Arquitectura de sistemas operativos

Facundo Strack

**TRABAJO PRÁCTICO N3**

**Procesos Multihilados**

**2- Hilos:**

**Ejercicios con tareas\_SIN\_hilos.py y tareas\_CON\_hilos.py**

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

Con respecto al tiempo de ejecución se puede notar cierta variabilidad en la Tarea\_1 ya que el mismo va a depender en gran medida de la capacidad del CPU y de las condiciones de carga del sistema. Sin embargo , es predecible en la Tarea\_2 y Tarea\_3 ya que sus instrucciones dependen de la función time.sleep que introduce pausas fijas.

**B-** 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 (ej.Ordenar una lista)

Un proceso o función de la vida real que pueden ser considerados procesos de máxima velocidad posible puede ser por ejemplo el renderizado de graficos 3D que ocurre mientras se juega un videojuego el cual es una tarea en tiempo real que no puede tener prácticamente demoras en su funcionamiento , ya que esto generaría una mala experiencia.

El renderizado 3D es un proceso tangible que muchos han experimentado o al menos visto, y la diferencia en tiempos de renderizado entre hardware antiguo y moderno es evidente y fácil de comprender.

**C-** 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 (ej. Leer un archivo externo)

Un proceso o función de la vida real que pueden ser considerados procesos de velocidad de respuesta no dependiente de la velocidad de procesamiento puede ser por ejemplo , esperar la entrada del usuario.

Una aplicación que espera que el usuario presione una tecla o haga clic en un botón

no depende de la capacidad de procesamiento de la máquina, sino de factores externos como la velocidad y disponibilidad del usuario. Este proceso es inherentemente impredecible porque depende de una acción externa.

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

Se puede notar que ahora con el uso de hilos , el tiempo de ejecución total del programa se reduce significativamente en comparación con la versión sin hilos , esto se debe a que las tareas se ejecutan paralelamente en lugar de secuencialmente.

Sí, el tiempo de respuesta se mejora notablemente. En la versión sin hilos, el tiempo total es aproximadamente la suma de los tiempos de todas las tareas. En la versión con hilos, el tiempo total es aproximadamente el tiempo de la tarea más larga, dado que las tareas se ejecutan simultáneamente.

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

No, las funciones no necesariamente completan su ejecución en el orden establecido. Dado que se están ejecutando en hilos separados, el orden en el que se inician y completan puede variar cada vez que se ejecuta el programa.

**F-** Nombrar un escenario real donde el multi-hilado puede mejorar considerablemente el tiempo de respuesta de un sistema (ej. Carga de una página WEB en un navegador).

Un escenario real donde el multi-hilado puede mejorar considerablemente el tiempo de respuesta de un sistema puede ser no solo el uso de una pagina web por parte de un usuario o cliente sino visto desde el lado del servidor donde el sitio recibe múltiples solicitudes de los clientes (navegadores web) casi simultáneamente. Cada solicitud puede requerir la lectura de datos desde una base de datos.

Utilizando multi-hilado, el servidor puede procesar múltiples solicitudes concurrentemente, mejorando así la capacidad de respuesta y reduciendo el tiempo de espera de los usuarios.

**3- Condición de carrera (Race Condition)**

**Ejercicios con sumador-restador.py y sumador-restador-CON\_race.py**

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

Con respecto a los tiempos de ejecución puede notarse que la función se ejecuta más rápido en el código con hilos ya que las tareas se ejecutan en paralelo y no en serie.

**B-** ¿Qué sucede con el valor final del acumulador?

El valor final del acumulador es 0 en el código con 1 hilo. Esto se debe a que las operaciones de suma y resta se cancelan entre sí perfectamente, ya que se ejecutan secuencialmente y no hay interferencia entre las operaciones.

Sin embargo en el código con hilos. El valor final del acumulador es impredecible y probablemente no será 0. La condición de carrera causada por los accesos concurrentes no controlados a acumulador desde ambos hilos lleva a resultados inconsistentes.

**C-** ¿Por qué sucede esto?

Esto ocurre debido a una condición de carrera que seria cuando ambos hilos (sumador y restador) acceden a la variable acumulador al mismo tiempo por lo tanto hay un problema con la secuencia de lectura y escritura.

Un hilo puede leer el valor de acumulador y antes de que lo actualice, el otro hilo también puede leer el mismo valor, llevando a una pérdida de actualización. Esto se traduce en resultados incorrectos en el valor final de acumulador.

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

La condición de carrera se puede corregir utilizando mecanismos de sincronización para garantizar que solo un hilo acceda a la variable acumulador a la vez. Una solución común es usar un candado (lock). Con la función threading.Lock

**4- Detección y corrección del problema**

**Ejercicios con sumador-restador\_SIN\_race.py**

**A-** ¿Qué sucede con el valor final del acumulador?

El uso de threading.Lock en el código garantiza que solo un hilo puede ejecutar el bloque de código que modifica el acumulador a la vez.

Dado que las operaciones están sincronizadas, no hay condiciones de carrera y las sumas y restas se cancelan entre sí correctamente, resultando en un valor final de 0.

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

El tiempo de ejecución será mas lento ya que el uso de threading.Lock puede hacer que el tiempo total de ejecución sea más largo debido a la espera producida por el bloqueo y desbloqueo.