Arellano Granados Angel Mariano

218123444

Computación Tolerante a Fallas

D06 2023B

Estatus

Introducción

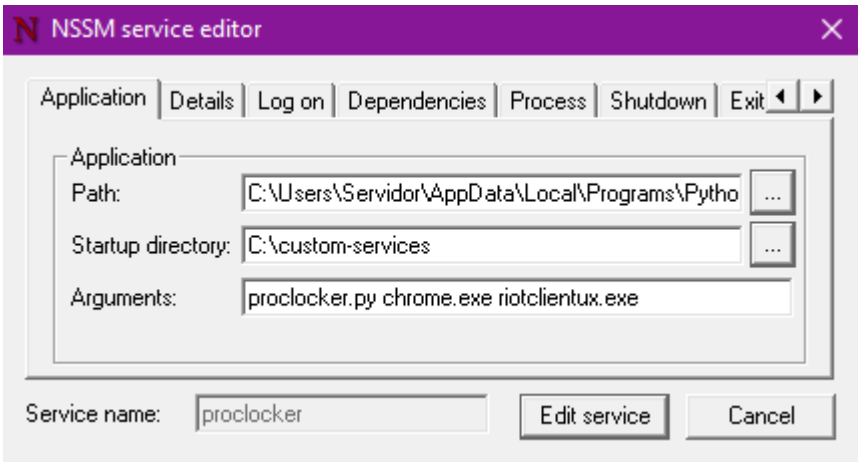
Los servicios de Windows son programas que inician y se ejecutan justo al iniciar el sistema operativo, es decir que estos no necesitan que un usuario los inicie de manera manual, existen muchos servicios que hacen una gran variedad de cosas, de hecho, en nuestras computadoras ya hay muchos que se instalan de manera predeterminada para gestionar varios aspectos del sistema operativo como por ejemplo que el pc actualice su hora cada vez que es encendida.

En esta actividad se busca que creemos un servicio que ejecute una tarea simple, pero sin la necesidad de que nosotros como usuario la ejecutemos manualmente, sino que esta este corriendo desque que encendemos nuestra máquina.

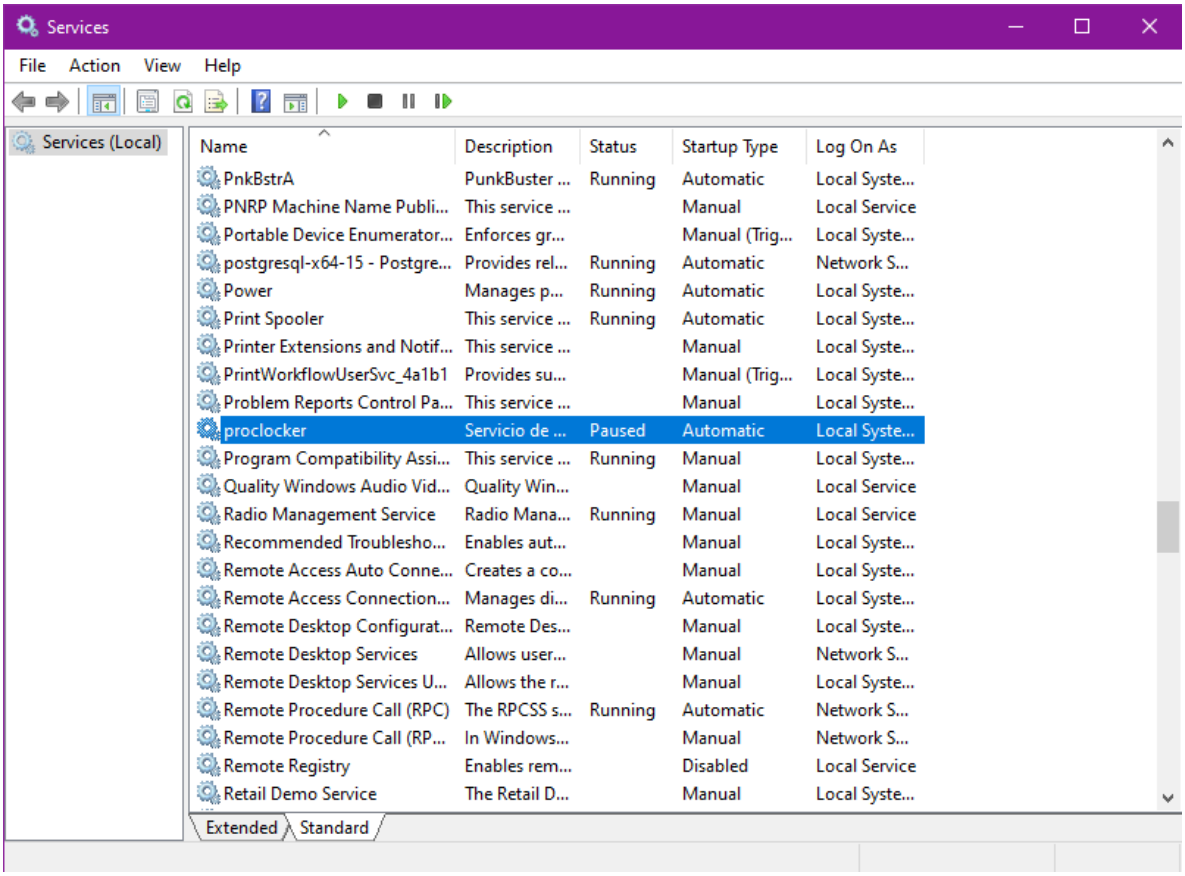
Desarrollo

Para esta actividad se contemplaron varios métodos para la creación de servicios usando el lenguaje de programación Python, sin embargo, el método que me había llamado más la atención no funcionaba por cuestiones de compatibilidad entre la librería win32serviceutil y Python en su versión 3.11.5, este ejemplo servía para abrir aplicaciones cada cierto tiempo, pero al no poder implementarlo por cuestiones de tiempo implemente el ejemplo dentro de las referencias del profesor, aun así añadí ambos códigos a la actividad.

Para la actividad se instalo el programa NSSM para crear los servicios de manera mas sencilla, dentro de este se creó el servicio procloker el cual cerraría varias aplicaciones justo después de iniciarlas, en este ejemplo fueron el Google Chrome, League of Legends y firefox.



Se crea el servicio y es visible desde el listado de servicios.



Por algún error que desconozco el servicio se pausa a si mismo y no cambia con ningún comando.

```
C:\custom-services>nssm.exe restart proclocker
proclocker: STOP: The operation completed successfully.
proclocker: Unexpected status SERVICE_PAUSED in response to START control.
```

El servicio fue creado pero no logre que arrancara y cerrara las aplicaciones al iniciarse.

Código:

Código usado en NSSM

```python
import sys
import psutil

def check_arguments():
    if len(sys.argv) == 1:
        print('Este programa no funciona sin argumentos')
        sys.exit(0)

def get_targets():

    targets = sys.argv[1:]

    i = 0
    while i < len(targets):

        if not targets[i].endswith('.exe'):
            targets[i] = targets[i] + '.exe'

        i += 1

    return targets

def lock(target):
    for proc in psutil.process_iter():
        if proc.name().lower() == target.lower():
            proc.kill()


if __name__ == '__main__':

    check_arguments()
    targets = get_targets()
```

```
    while True:
        for target in targets:
            lock(target)
```

Código descartado:

```
'''
SMWinservice
by Davide Mastromatteo

Base class to create winservice in Python
-----------------------------------------

Instructions:

1. Just create a new class that inherits from this base class
2. Define into the new class the variables
   _svc_name_ = "nameOfWinservice"
   _svc_display_name_ = "name of the Winservice that will be displayed in
scm"
   _svc_description_ = "description of the Winservice that will be displayed
in scm"
3. Override the three main methods:
    def start(self) : if you need to do something at the service
initialization.
                      A good idea is to put here the inizialization of the
running condition
    def stop(self)  : if you need to do something just before the service is
stopped.
                      A good idea is to put here the invalidation of the
running condition
    def main(self)  : your actual run loop. Just create a loop based on your
running condition
4. Define the entry point of your module calling the method
"parse_command_line" of the new class
5. Enjoy
'''

import socket
import win32serviceutil

import servicemanager
import win32event
import win32service
```

```python
class SMWinservice(win32serviceutil.ServiceFramework):
    '''Base class to create winservice in Python'''

    _svc_name_ = 'pythonService'
    _svc_display_name_ = 'Python Service'
    _svc_description_ = 'Python Service Description'

    @classmethod
    def parse_command_line(cls):
        '''
        ClassMethod to parse the command line
        '''
        win32serviceutil.HandleCommandLine(cls)

    def __init__(self, args):
        '''
        Constructor of the winservice
        '''
        win32serviceutil.ServiceFramework.__init__(self, args)
        self.hWaitStop = win32event.CreateEvent(None, 0, 0, None)
        socket.setdefaulttimeout(60)

    def SvcStop(self):
        '''
        Called when the service is asked to stop
        '''
        self.stop()
        self.ReportServiceStatus(win32service.SERVICE_STOP_PENDING)
        win32event.SetEvent(self.hWaitStop)

    def SvcDoRun(self):
        '''
        Called when the service is asked to start
        '''
        self.start()
        servicemanager.LogMsg(servicemanager.EVENTLOG_INFORMATION_TYPE,
                              servicemanager.PYS_SERVICE_STARTED,
                              (self._svc_name_, ''))
        self.main()

    def start(self):
        '''
        Override to add logic before the start
        eg. running condition
        '''
```

```
        pass

    def stop(self):
        '''
        Override to add logic before the stop
        eg. invalidating running condition
        '''
        pass

    def main(self):
        '''
        Main class to be ovverridden to add logic
        '''
        pass

# entry point of the module: copy and paste into the new module
# ensuring you are calling the "parse_command_line" of the new created class
if __name__ == '__main__':
    SMWinservice.parse_command_line()
```

Código creado para usar el anterior también descartado.

```
import time
import subprocess
from pathlib import Path
from SMWinservice import SMWinservice

class PythonCornerExample(SMWinservice):
    _svc_name_ = "PythonService"
    _svc_display_name_ = "Python Service Actividad"
    _svc_description_ = "Servicio de pureba 1 (calcualdora)"

    def start(self):
        self.isrunning = True

    def stop(self):
        self.isrunning = False

    def main(self):
        i = 0
        while self.isrunning:
            subprocess.Popen('C:\\Windows\\System32\\calc.exe')
            time.sleep(180)

if __name__ == '__main__':
    PythonCornerExample.parse_command_line()
```

Conclusión

Los servicios son una opción con la cual podemos monitorear otras aplicaciones y procesos para llegar a prevenir fallas dentro de estas, con ayuda de otros programas menores que cada siento tiempo o cada vez que se active un evento comprueben es estado de una aplicación para asegurar si correcto funcionamiento.

Sin embargo, tras varios intentos no logre que funcionara mi servicio creado por cuestiones de tiempo.

Referencias

- https://thepythoncorner.com/posts/2018-08-01-how-to-create-a-windows-service-in-python/
- https://tecnobillo.com/sections/python-en-windows/servicios-windows-python/servicios-windows-python.html