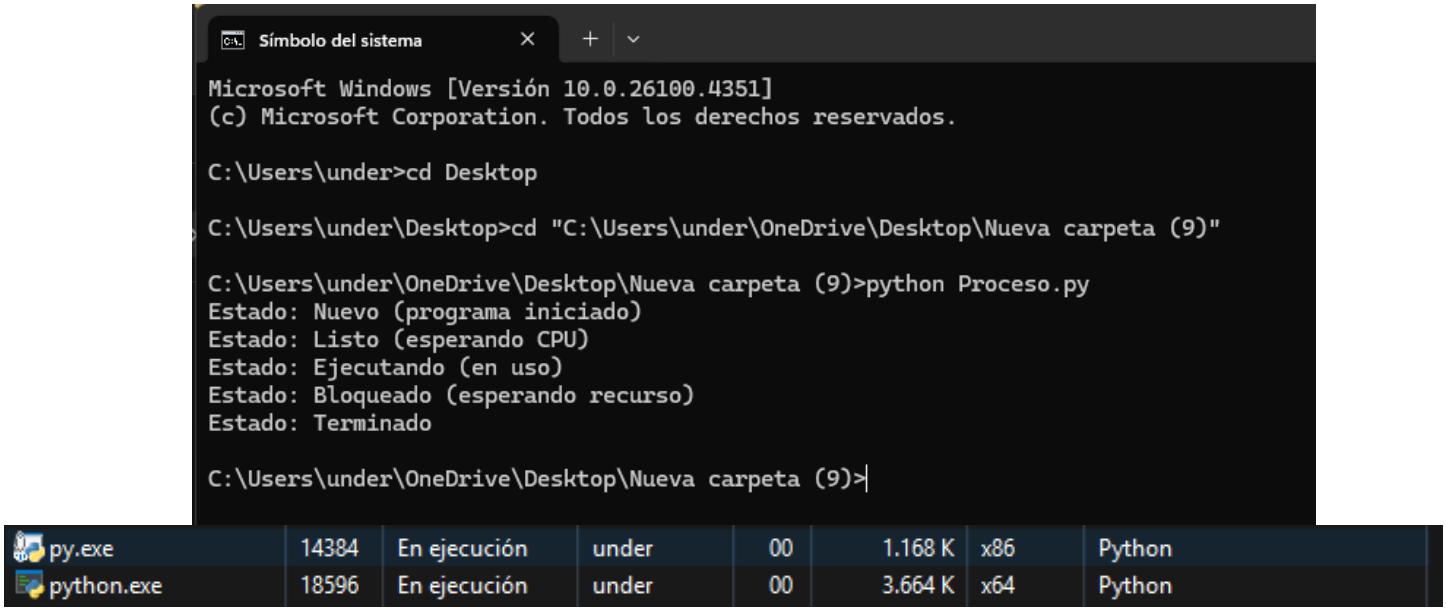
Laboratorio 1 - Gestión de Procesos

Este informe documenta los resultados del laboratorio 1, enfocado en la Gestión de Procesos.

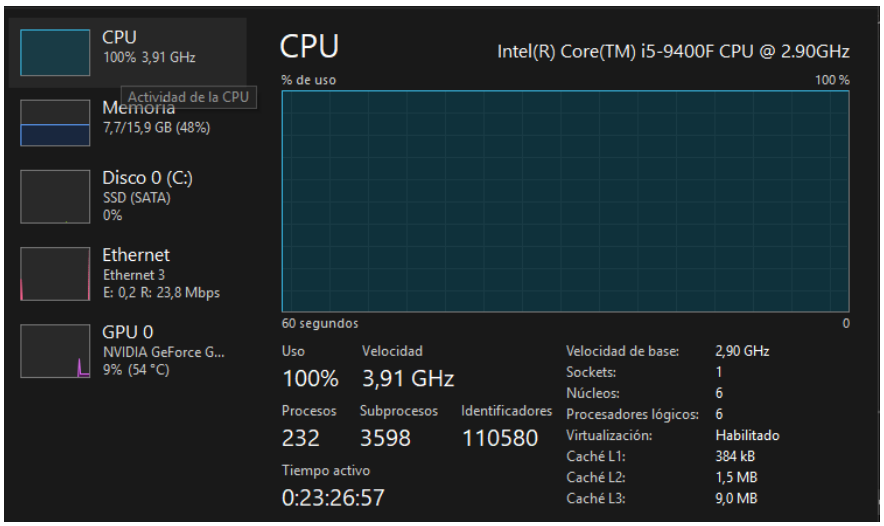
1. Simulación de Estados de Proceso:

Se creó un script en Python llamado 'proceso.py' que simula los estados: Nuevo, Listo, Ejecutando, Bloqueado y Terminado. Se midieron los tiempos de transición entre estados y se tomo captura del mismo junto como se veía en el administrador de tareas también



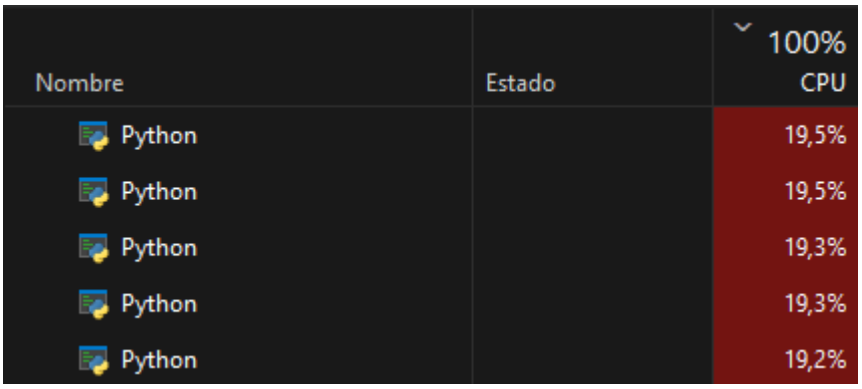
2. Scheduling de CPU:

Se ejecutaron múltiples scripts 'cpu_loader.py' para simular alta carga en el CPU. Se observó cómo el sistema operativo distribuye el uso del CPU entre los procesos en ejecución. Las observaciones se compararon con los algoritmos FIFO y Round Robin.



3. Comparación con algoritmos teóricos de planificación:

Durante la ejecución simultánea de cinco scripts que consumen CPU (cpu_loader.py), se observó en el Administrador de tareas que el uso del CPU se reparte casi equitativamente entre los cinco procesos. Esto se refleja en la captura adjunta, donde cada proceso de Python consume aproximadamente el 19-20% del CPU.

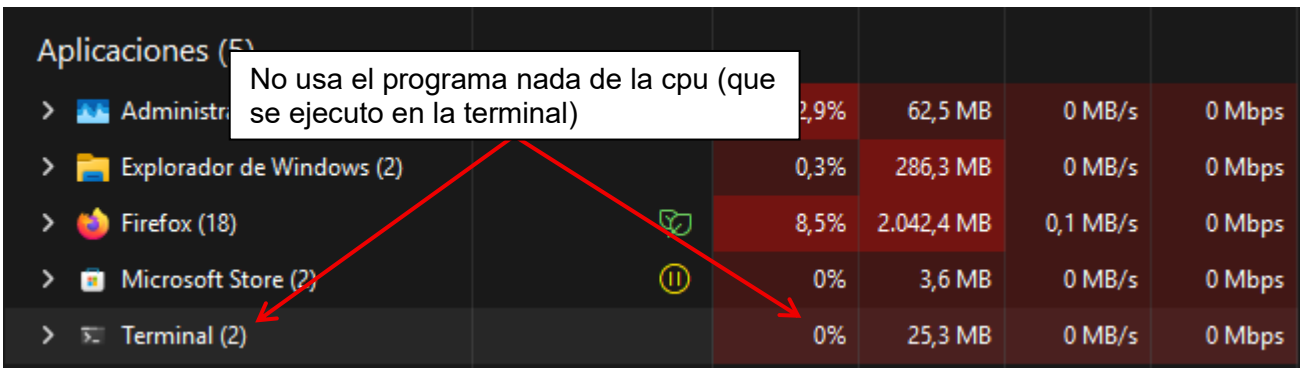


Esta distribución es coherente con la planificación Round Robin, uno de los algoritmos clásicos utilizados por los sistemas operativos modernos. En este algoritmo, cada proceso recibe un "quantum" de tiempo de CPU, y luego se pone en espera para que otro proceso pueda ejecutarse. Esto garantiza equidad y uso compartido del procesador, tal como se observa en la práctica.

En cambio, si el sistema usara planificación FIFO (First In, First Out), el primer proceso que llega se ejecutaría hasta completarse, y los demás esperarían su turno sin compartir CPU. En ese caso, el Administrador de tareas mostraría un único proceso consumiendo la mayoría de los recursos, lo cual no ocurre en esta prueba.

4. Deadlock:

Se simuló un deadlock real mediante el uso de hilos y bloqueos (locks) en Python. Dos hilos intentan acceder a recursos bloqueados por el otro, generando una espera mutua infinita. Se documentaron los mensajes en consola y se explicó el fenómeno, por lo cual en la siguiente captura vemos como no usa recursos de la cpu



Archivos incluidos:

- proceso.py
- cpu_loader.py
- deadlock.py
- Capturas de pantalla de cada estado del proceso y del uso del CPU
- datos_mediciones.xlsx con tiempos registrados

Conclusión:

Este laboratorio permitió observar de manera práctica el ciclo de vida de un proceso, cómo el sistema operativo maneja múltiples procesos de forma concurrente, y cómo un mal manejo de recursos puede generar deadlocks. Estas observaciones son fundamentales para entender el comportamiento interno del sistema operativo que en este caso se utiliza Windows 11 home.