Práctica 4. Gestión Básica Hilos y Procesos

Disponible: 2 de octubre. 13:00h

Entrega: 10 de octubre. 23:55h

1 Introducción

Esta práctica va a consistir en la creación básica tanto de procesos como de hilos empleando las clases de Threads y de Process (y también las clases de Pool y ThreadPool), haciendo uso de las librerías de threading y multiprocessing. Se recomienda ver las últimas transparencias del tema de procesos e hilos. **Para esta práctica se hará uso del editor que queráis para programar en Python (e.g. Visual Studio Code) en el entorno que queráis (Windows, MAC, Linux). No habrá que instalarse nada adicional.**

2 Instrucciones de entrega

Como entrega, cada alumno deberá proporcionar en un zip todos los ficheros Python requeridos para cada uno de los apartados. De igual manera, en algunos apartados se pedirá responder a algunas preguntas, que pueden ser contestadas en los mismos ficheros ".py" como comentarios o en un fichero aparte llamado "memoria.pdf". **En cualquier caso, debe quedar bien definida la pregunta qué se está respondiendo en cada apartado**. El nombre del zip entregado debe seguir este formato:

```
<nombreAlumno>_<Apellido1>_<Apellido2>_Procesos_Hilos_Basico.zip
```

Por ejemplo:

```
Pablo_DelasHeras_Fernandez_Procesos_Hilos_Basico.zip
```

IMPORTANTE: recordad que a la hora de programar, no sólo importa la **funcionalidad**. También importa el **estilo de código, siguiendo la guía publicada de estilo de programación** y la **documentación**. **Por esta razón, todo código entregado debe seguir un estilo correcto de programación y todas las funciones/clases deben estar correctamente documentadas.**

3 Ejercicios de la práctica

3.1 Ejercicio1: Creación de un hilo en Python

Crea un fichero llamado "Ejercicio1_<Nombre>.py", sin los símbolos de < y > donde Nombre sea tu nombre de pila y empleando la librería de Thread, se pide:

- Crear una función que reciba como argumento un número que servirá un contador en segundos. La función debe mostrar en bucle el siguiente mensaje: "El usuario dispone de X segundos para introducir un nombre y un apellido". Dicho mensaje se repetirá cada cinco segundos mostrando en la variable de X el número de segundos que le queda (inicialmente X será el número recibido por argumento).

- El programa principal de ejecución (main) debe crear un hilo que ejecute la función mencionada en el punto anterior y esperar la entrada por pantalla del nombre y apellido del usuario (primero deberá pedirse el nombre, y cuando se haya introducido, deberá pedirse el apellido). En ambos casos, se puede simplemente llamar a la función input en el main. Tanto el nombre como el apellido deben almacenarse en variables independientes. Si a la hora de terminar el contador (será de 20 segundos) de la función creada no se ha introducido el nombre y el apellido, la función del contador debe mostrar el siguiente mensaje: "No se ha introducido ningún nombre" y terminar. En el caso de que SÍ se haya introducido tanto el nombre como el apellido dentro de los 20 segundos, la función main deberá mostrar el siguiente mensaje "Bienvenido/a <nombre> <apellido>". En caso de que se introduzca el nombre o el apellido después de los segundos estipulados, se mostrará en el main el siguiente mensaje: "Ha introducido el nombre demasiado tarde".

- El programa principal debe esperar correctamente a la finalización del hilo hijo, y no finalizará hasta que se hayan introducido tanto el nombre como el apellido (sea tarde o no).

Se incluye un ejemplo de la salida mostrada introduciendo el nombre dentro de los 20 segundos.

```
El usuario dispone de 20 segundos para introducir un nombre:
Nombre: Pablo
Apellido: Sanchez
Bienvenido Pablo Sanchez
```

Y otro ejemplo de la salida esperada si pasan esos 20 segundos sin introducir el nombre:

```
El usuario dispone de 20 segundos para introducir un nombre:
Nombre: pablo
Apellido: El usuario dispone de 15 segundos para introducir un
    nombre:
El usuario dispone de 10 segundos para introducir un nombre:
El usuario dispone de 5 segundos para introducir un nombre:
No se ha introducido ningun nombre
Sanchez
Ha introducido el nombre demasiado tarde
```

3.2 Ejercicio 2: Creación de hilos en Python (II)

Se pide implementar un programa en Python llamado "Ejercicio_<Nombre>.py", siguiendo la misma nomenclatura que en el apartado anterior, que cree una matriz de 10 x 10 (deberá poderse modificar el tamaño de la matriz de manera

sencilla en el programa) inicializado a nans (puedes hacer uso del nan de la librería de numpy). El main, deberá pedirle al usuario introducir un número por teclado y deberá crear tantos hilos como el número indicado por el usuario. Si el usuario no introduce un número, el programa no debe fallar. Cada hilo tendrá un índice que irá del número 1 hasta el n, y cada uno de ellos cambiará una posición aleatoria de la matriz con su valor de hilo (es decir, del 1 al n) 20 veces. Cada vez que actualice una de las posiciones de la matriz, el hilo deberá esperar 1 segundo. Finalmente, una vez que todos los hilos hayan hecho los cambios, se pide imprimir la matriz en el hilo principal y mostrar el número de valores que tiene cada hilo en un diccionario. Se muestra a continuación un ejemplo de la salida que debe producir el programa para 4 hilos:

```
[3, nan, 1, nan, 4, 2, 2, nan, 4, 1]
[2, 1, nan, nan, 3, 1, 2, nan, 4, 1]
[nan, 1, 1, 3, nan, 3, 1, nan, 1, 1]
[nan, 3, nan, 4, 4, 2, nan, nan, nan, 3]
[nan, 2, 1, nan, nan, 1, 4, 3, 4, nan]
[3, 2, 3, 1, 4, 1, 1, nan, 3, 3]
[1, 2, 4, 2, 4, 3, 4, nan, nan, 2]
[nan, nan, 2, nan, 3, nan, 2, 2, nan, nan]
[1, nan, 4, 4, 2, nan, 2, nan, 4, nan]
[nan, 4, 3, nan, nan, 4, 1, nan, 4, 2]
Resultado: {3: 14, 1: 18, 4: 17, 2: 16}
```

- Ahora cambia la creación de los hilos por procesos y vuelve a ejecutar el programa. ¿Qué ocurre ahora? ¿Por qué? Deja un comentario en el código indicando qué cambia.

3.3 Ejercicio 3: Creación de procesos en Python (I)

Crea un fichero llamado "Ejercicio3_<Nombre>.py", siguiendo la misma nomenclatura de los apartados anteriores y empleando la librería de Process de multiprocessing, se pide:

- Definir un main que pida un número al usuario. Si ese número es un 1, ejecutará la primera configuración y si el número es un 2, ejecutará la segunda configuración. En caso de introducir cualquier otro número, finalizaría la ejecución. Las configuraciones son las siguientes:

- Configuración 1: el proceso principal creará tres procesos independientes (llamados proceso-hijo1, proceso-hijo2, proceso-hijo3). Podéis usar el atributo name de Process para darle un nombre a cada proceso. Cada uno de esos tres procesos se quedarán ejecutando un bucle infinito (while True). El proceso principal esperará 15 segundos y terminará cada uno de los tres procesos (se terminará el primer proceso, después de 5 segundos el segundo proceso y después de 5 segundos el tercero). Cada proceso (incluido el principal), debe imprimir una vez su nombre, su identificador y

cada uno de los procesos hijos deben imprimir tanto su id como el id de su padre.

- Configuración 2: el proceso principal creará un proceso hijo. Dicho proceso hijo creará otro proceso hijo y ese proceso hijo creará otro proceso hijo. Cada uno de esos procesos hijos se quedará ejecutando un bucle infinito. El proceso principal, después de 15 segundos, terminará el primer proceso creado y luego finalizará su ejecución. Cada proceso (incluido el principal), debe imprimir una vez su nombre, su identificador y cada uno de los procesos hijos deben imprimir además el id de su padre.

Una vez programado el código, se pide responder a las siguientes preguntas:

- ¿Qué ocurre en el primer caso? ¿Los procesos finalizan de manera correcta?
- ¿Qué ocurre en el segundo caso? ¿Los procesos finalizan de manera correcta?

3.4 Ejercicio 4: Pequeño benchmark

Se pide implementar la siguiente función que aproxima el número π en un fichero llamado "Ejercicio4_<Nombre>.py", siguiendo la misma nomenclatura que en los formatos anteriores:

$$\pi = \sqrt{6 \sum_{n=1}^{\infty} \frac{1}{n^2}} \tag{1}$$

Se pide crear un main que ejecute esa función con los siguientes valores: 1.000.000, 2.000.000, 4.000.000, 8.000.000 (son los valores a sustituir como "infinito" en la función). Anotad el tiempo que tarda en ejecutarse dicha función (comprobando que la función está bien implementada). Posteriormente, se pide generar 4 procesos que ejecuten esa función con esos valores. En este caso, haremos que cada proceso se encargue de operar con cada uno de esos valores (es decir, dividir la tarea entre los 4 procesos). Cada proceso debe imprimir una salida con el siguiente formato (con el valor de la lista asignado):

```
Con un valor de <valor_lista>, el resultado es <resultado_funcion
    >, con un tiempo de <segundos_empleado>
```

Ambas versiones (la de un único procesador y la de múltiples procesadores) deben incluirse en el fichero, de forma que se pueda elegir entre una y otra en el propio main a decisión del usuario (pidiéndosela por pantalla). Calcula el speedUp obtenido frente a la versión de un solo procesador (esto lo puedes hacer a mano).

3.5 Ejercicio 5: Terminación de procesos

Implementa una función que recibido un número por argumento (lo llamaremos n), haga n sumas de números aleatorios entre 0 y 1. Después de esa suma debe dormir 20 segundos. Posteriormente, crea un main que permita mediante una variable (definida en el propio main), crear 2 procesos o 2 hilos en función de dicha variable que realicen la función anteriormente mencionada. El primer proceso/hilo con un valor de 10.000.000 y el segundo con 100.000.000. Pásate el código a un entorno linux (sea cluster, máquina virtual o wsl). Ejecuta la versión con hilos en una terminal, y en otra, muestra con este comando que observas:

```
ps u | grep python3
```

Recuerda que el comando ps nos permite saber los detalles de los procesos y con el grep python3, estaríamos filtrando los detalles de los programas que están ejecutando python3. Para el código de hilos pulsando CTRL+C y ejecuta la versión con procesos. Vuelve a ejecutar el comando anterior otra terminal. ¿Qué observas y por qué? La segunda columna recuerda que indica el identificador del proceso. Si quieres matar el proceso, puedes ejecutar el comando dede otra terminal:

```
kill -9 id_proceso
```

Donde el id del proceso es la segunda columna. Lo veremos más adelante, pero con ese comando le estamos mandando una señal para matar un proceso. Llama a tu código "Ejercicio5_<Nombre>".py y súbelo a la entrega.

3.6 Ejercicio Opcional

3.6.1 Apartado a

Crea un fichero llamado "Extra_a.py" y empleando la librería de Process de multiprocessing, se pide:

- Definir una función main que cree un proceso
- El proceso creado debe realizar una tarea de sumar 70.000.000 de elementos (creados en un bucle de 1 a 70.000.000 que vaya sumando cada elemento, es decir, que haga la suma de 1 + 2 + 3 + ... + 70.000.000). Una vez que haya terminado esa suma, debe imprimirla por pantalla.
- El proceso principal debe esperar 3 segundos a que el proceso secundario termine dicha suma. Después de ese segundo, el primer proceso debe terminar la ejecución del segundo proceso.

3.6.2 Apartado b

Una vez completado el programa anterior, comprueba que la salida es algo similar a esto (es decir, que no muestra la suma de los elementos). En caso de que si la muestre, amplía el contador.

```
Procedemos a terminar el proceso
```

Se pide ahora:

- Que el segundo proceso se reparta la tarea con un hilo adicional para poder cumplir la restricción de terminar la ejecución de la suma en 3 segundos (se puede partir los 70.000.000 del contador en distintos bloques si se desea). Incluye ese programa en el fichero "Extra_b.py". ¿Eres capaz de hacer que se muestre la suma? ¿A qué crees que se debe?

3.6.3 Apartado c

Una vez completado el programa anterior, vamos a crear procesos auxiliares en lugar de hilos. Para ello, en lugar de que el segundo proceso divida la carga de la suma entre un hilo y él mismo, divida la carga con otro proceso (se recomienda usar Queue de multiprocessing para compartir datos). Se buscará de esta manera la forma de cumplir la restricción de terminar la ejecución de la suma en 3 segundos. ¿Eres capaz ahora de hacer que se muestre la suma? ¿A qué crees que se debe?