tareas\_SIN\_hilos.py

* ¿Qué se puede notar con respecto al tiempo de ejecución? ¿Es predecible?

El programa es más o menos predecible, pero no al 100%. La velocidad de ejecución de la tarea 1 puede variar en cada ejecución debido a diferencias en la velocidad del procesamiento y otros factores. En cambio, las tareas 2 y 3 son más constantes en términos de tiempo.

* Nombrar un proceso o función de la vida real que pueden ser considerados procesos de “máxima velocidad posible” que dependen casi exclusivamente de la velocidad de la máquina que los ejecuta.

Un ejemplo común es la compilación de código fuente a código máquina en lenguajes como C# o Java. La velocidad de este proceso depende principalmente de la velocidad del CPU y otros recursos del sistema operativo.

* Nombrar un proceso o función de la vida real que pueden ser considerados procesos de “velocidad de respuesta no dependiente de la velocidad de procesamiento” o que sea de naturaleza impredecible o externa.

La comunicación en red es un ejemplo. Cuando un programa solicita datos a través de la red, la velocidad de respuesta puede variar por factores externos como congestión de red, calidad de la conexión y distancia entre el cliente y el servidor.

tareas\_CON\_hilos.py

* ¿Qué se puede notar con respecto al tiempo de ejecución? ¿Se mejoró el tiempo de respuesta con respecto al mismo programa sin hilos?

Sí, el tiempo de ejecución mejoró significativamente en comparación con el mismo programa sin hilos. Esto se debe a que las tareas se ejecutan simultáneamente en diferentes hilos, aprovechando mejor los recursos de la CPU, especialmente en sistemas con múltiples núcleos.

* ¿Completan las funciones su ejecución en el orden establecido?

Aunque los hilos se inician en el orden en que se crean, no se garantiza que terminen en el mismo orden. Debido a la naturaleza concurrente de los hilos, el sistema operativo puede asignar recursos de CPU de manera diferente, afectando el orden de finalización.

* Nombrar un escenario real donde el multihilado puede mejorar considerablemente el tiempo de respuesta de un sistema.

Un ejemplo sería un servidor web que maneja múltiples solicitudes simultáneas. Cada solicitud podría ser manejada por un hilo separado, permitiendo al servidor responder más rápidamente a los usuarios.

sumador-restador.py

* Nos da un resultado neto igual a 0 al final de la ejecución del programa.

sumador-restador\_CON\_race.py

* ¿Qué se puede notar con respecto al tiempo de ejecución?

El tiempo de ejecución puede variar debido a los hilos. En cada ejecución, los hilos pueden ser programados y ejecutados en diferentes momentos, lo que afecta el tiempo total que tarda el programa en completarse.

* ¿Qué sucede con el valor final del acumulador y por qué sucede?

El valor final del acumulador puede variar en cada ejecución porque múltiples hilos intentan acceder y modificar la misma variable compartida al mismo tiempo, sin sincronización. Esto puede llevar a resultados incorrectos o inconsistentes.

* ¿Cómo se puede corregir esta condición de carrera sin dejar de utilizar hilos?

Para evitar estos problemas, se puede usar un mecanismo de sincronización como un "bloqueo" (lock). Esto garantiza que solo un hilo pueda acceder y modificar la variable compartida a la vez, asegurando que los cambios se realicen de manera ordenada y correcta.

sumador-restador\_SIN\_race.py