



Benemérita Universidad Autónoma de Puebla



Facultad de Ciencias de la Computación

Ingeniería en Ciencias de la Computación

Materia: Programación Distribuida Aplicada

Practica 1

Profesor: Gustavo Emilio Mendoza Olguin

Alumnos:

Pérez Flores Ivonne

202141158

Primavera 2025

18 de enero de 2025

Introducción

Como practica uno, se plantea la realización de un programa servidor de nombres de recursos de archivos, el cual debe de cumplir con los siguientes requisitos:

- El proceso debe de obtener la lista de archivos de una carpeta de la computadora (especificada por el usuario) y crear una lista dinámica de objetos que contenga:
 - Nombre del archivo y extensión
 - Permitir al usuario definir que archivos pueden publicarse y cuales no y el tiempo de actualización (TTL) (Guardar un archivo de configuración).
 - Crear un hilo que actualice la lista de archivos cada 5 minutos, si hay un archivo que estaba en la lista anterior, entonces debe preguntar si se podrá publicar, si existe un archivo en la lista que ya no esta en la carpeta, entonces deberá eliminarse (los mensajes de cambio de la lista deben enviarse a un log)
 - El archivo de configuración debe actualizarse cada vez que inicie el proceso con los cambios en el sistema de archivos

Asimismo, se requiere de otro hilo que inicie un socket que escuche en el puerto 50000 UDP.

Desarrollo

Para comenzar con el desarrollo de esta práctica, se consideraron varios aspectos para la implementación, los cuales se enlistan a continuación:

- Para la recopilación de los archivos desde la ruta proporcionada por el usuario, se toman en cuenta solo aquellos que tienen una extensión en el nombre del archivo, excluyendo carpetas del listado.
- El almacenamiento de un archivo de configuración se realiza mediante un archivo json para facilitar el acceso a los datos del listado.
- Los mensajes de cambio se almacenan en un archivo de extensión .log.

- Por el momento, no se implementan mas pasos mas allá de la inicialización del socket correspondiente al puerto 50000.
- El lenguaje de programación utilizado para esta práctica es Python.

Una vez considerando los pasados puntos, vamos a desglosar cada parte del código para comprender su funcionamiento.

Importación de librerías de Python

```
1  import threading
2  import time
3  import os
4  import json
5  import socket
6  import logging
```

Estas librerías cumplen con los siguientes aspectos:

- **Threading.** Librería para gestión y creación de hilos.
- **Time.** Gestión de tiempo en Python, nos permitirá suspender o poner a dormir un hilo el tiempo necesario.
- **Os.** Permite interactuar con el sistema operativo, se utilizará para las consultas de los listados de archivos en el sistema.
- **Json.** Librería para gestionar archivos json.
- **Socket.** Creación y gestión de sockets en Python.
- **Logging.** Librería para generar mensajes de registro para archivos log.

Declaración de clase Server y constructor

```
class Server:
    def __init__(self):
        self.ruta = ""
        self.contenido = []
        self.nuevos_datos = []
        self.ruta_salida = "config.json"
        #Iniciación de socket
        self.udp_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        self.host=socket.gethostname()
        self.port=50000
        # Configuración del logger
        logging.basicConfig(
            filename="historial.log",
            level=logging.INFO,
            format="%(asctime)s - %(levelname)s - %(message)s",
            datefmt="%Y-%m-%d %H:%M:%S"
        )
        self.evento_ruta_completa = threading.Event()
```

Se definen atributos como ruta, contenido, nuevos_datos, ruta_salida lo cuales ayudaran con el proceso de actualización de archivos de configuración, entre otros. De igual forma, se inicializa un socket UDP, en el puerto 50000, así como la definición de host, se define el formato para la configuración de logger para la salida de archivos .log. finalmente, se define un evento para realizar la gestión de los hilos, el que realice la lectura de la ruta y el que agregue los datos al archivo de configuración para evitar condiciones de carrera.

Método para leer ruta

```
def leerRuta(self):
    while True: # Bucle para permitir reintentos en caso de ruta incorrecta
        print("Inserte la ruta del directorio a conocer su listado:")
        self.ruta = input()
        if not os.path.isdir(self.ruta): # Verifica si la ruta es un directorio válido
            print("La ruta proporcionada no es válida. Intente nuevamente.\n")
            logging.error(f"Ruta no valida proporcionada: {self.ruta}")
        else:
            try:
                contenido_bruto = os.listdir(self.ruta)
                # Filtrar solo los archivos con extensión
                self.contenido = [archivo for archivo in contenido_bruto if '.' in archivo]
                logging.info(f"Archivos obtenidos desde la ruta: {self.ruta}")
                break
            except PermissionError:
                print("No tiene permiso para acceder a esta ruta.\n")
                logging.error(f"Permiso denegado para acceder a la ruta: {self.ruta}")
                break
            except Exception as e:
                print(f"Error inesperado al intentar acceder a la ruta: {e}\n")
                logging.error(f"Error inesperado al intentar acceder a la ruta: {e}")
                break
        self.evento_ruta_completa.set()
```

Este método nos ayudara con la gestión de la obtención de archivos desde el sistema a partir de una ruta definida por el usuario, la cual es solicitada desde teclado. De igual manera, realiza la gestión de errores, ciclando el código hasta que la ruta sea válida e informando en el archivo .log. De igual manera, acciona el evento para ejecutar el hilo que realizara los cambios al archivo de configuración.

Método para obtener contenido

```
def getContenido(self):
    contenido_bruto = os.listdir(self.ruta)
    self.contenido = [archivo for archivo in contenido_bruto if '.' in archivo]
```

Una vez que se realizó la lectura de la ruta, este método será utilizado para las consultas posteriores para la actualización del archivo de configuración.

Método para obtener datos de usuario

```
def obtenerDatosDeUsuario(self, lista_archivos):
    for archivo in lista_archivos:
        ttl = input(f"Ingrese el valor de ttl para el archivo '{archivo}': ")
        try:
            ttl = int(ttl)
        except ValueError:
            print("Valor inválido para ttl, se asignará el valor predeterminado de 3600.")
            ttl = 3600
        publish = input(f"¿Desea publicar el archivo '{archivo}'? (S/N): ").strip().lower()
        publish = True if publish == 's' or publish == 'S' else False
        self.nuevos_datos.append({
            "nombre": archivo,
            "ttl": ttl,
            "publish": publish
        })
```

Estos datos son solicitados al usuario por medio de entradas de teclado, por lo que se creyó conveniente solicitarlo por medio de un método que recorra todos los datos nuevos a los que se les asignaran estos valores. Estos datos serán enviados como parámetro desde otro método.

Método para cargar datos al archivo de configuración

```
def guardarEnJson(self):  
    self.evento_ruta_completa.wait()  
    while True:  
        self.getContenido()
```

Este método comienza a ejecutarse una vez que se leyó la ruta. Cuenta con un ciclo infinito, el cual verificara cada cierto tiempo las condiciones dadas entre el archivo de configuración y los archivos en el directorio indicado por el usuario.

```
if os.path.exists(self.ruta_salida):
```

Hace la verificación, si el archivo de configuración existe, realiza el proceso de carga de datos desde el archivo de configuración para cargar los datos guardados, en caso contrario, crea el archivo y carga los datos provenientes del fichero indicado, no sin antes preguntar por los datos.

```
else:  
    self.obtenerDatosDeUsuario(self.contenido)  
    try:  
        with open(self.ruta_salida, "w", encoding="utf-8") as archivo_json:  
            json.dump(self.nuevos_datos, archivo_json, ensure_ascii=False, indent=4)  
            logging.info(f"Archivo JSON creado con los datos iniciales: {self.nuevos_datos}")  
            print("Se guardaron nuevos archivos en la ruta especificada")  
    except Exception as e:  
        print(f"Error al crear el archivo JSON: {e}")
```

Indica las modificaciones en el archivo .log.

```
with open(self.ruta_salida, "r", encoding="utf-8") as archivo_json:  
    datos_existentes = json.load(archivo_json)  
    nombres_existentes = {item["nombre"] for item in datos_existentes}  
    datos_a_guardar = []
```

Una vez verificado que si existe el archivo de configuración, cargamos los datos del json y extraemos los elementos de clave "nombre", para la verificación posterior.

```
for archivo in self.contenido:
    if archivo not in nombres_existentes:
        datos_a_guardar.append(archivo)
```

Adjuntamos en una lista los nombres de archivos que se encuentran en el sistema, pero no en el json. Recordemos que estos elementos se deben almacenar en el archivo de configuración, por lo que serán datos para guardar.

```
datos_eliminados = [
    item["nombre"] for item in datos_existentes if item["nombre"] not in self.contenido
]
datos_existentes = [
    item for item in datos_existentes if item["nombre"] in self.contenido
]
```

Guardamos los datos que están en el archivo de configuración y no en el contenido del sistema en la lista datos_eliminados, y los datos que se encuentran tanto en el sistema como en el archivo de configuración en la lista datos_existentes.

```
if datos_a_guardar:
    self.obtenerDatosDeUsuario(datos_a_guardar)
    for nuevo_dato in self.nuevos_datos:
        if not any(dato["nombre"] == nuevo_dato["nombre"] for dato in datos_existentes):
            datos_existentes.append(nuevo_dato)
    logging.info(f"Archivos agregados: {datos_a_guardar}")
    print("Se agregaron nuevos archivos al archivo de configuracion")

if datos_eliminados:
    logging.info(f"Archivos eliminados: {datos_eliminados}")
    print("Se eliminaron archivos en el archivo de configuracion")
```

Si hay elementos en las listas anteriores, realizamos las acciones para el almacenamiento en el archivo de configuración.

```
with open(self.ruta_salida, "w", encoding="utf-8") as archivo_json:
    json.dump(datos_existentes, archivo_json, ensure_ascii=False, indent=4)
```

Guarda en el archivo de extensión json los datos.

Método para iniciar socket

```
def iniciar_socket_udp(self):
    print(f"Iniciando socket UDP en el puerto {self.port}...")
    self.udp_socket.bind(("0.0.0.0", self.port))
    while True:
        print("El socket servidor esta activo")
        time.sleep(30)
    #     data, addr = self.udp_socket.recvfrom(1024)
    #     mensaje = data.decode("utf-8")
    #     logging.info(f"Mensaje recibido desde {addr}: {mensaje}")
    #     print(f"Mensaje recibido desde {addr}: {mensaje}")
```

Esta función aun realizará una implementación de la comunicación cliente-servidor, dado que esto se realizará en una implementación posterior, por el momento solo lo dejaremos activo.

Función main

```
def main():
    server = Server()

    # Crear el hilo para leer la ruta
    hilo_leer_ruta = threading.Thread(target=server.leerRuta)
    hilo_leer_ruta.start()
    hilo_leer_ruta.join() # Asegurarse de que termine antes de iniciar el

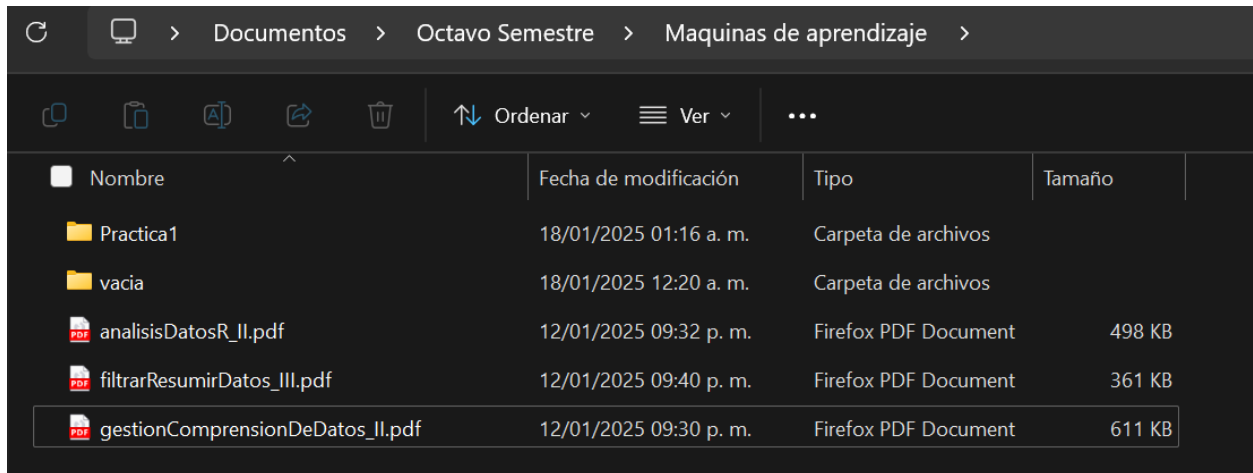
    # Crear el hilo para guardar en JSON
    hilo_guardar_json = threading.Thread(target=server.guardarEnJson)
    hilo_guardar_json.start()

    # Crear el hilo para iniciar el socket UDP
    hilo_socket_udp = threading.Thread(target=server.iniciar_socket_udp)
    hilo_socket_udp.daemon = True # Permitir que el programa termine inc
    hilo_socket_udp.start()
```

Generamos el hilo *hilo_leer_ruta*, una característica importante de este hilo es que necesita terminar su ejecución para que los demás se ejecuten. Posteriormente se inician los hilos *hilos_guardar_json* y *hilo_socket_udp*, que realizaran de manera paralela su ejecución de la revisión de archivos y la conexión del socket udp.

Ejecución

- Carga inicial de archivos



The screenshot shows a Windows File Explorer window with the address bar displaying the path: > Documentos > Octavo Semestre > Maquinas de aprendizaje >. The toolbar includes icons for copy, paste, delete, and other file operations, along with 'Ordenar' (Sort) and 'Ver' (View) dropdown menus. The main area displays a table of files and folders.

Nombre	Fecha de modificación	Tipo	Tamaño
Practica1	18/01/2025 01:16 a. m.	Carpeta de archivos	
vacía	18/01/2025 12:20 a. m.	Carpeta de archivos	
analisisDatosR_II.pdf	12/01/2025 09:32 p. m.	Firefox PDF Document	498 KB
filtrarResumirDatos_III.pdf	12/01/2025 09:40 p. m.	Firefox PDF Document	361 KB
gestionCompresionDeDatos_II.pdf	12/01/2025 09:30 p. m.	Firefox PDF Document	611 KB

```
Inserte la ruta del directorio a conocer su listado:
C:\Users\Bobe\Documents\Octavo Semestre\Maquinas de aprendizaje
Iniciando socket UDP en el puerto 50000...
El socket servidor esta activo
Ingrese el valor de ttl para el archivo ' analisisDatosR_II.pdf': 2434
¿Desea publicar el archivo ' analisisDatosR_II.pdf'? (S/N): s
Ingrese el valor de ttl para el archivo ' filtrarResumirDatos_III.pdf': 8565
¿Desea publicar el archivo ' filtrarResumirDatos_III.pdf'? (S/N): n
Ingrese el valor de ttl para el archivo ' gestionCompresionDeDatos_II.pdf': 1367
¿Desea publicar el archivo ' gestionCompresionDeDatos_II.pdf'? (S/N): s
Se guardaron nuevos archivos en la ruta especificada
```

```
[
  {
    "nombre": " analisisDatosR_II.pdf",
    "ttl": 2434,
    "publish": true
  },
  {
    "nombre": " filtrarResumirDatos_III.pdf",
    "ttl": 8565,
    "publish": false
  },
  {
    "nombre": " gestionCompresionDeDatos_II.pdf",
    "ttl": 1367,
    "publish": true
  }
]
```

```

codigos > practica1 > historial.log
1 2025-01-18 15:25:54 - INFO - Archivos obtenidos desde la ruta: C:\Users\Bobe\Documents\Octavo Semestre\
2 2025-01-18 15:26:06 - INFO - Archivo JSON creado con los datos iniciales: [{ 'nombre': ' analisisDatosR_I

```

- Carga de nuevo archivo en directorio

Documentos > Octavo Semestre > Maquinas de aprendizaje >

Nombre	Fecha de modificación	Tipo	Tamaño
Practica1	18/01/2025 01:16 a. m.	Carpeta de archivos	
vacía	18/01/2025 12:20 a. m.	Carpeta de archivos	
analisisDatosR_II.pdf	12/01/2025 09:32 p. m.	Firefox PDF Document	498 KB
filtrarResumirDatos_III.pdf	12/01/2025 09:40 p. m.	Firefox PDF Document	361 KB
gestionCompresionDeDatos_II.pdf	12/01/2025 09:30 p. m.	Firefox PDF Document	611 KB
Nuevo Documento de texto.txt	18/01/2025 03:29 p. m.	Documento de texto	0 KB

```

Ingrese el valor de ttl para el archivo 'Nuevo Documento de texto.txt': 4567
¿Desea publicar el archivo 'Nuevo Documento de texto.txt'? (S/N): s
Se agregaron nuevos archivos al archivo de configuracion
Esperando 5 minutos para la próxima actualización...

```

```

codigos > practica1 > config.json > ...
1 [
2   {
3     "nombre": " analisisDatosR_II.pdf",
4     "ttl": 2434,
5     "publish": true
6   },
7   {
8     "nombre": " filtrarResumirDatos_III.pdf",
9     "ttl": 8565,
10    "publish": false
11  },
12  {
13    "nombre": " gestionCompresionDeDatos_II.pdf",
14    "ttl": 1367,
15    "publish": true
16  },
17  {
18    "nombre": "Nuevo Documento de texto.txt",
19    "ttl": 4567,
20    "publish": true
21  }
22 ]

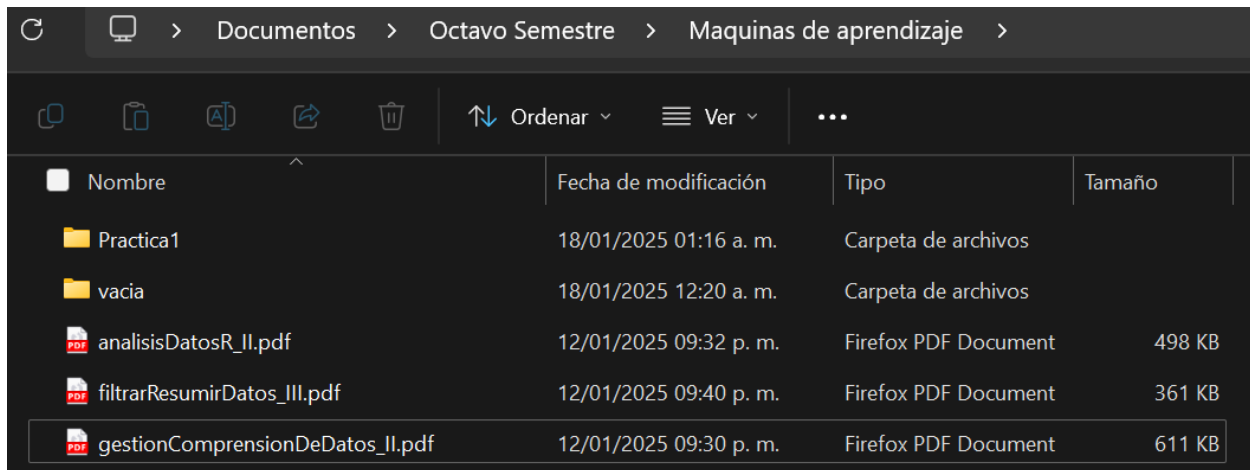
```

```

codigos > practica1 > historial.log
1 2025-01-18 15:25:54 - INFO - Archivos obtenidos desde la ruta: C:\Users\Bobe\Documents\Octavo Semestre\
2 2025-01-18 15:26:06 - INFO - Archivo JSON creado con los datos iniciales: [{ 'nombre': ' analisisDatosR_I
3 2025-01-18 15:30:10 - INFO - Archivos agregados: ['Nuevo Documento de texto.txt']

```

- Eliminación de archivo no existente en fichero



The screenshot shows a Windows File Explorer window with the address bar displaying the path: Documentos > Octavo Semestre > Maquinas de aprendizaje. The toolbar includes icons for copy, paste, delete, and other file operations, along with buttons for 'Ordenar' (Sort) and 'Ver' (View). The main area displays a table of files and folders.

Nombre	Fecha de modificación	Tipo	Tamaño
Practica1	18/01/2025 01:16 a. m.	Carpeta de archivos	
vacía	18/01/2025 12:20 a. m.	Carpeta de archivos	
analisisDatosR_II.pdf	12/01/2025 09:32 p. m.	Firefox PDF Document	498 KB
filtrarResumirDatos_III.pdf	12/01/2025 09:40 p. m.	Firefox PDF Document	361 KB
gestionComprensionDeDatos_II.pdf	12/01/2025 09:30 p. m.	Firefox PDF Document	611 KB

Se eliminaron archivos en el archivo de configuracion
Esperando 5 minutos para la próxima actualización...

```
[  
  {  
    "nombre": "analisisDatosR_II.pdf",  
    "ttl": 2434,  
    "publish": true  
  },  
  {  
    "nombre": "filtrarResumirDatos_III.pdf",  
    "ttl": 8565,  
    "publish": false  
  },  
  {  
    "nombre": "gestionComprensionDeDatos_II.pdf",  
    "ttl": 1367,  
    "publish": true  
  }  
]
```

```
codigos > practica1 > historial.log  
1 2025-01-18 15:25:54 - INFO - Archivos obtenidos desde la ruta: C:\Users\Bobe\Documents\Octavo Semestre\  
2 2025-01-18 15:26:06 - INFO - Archivo JSON creado con los datos iniciales: [{'nombre': 'analisisDatosR_I  
3 2025-01-18 15:30:10 - INFO - Archivos agregados: ['Nuevo Documento de texto.txt']  
4 2025-01-18 15:33:10 - INFO - Archivos eliminados: ['Nuevo Documento de texto.txt']  
5
```

Conclusión

Como podemos observar, la implementación de este programa se realizó con las limitaciones que esto implica, haciendo énfasis en la gestión de archivos a manera de servidor, hasta el momento se tiene una correcta implementación del servidor.