



Estatus

Computación Tolerante a Fallas

Alumno:

Código:

Profesor(a): Dr. Michel Emanuel López Franco

Sección: D06

Fecha de Entrega: 20 de septiembre de 2022

Procesos y Servicios

Esta practica tiene como objetivo implementar una aplicación que haga uso de hilos y utilizar procesos y servicios para mantenerla activa siempre, sin importar si es cerrada inesperadamente.

Un **proceso** no es más que una aplicación ejecutada por el sistema operativo en primer plano, es decir, que el usuario puede interactuar con ella y ofrece algún tipo de funcionalidad. Por otro lado, un **servicio** es algo que se ejecuta en segundo plano y normalmente el usuario no esta destinado a interactuar con este; simplemente ayuda a que se ejecuten otros procesos.

Monte un pequeño y simple programa que utiliza dos hilos para trabajar con la misma función al mismo tiempo. Dicha función obtiene la hora del sistema usando el módulo *time* y la imprime en la terminal; recibe como parámetros el nombre del hilo y un valor de *n* que equivale a la cantidad de segundos que debe esperar antes de obtener la hora. El hilo 1 recupera la hora cada 2 segundos, mientras que el hilo 2 lo hace cada 4 segundos.

```
#Imprime el tiempo de la maquina cada n segundos en 20 ocasiones.
def imprimirTiempo (name, n) → None:
    contador = 0

    while contador < 20:
        time.sleep (n)
        contador += 1

        print (f"{name}: {time.ctime (time.time ())}")
        print ("")
```

Los hilos son creados con el módulo *threading*, enviando la función como el *target* y en los argumentos los valores adecuados.

```
if __name__ == "__main__":
    try:
        t1 = threading.Thread (target=imprimirTiempo, args=("Hilo-1", 2)) #Obtiene el tiempo cada 2 segundos
        t2 = threading.Thread (target=imprimirTiempo, args=("Hilo-2", 4)) #Obtiene el tiempo cada 4 segundos

        except:
            print ("No se pudo ejecutar el hilo.")

        t1.start ()
        t2.start ()
```

Para manejar la ejecución de este programa hice otro script. Dentro de este, existe una función para saber si el programa del reloj esta ejecutándose; obtenemos una lista de los procesos que se están ejecutando con la función *process_iter* del modulo *psutil* e iteramos sobre esta comparando los nombres con los de nuestra aplicación. Si encontramos una coincidencia, continuamos la ejecución; caso contrario, usamos la función *system* para ejecutar el reloj.

```
#Funcion para verificar si el proceso esta vivo.
def isAlive () → bool:
    #Iteramos sobre todos los procesos actuales y obtenemos su nombre
    for proc in psutil.process_iter():
        if proc.name () = "reloj.exe":
            return True

    return False





if __name__ == "__main__":
    while True:
        if not isAlive ():
            system ("start reloj.exe")
```

Creamos archivos ejecutables para ambos scripts usando *pyinstaller*.

```
Raúl@LAPTOP-K05TVMDU MINGW64 ~/Desktop/UDG 2022B/Computación Tolerante a Fallas/Programas/Estat
us
$ pyinstaller -F reloj.py
```

```
Raúl@LAPTOP-K05TVMDU MINGW64 ~/Desktop/UDG 2022B/Computación Tolerante a Fallas/Programas/Estat
us
$ pyinstaller -F estatus.py
```

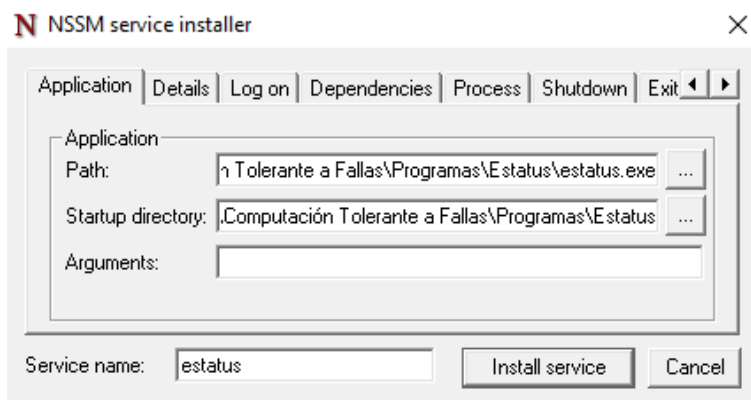
Dentro de la carpeta, aparecen de la siguiente forma:

	estatus	12/09/2022 05:47 p. m.	Application	7,756 KB
	estatus	12/09/2022 05:47 p. m.	Python File	1 KB
	reloj	12/09/2022 05:43 p. m.	Application	7,502 KB
	reloj	12/09/2022 05:41 p. m.	Python File	1 KB

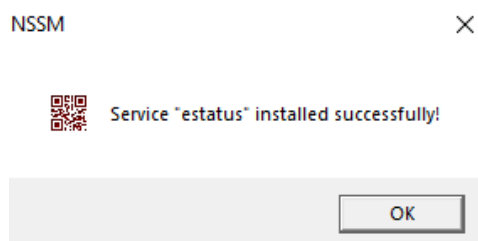
Ahora, es necesario convertir a *estatus* en un servicio para que este siempre verificando el estado de *reloj*. Esto lo hacemos usando Non-Sucking Service Manager (NSSM). Dentro de una terminal con privilegios de administrador usamos el comando *install* para crear el servicio.

```
C:\custom-services>nssm.exe install estatus
```

Nos abrirá una ventana en la que debemos poner la ruta al archivo ejecutable y la ruta donde se encuentra. También da la opción de agregar argumentos, pero como nuestro programa no lo requiere, dejamos el campo en blanco.



Una vez damos en instalar, se debería mostrar el mensaje de éxito.



Usando el comando start, iniciamos la ejecución del servicio.

```
C:\custom-services>nssm.exe start estatus
estatus: START: The operation completed successfully.
```

Dentro del administrador de tareas, podemos ver que efectivamente se está ejecutando *estatus.exe* como un servicio. A su vez, se encarga de ejecutar a *reloj.exe* como un proceso.

estatus	5800	estatus	Running	
---------	------	---------	---------	--

reloj.exe	7952	Running	SYSTEM	00	528 K	Not allowed
reloj.exe	7352	Running	SYSTEM	00	7,760 K	Not allowed

Se muestra la siguiente terminal en pantalla, en la cual cada hilo va mostrando la hora en los intervalos que se indicaron al inicio. Si cerramos la aplicación, se vuelve a abrir

automáticamente; gracias a los conocimientos adquiridos en la practica anterior, el programa puede recuperar el estado que tenía antes de cerrarse y continuar desde ahí.

```
C:\Users\raul\Desktop\UDG 2022B\Computación Tolerante a Fallas\Programas\Estatus\reloj.exe
Hilo-1: Mon Sep 12 18:22:44 2022
Hilo-1: Mon Sep 12 18:22:46 2022
Hilo-2: Mon Sep 12 18:22:46 2022

Hilo-1: Mon Sep 12 18:22:48 2022
Hilo-1: Mon Sep 12 18:22:50 2022
Hilo-2: Mon Sep 12 18:22:50 2022

Hilo-1: Mon Sep 12 18:22:52 2022
Hilo-1: Mon Sep 12 18:22:54 2022
Hilo-2: Mon Sep 12 18:22:54 2022

Hilo-1: Mon Sep 12 18:22:56 2022
Hilo-1: Mon Sep 12 18:22:58 2022
Hilo-2: Mon Sep 12 18:22:58 2022
```

Finalmente, detengo el servicio con el comando `stop` para evitar que este constantemente consumiendo recursos del equipo.

```
C:\custom-services>nssm.exe stop estatus
estatus: STOP: The operation completed successfully.
```

Conclusiones

Esta practica fue muy desafiante para mí, ya que no comprendía del todo como es que trabajaban los hilos y donde es útil aplicarlos. Sin embargo, después de investigar por internet, logre tener una idea un poco más clara sobre lo que hacen, logrando crear el ejemplo que muestro en este reporte. Intente aplicarlo a mi aplicación con *Checkpointing*, pero no logre encontrar un lugar adecuado para incluirlos, así que mejor decidí crear una aplicación nueva y agregarle dicha función implícitamente (no utilice archivos esta vez).