Nama : Alvin Zanua Putra

NRP : 5025231064

Kelas : Pemrograman Jaringan – D

TUGAS 3

Link Github Folder TUGAS3 :

https://github.com/alvinzanuaputra/PROGJAR24/tree/main/assets/src/TUGAS3

Ringkasan :

Menambahkan dua kemampuan baru yaitu operasi **upload file** dan **hapus file**. Pada operasi upload, isi file yang akan dikirim ke server terlebih dahulu diencode ke dalam format **Base64** untuk memastikan file dapat dikirim dalam bentuk teks melalui protokol yang ada. Sedangkan operasi hapus file berfungsi untuk menghapus file tertentu yang ada di server sesuai dengan nama file yang dikirimkan oleh client.. Perintah upload mencakup nama file dan konten file yang sudah dalam bentuk Base64, sementara perintah hapus hanya membutuhkan nama file yang akan dihapus. Selain itu, implementasi client untuk kedua operasi ini ditambahkan pada file **file_interface.py**, dan server diperbarui di **file_server.py** untuk menangani perintah baru tersebut. Pengujian dilakukan dengan menjalankan komunikasi client-server.

Jawaban :

A. Pertama kita akan membuat fungsi HAPUS di file

```
    def hapus(self, params=[]):
        try:
            filename = params[0]
            if not os.path.exists(filename):
                return dict(status='ERROR', data=f'File "{filename}" tidak ditemukan dalam direktori')
            os.remove(filename)
            return dict(status='OK', data=f'{filename} berhasil dihapus dari server')
        except Exception as e:
            return dict(status='ERROR', data=str(e))
```

Pertama-tama, fungsi memeriksa apakah file tersebut benar-benar ada di direktori dengan os.path.exists(). Jika file tidak ditemukan, maka fungsi langsung membalas dengan status ERROR dan pesan bahwa file tidak tersedia, sehingga tidak akan terjadi kesalahan lebih lanjut saat mencoba menghapus file yang tidak ada.

Jika file ditemukan, maka fungsi menggunakan os.remove() untuk menghapus file tersebut dari sistem file. Setelah berhasil dihapus, fungsi mengembalikan respons dengan status OK dan pesan bahwa file telah berhasil dihapus dari server. Sama seperti fungsi upload, fungsi hapus juga dibungkus dalam blok try-except untuk menangani kemungkinan kesalahan seperti error akses file

B. Selanjutnya membuat fungsi UPLOAD file ke server :

```
    def upload(self, params=[]):
        try:
            filename, filecontent = params[0], params[1]
            if os.path.exists(filename):
                return dict(status='ERROR', data=f'File "{filename}" sudah tersedia dalam direktori')
            with open(filename, 'wb') as fp:
                fp.write(base64.b64decode(filecontent.encode()))
            return dict(status='OK', data=f'{filename} berhasil diupload ke server')
        except Exception as e:
            return dict(status='ERROR', data=str(e))
```

Menerima parameter berupa nama file (filename) dan isi file dalam bentuk string yang sudah di-*encode* ke format Base64 (filecontent). Sebelum menyimpan file, fungsi melakukan pengecekan menggunakan os.path.exists() untuk memastikan fiel nama itu ada di direktori. Jika file ada, mengembalikan respons error dengan pesan file sudah tersedia dan tidak akan menimpa file lama.

Jika file belum ada, isi file akan didekode dari format Base64 menjadi biner menggunakan base64.b64decode(), kemudian disimpan ke dalam file dengan nama yang diberikan. File dibuka dalam mode wb (write binary) agar bisa menampung data biner hasil decoding. Jika semua proses berjalan tanpa kesalahan, fungsi akan mengembalikan status OK. Namun jika terjadi kesalahan decoding gagal atau masalah lain, maka pengecualian akan ditangkap dengan status ERROR

C. Dokumen update protocol bertambah fungsi yaitu hapus dan upload beserta validasi nya

FILE SERVER
TUJUAN: melayani client dalam request file server

ATURAN PROTOKOL:
- client harus mengirimkan request dalam bentuk string
- string harus dalam format:
REQUEST spasi PARAMETER1 spasi PARAMETER2 dst.
- Semua result akan diberikan dalam bentuk JSON
- Diakhiri dengan karakter ASCII "\r\n\r\n"

REQUEST YANG DILAYANI:

LIST
** TUJUAN: Mendapatkan daftar seluruh file yang tersedia di direktori server*
** PARAMETER: tidak ada*
** RESULT:*
- BERHASIL:
- status: OK
- data: list file
- GAGAL:
- status: ERROR
- data: pesan kesalahan

GET
** TUJUAN: Mengambil isi file dengan nama tertentu*
** PARAMETER:*
- PARAMETER1: nama file
** RESULT:*
- BERHASIL:
- status: OK
- data_namafile: nama file
- data_file: isi file (dalam base64)
- GAGAL:
- status: ERROR
- data: pesan kesalahan

UPLOAD
** TUJUAN: Mengunggah file ke server*
** PARAMETER:*
- PARAMETER1: nama file yang akan disimpan

- *PARAMETER2: isi file dalam bentuk base64*

** VALIDASI:*
- *Jika file sudah ada di server, maka upload akan ditolak*

** RESULT:*
- *BERHASIL:*
  - *status: OK*
  - *data: nama file berhasil diupload ke server*
- *GAGAL:*
  - *status: ERROR*
  - *data: file sudah ada atau kesalahan lain*

HAPUS
** TUJUAN: Menghapus file dari server*
** PARAMETER:*
- *PARAMETER1: nama file yang akan dihapus*

** VALIDASI:*
- *Jika file tidak ada, maka penghapusan akan ditolak*

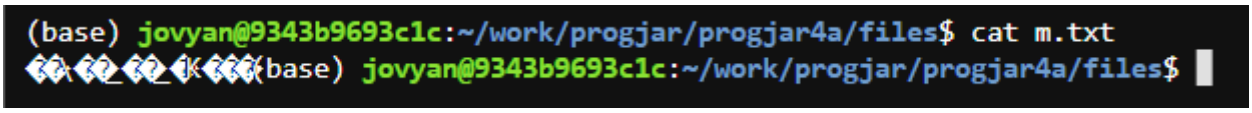
** RESULT:*
- *BERHASIL:*
  - *status: OK*
  - *data: nama file berhasil dihapus dari server*
- *GAGAL:*
  - *status: ERROR*
  - *data: file tidak ditemukan atau kesalahan lain*

INFORMASI TAMBAHAN:
- Jika request tidak dikenali:
- *status: ERROR*
- *data: request tidak dikenali*

.

D. Setelah mencoba testing ternyata ada kesalahan dalam menerima file yang isinya corrupt dikarenakan pada file_protocol.py ada fungsi yang menyebabkan string data dalam file corrupt yaitu string_data_masuk.lower yang menyebabkan semua huruf hasil encode diformat kecil semua dan tidak dapat diproses ,

Output saat error memasukkan data file karena string data_namafile dipaksa huruf kecil semua




```
logging.warning(f"string diproses: {string_datamasuk}")
        c = shlex.split(string_datamasuk)
```

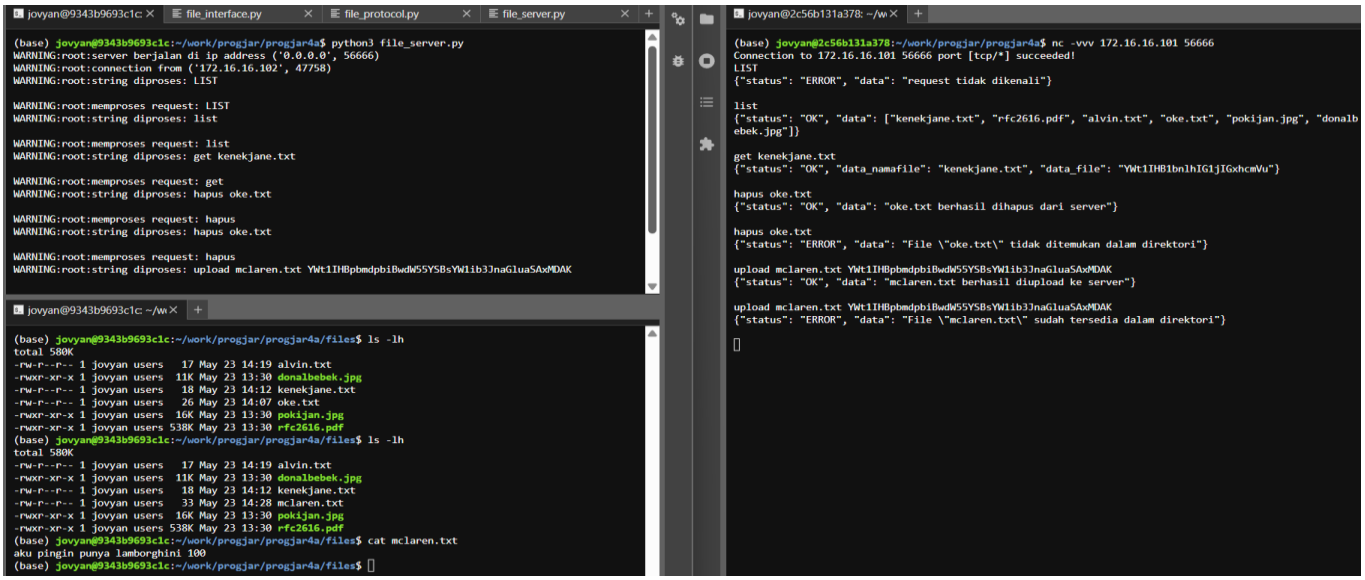
Akan tetapi karena string_datamasuk dihapus dalam pemanggilan command LIST, GET, HAPUS, UPLOAD harus di format dalam huruf kecil semua karena keterbalikan antara uppercase dan lower sehingga contoh pemanggilanya yaitu list, get namafile, hapus namafile, upload namefile isifile encodebase64.

Ada bug ke 2 dimana kalau kita upload encodedbase64 dengan string terlalu Panjang server tidak dapat menerima pada file_server.py pada fungsi def run (self) pada self.connection.recv atau maknanya nya koneksi data server yang diterima

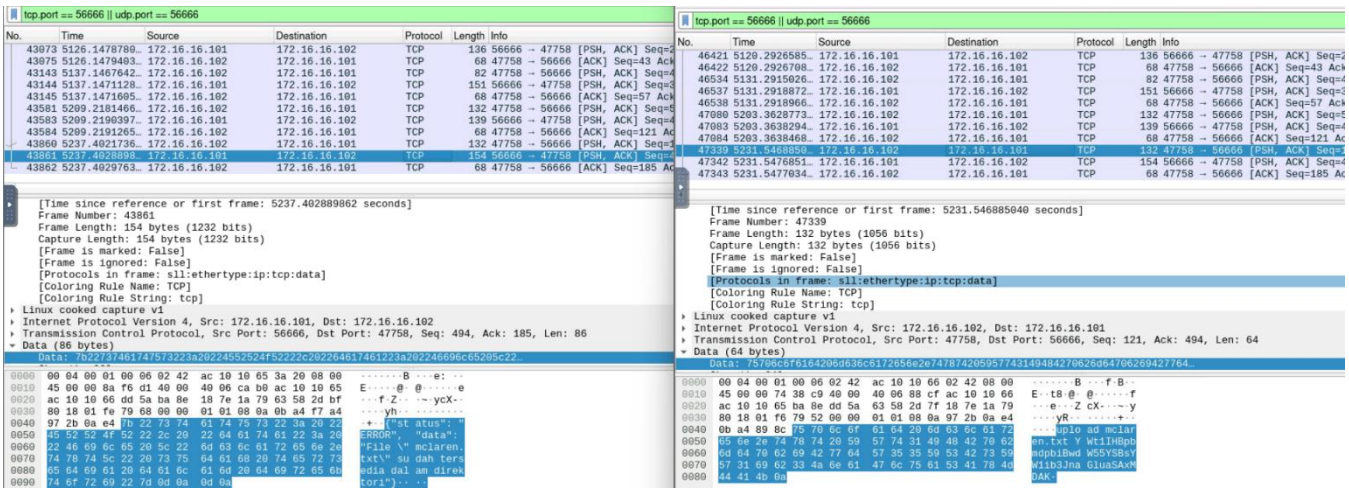
Contoh output error string terlalu Panjang dalam isi file :

```python
def run(self):
        while True:
            data = self.connection.recv(8096)
            if data:
                d = data.decode()
                hasil = fp.proses_string(d)
                hasil=hasil+"\r\n\r\n"
                self.connection.sendall(hasil.encode())
            else:
                break
        self.connection.close()
```

oleh karena itu receive server harus diubah size nya yang awalnya 32 menjadi 4096 – 8096 tergantung kebutuhan.



Server berhasil dijalankan di alamat 0.0.0.0 dan port 56666, serta menerima koneksi dari client dengan IP 172.16.16.102. Server mencatat semua perintah yang masuk dengan format log Python. Client berkomunikasi menggunakan nc (netcat) untuk mengirimkan perintah seperti list, get, hapus, dan upload. Saat client mengirim perintah LIST (huruf kapital), server merespon dengan "request tidak dikenali", menunjukkan bahwa saat string_datafile dikirim format lower sudah dihapus . Setelah perintah list, server berhasil mengembalikan daftar file yang ada. Selanjutnya, client mencoba mengambil file kenekjane.txt dengan perintah get kenekjane.txt, dan server berhasil mengirimkan isi file dalam format base64. Ketika file oke.txt dihapus menggunakan perintah hapus oke.txt, server merespons sukses. Namun, saat perintah yang sama dikirim lagi, server menolak karena file tersebut sudah tidak ada, dan mengembalikan pesan error yang sesuai. Hal serupa juga terjadi pada perintah upload: ketika client mengunggah file mclaren.txt untuk pertama kalinya, server berhasil menyimpannya. Tetapi saat perintah yang sama diulang dengan isi file yang sama, server menolak dengan pesan bahwa file sudah tersedia.



Analisis wireshark :

Hasil tangkapan paket menggunakan Wireshark yang memperlihatkan komunikasi TCP antara client (172.16.16.102) dan server (172.16.16.101) pada port 56666. Pada sisi kanan gambar, terlihat bahwa client mengirimkan request upload mclaren.txt beserta isi file dalam format base64. Data ini dikirim dari port client 47758 menuju port server 56666 dan terlihat dengan jelas dalam bentuk payload string di bagian bawah (upload mclaren.txt <data base64>). Di sisi kiri gambar, server memberikan respon dengan pesan JSON yang menunjukkan bahwa file mclaren.txt sudah ada: "status":"ERROR", "data":"File \"mclaren.txt\" sudah tersedia dalam direktori". Respon ini memperlihatkan bahwa validasi pada fungsi upload berjalan dengan baik, dan protokol aplikasi berhasil menangani kondisi di mana file sudah pernah diunggah sebelumnya. Paket TCP tersebut mengandung payload berformat string JSON, yang merupakan bagian dari protokol aplikasi berbasis teks yang dikembangkan. Proses Push (PSH, ACK) menunjukkan bahwa data dikirim langsung tanpa menunggu buffer penuh.