**Arquitectura y Sistemas Operativos**

**Tecnicatura Universitaria en Programación (TUP) – UTN FRBB**

**Profesor: Gustavo Ramoscelli**

**Ayudantes: Leandro M. Regolf - Sergio Antozzi**

**Trabajo Práctico N.º 3**

**Procesos multihilados (multi-threads)**

* Tareas\_SIN\_hilos.py

1. Con respecto al tiempo de ejecución se puede notar que cada vez que iniciamos el programa, cada tarea siempre varía el tiempo de demora y este nunca es el mismo. Por lo tanto, dicho tiempo no es predecible.
2. Un proceso o función que siempre depende de la velocidad de la maquina que los ejecuta puede ser el encendido de una PC o Notebook, las cuales dependiendo de la velocidad del procesador junto con la de la memoria determinan lo que tardan en iniciar las mismas.
3. Un proceso o función que no dependa de la velocidad de procesamiento puede ser la descarga de archivos por internet, ya que depende de la velocidad del internet al igual que del tamaño del mismo.

* Tareas\_CON\_hilos.py

1. Lo que se puede notar con respecto al tiempo de ejecución es una mejora notable en la ejecución de las tareas, ya que el mismo programa tardo 2 segundos menos en ejecutar las tareas.
2. Las funciones no se completan en el orden establecido, las mismas dependen de su velocidad para completarse, por lo que en mi caso la tarea 2 siempre termina antes que la tarea 1.
3. Un escenario real donde el multi-hilado puede mejorar el tiempo de respuesta es en el procesamiento de imágenes el cual incluye tareas como la carga, el filtrado, la transformación, la compresión y la renderización de imágenes. Estas tareas son comunes en aplicaciones de edición de imágenes, visión por computadora, y en la generación de gráficos en tiempo real.

* Sumador-restador.py/ Sumador-restador\_CON\_race.py

1. Con respecto al tiempo de ejecución no noto diferencia entre las mismas, la velocidad de ejecución es muy similar.
2. El valor final del acumulador varia cada vez que se ejecuta en el programa que esta con hilos.
3. Esto sucede porque los hilos “sumador” y “restador” están modificando el valor de “acumulador” sin estar sincronizados.
4. Para que el valor final del acumulador sea el correcto se debería incluir en el código el uso del “lock”, para que el resultado sea consistente y predecible.

* Sumador-restador\_SIN\_race.py

1. Lo que se puede ver con el valor final es que es constante y predecible.
2. Lo que se nota en cuanto a la velocidad de ejecución es que sigue teniendo la misma velocidad que antes, por lo que este cambio no tiene efectos negativos.