

Nama : Christoforus Indra Bagus Pratama
NRP : 5025231124
Kelas : Pemrograman Jaringan – D
Link Github : <https://github.com/itozt/tugas3Progjar/tree/main>

TUGAS 3

Perintah Pengerjaan

1. Menambahkan fungsi operasi untuk menghapus file dan mengupload file (dengan isi file yang sudah diencode base 64) pada file file_interface.py
2. Mengupdate spesifikasi protokol pada file PROTOKOL.py
3. Melakukan client implementation dari operasi tambahan tersebut

Nomor 1

Menambahkan fungsi operasi untuk menghapus file dan mengupload file (dengan isi file yang sudah diencode base 64) pada file file_interface.py

- ❖ Link Github : https://github.com/itozt/tugas3Progjar/blob/main/file_interface.py
- ❖ Fungsi “delete”

```
def delete(self, params=[]):  
    try:  
        filename = params[0]  
        if os.path.exists(filename):  
            os.remove(filename)  
            return dict(status='OK', data=f'{filename} berhasil dihapus')  
        else:  
            return dict(status='ERROR', data='File not found')  
    except Exception as e:  
        return dict(status='ERROR', data=str(e))
```

Penjelasan cara kerja :

1. params[0] : Mengambil nama file dari daftar parameter.
2. os.path.exists(filename) : Memeriksa apakah file ada di direktori kerja (files/).
3. os.remove(filename) : Jika ada, hapus file dari sistem.
4. Pengembalian nilai :
 - Jika berhasil, mengembalikan status='OK' dan pesan konfirmasi.
 - Jika file tidak ditemukan, status='ERROR' dengan data='File not found'.
5. except Exception as e : Menangkap kesalahan lain (misalnya izin akses) dan merespons dengan status='ERROR' plus detail pengecualian.

- ❖ Fungsi “upload”

```
def upload(self, params=[]):  
    try:  
        filename = params[0]  
        if os.path.exists(filename):  
            return dict(status='OK', data=f'{filename} file sudah ada')  
        filedata_64 = " ".join(params[1:])  
        filedata = base64.b64decode(filedata_64.encode())  
  
        with open(filename, 'w') as f:  
            f.write(filedata)  
  
        return dict(status='OK', data=f'File {filename} berhasil upload')  
    except Exception as e:  
        return dict(status='ERROR', data=str(e))
```

Penjelasan cara kerja :

1. `params[0]` : Mengambil nama file yang ingin diunggah.
2. Cek keberadaan : Jika file sudah ada, langsung kembalikan status='OK' tanpa menimpa.
3. Menggabungkan data Base64 : `params[1:]` berisi fragmen string Base64 (karena pemisahan oleh spasi), lalu digabung dengan `" ".join()`.
4. Dekode Base64 : `base64.b64decode()` mengubah string Base64 menjadi bytes asli.
5. Membuka dan menulis file : `with open(filename, 'w')` membuka file dalam mode teks (idealnya seharusnya 'wb' untuk binary), lalu `f.write(filedata)` menulis konten hasil decode.
6. Pengembalian nilai :
 - Jika berhasil, kembalikan status='OK' dan pesan sukses.
 - Jika terjadi kesalahan (misalnya Base64 tidak valid atau izin tulis), `except` menangkapnya dan mengembalikan status='ERROR' dengan detail pengecualian.

Nomor 2

Mengupdate spesifikasi protokol pada file PROTOKOL.py

- ❖ Link Github : <https://github.com/itozt/tugas3Progar/blob/main/PROTOKOL.txt>
- ❖ Berikut tambahan spesifikasi yang ditambahkan

DELETE

* TUJUAN: untuk menghapus file yang tersedia

* PARAMETER: nama file

* RESULT:

- BERHASIL:

- status: OK

- data: nama file yang dihapus

- GAGAL:

- status: ERROR

- data: pesan kesalahan (File not found)

* PENJELASAN TAMBAHAN:

Fungsi delete pada FileInterface bekerja dengan mengambil nama file dari parameter pertama (`params[0]`), lalu memeriksa keberadaan file tersebut di direktori kerja menggunakan `os.path.exists()`. Jika file ditemukan, metode `os.remove()` dipanggil untuk menghapus file, dan fungsi mengembalikan kamus (dict) dengan status='OK' serta pesan konfirmasi bahwa file berhasil dihapus. Apabila file tidak ada, fungsi akan mengembalikan status='ERROR' dengan keterangan "File not found". Seluruh proses dilindungi blok `try...except` untuk menangkap dan melaporkan kesalahan lain (misalnya perizinan) sebagai status='ERROR' beserta detail pengecualian.

UPLOAD

* TUJUAN: mengupload file dengan membuat file baru dan mengirimkan isi file tersebut dalam encode base64

* PARAMETER: nama file dan isi file dalam encode base64

* RESULT:

- BERHASIL:

- status: OK

- data: nama file yang dibentuk/diupload

- GAGAL:

- status: ERROR

- data: pesan kesalahan

* PENJELASAN TAMBAHAN:

Fungsi upload dimulai dengan membaca nama file yang ingin diunggah dari `params[0]` dan langsung memeriksa apakah file tersebut sudah ada. Jika sudah ada, fungsi segera mengembalikan status='OK' tanpa menimpa file lama. Jika belum, sisa parameter (`params[1:]`) dianggap sebagai potongan string Base64, digabung kembali menjadi satu string dengan `" ".join()`, lalu didekode menjadi bytes menggunakan `base64.b64decode()`. Hasil decode kemudian ditulis ke disk dengan membuka file dalam mode tulis (`open(filename, 'w')`) dan memanggil `f.write(filedata)`. Jika semua langkah berhasil, fungsi mengembalikan status='OK' dan pesan sukses; jika terjadi kesalahan (misalnya data Base64 tidak valid atau masalah izin), blok `except` menangkap pengecualian dan mengembalikan status='ERROR' beserta penjelasan kesalahan.

Nomor 3

Melakukan client implementation dari operasi tambahan tersebut

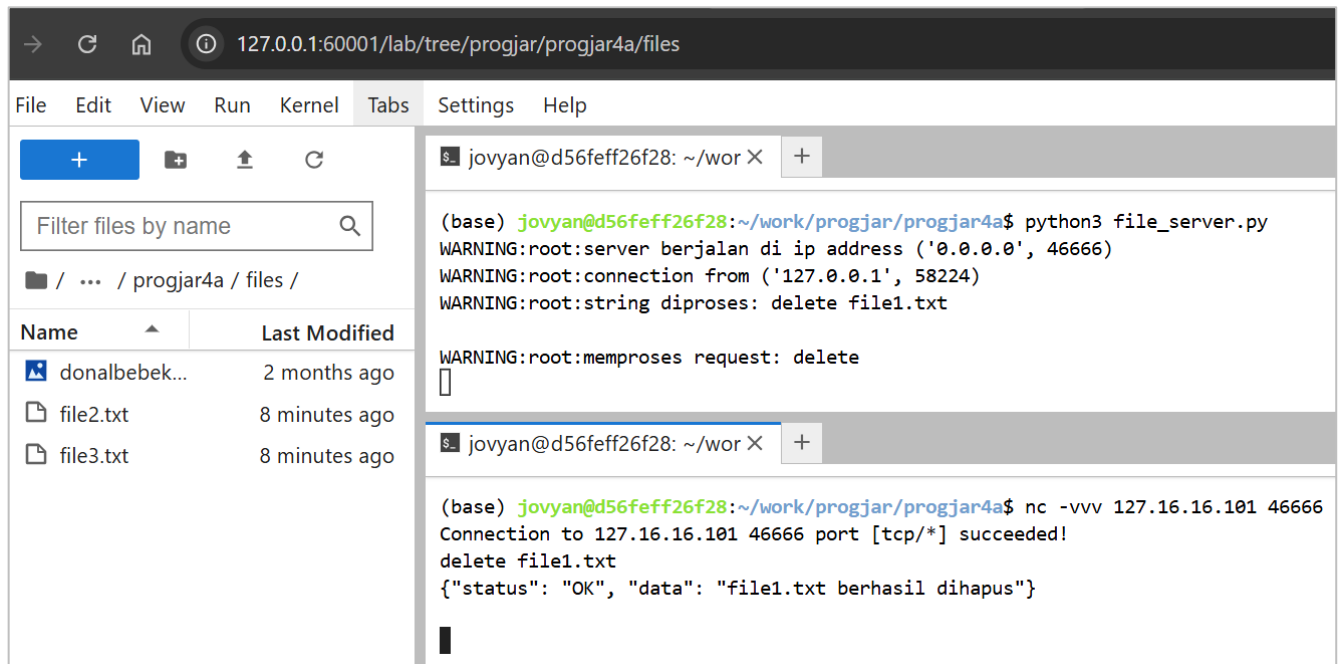
❖ DELETE

Tujuan : Menghapus file dengan nama file1.txt

Cara : Client memberikan command `delete [nama file yang ingin dihapus]`

Hasil : File file1.txt berhasil dihapus

Dokumentasi :



```
→ 127.0.0.1:60001/lab/tree/progjar/progjar4a/files
File Edit View Run Kernel Tabs Settings Help

Filter files by name
/ ... / progjar4a / files /
Name Last Modified
donalbebek... 2 months ago
file2.txt 8 minutes ago
file3.txt 8 minutes ago

jovyan@d56feff26f28: ~/wor X
(base) jovyan@d56feff26f28:~/work/progjar/progjar4a$ python3 file_server.py
WARNING:root:server berjalan di ip address ('0.0.0.0', 46666)
WARNING:root:connection from ('127.0.0.1', 58224)
WARNING:root:string diproses: delete file1.txt

WARNING:root:memproses request: delete
[]

jovyan@d56feff26f28: ~/wor X
(base) jovyan@d56feff26f28:~/work/progjar/progjar4a$ nc -vvv 127.16.16.101 46666
Connection to 127.16.16.101 46666 port [tcp/*] succeeded!
delete file1.txt
{"status": "OK", "data": "file1.txt berhasil dihapus"}
```

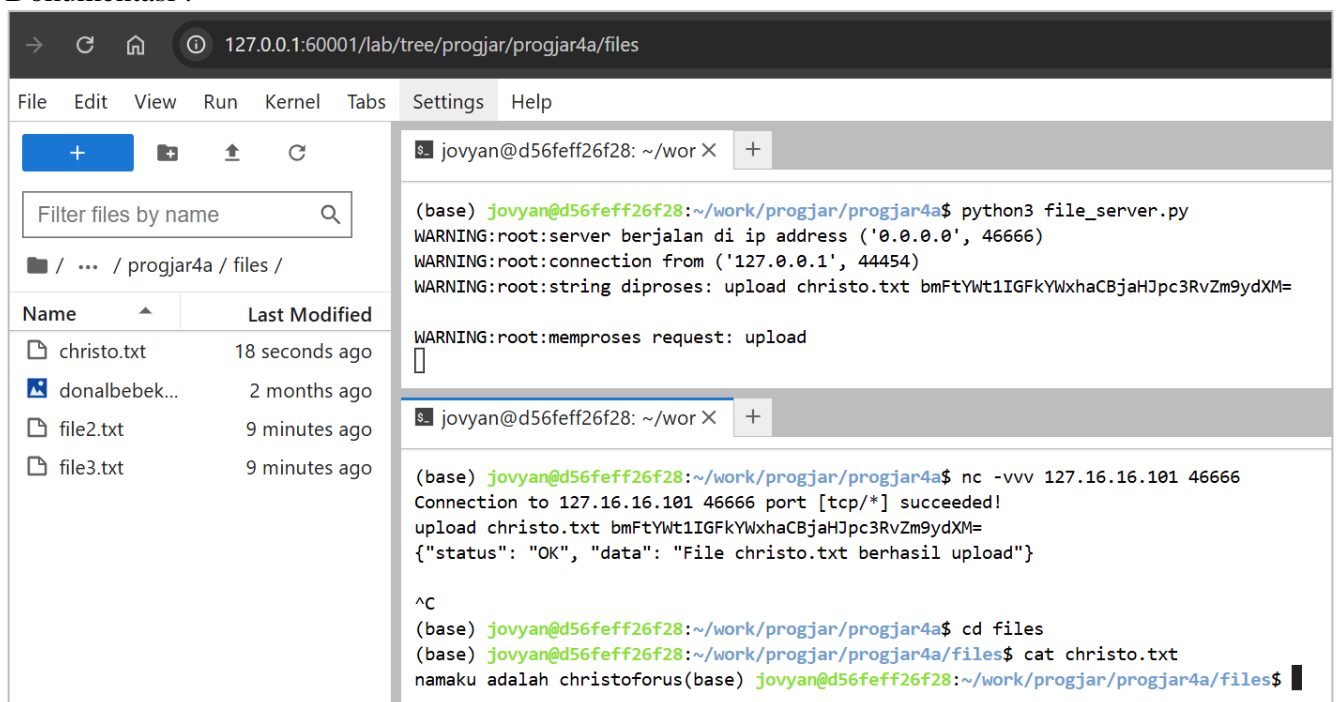
❖ UPLOAD

Tujuan : Mengupload file dengan nama christo.txt dengan isi `namaku adalah christoforus`

Cara : Client melakukan encode string `namaku adalah christoforus` terlebih dahulu sehingga menghasilkan string base64. Kemudian, client memberikan command `upload [nama file yang diupload] [isi file dalam encode base64]`.

Hasil : File christo.txt berhasil terbentuk dan isinya berhasil didecode dan terbaca sebagai `namaku adalah christoforus`.

Dokumentasi :



```
→ 127.0.0.1:60001/lab/tree/progjar/progjar4a/files
File Edit View Run Kernel Tabs Settings Help

Filter files by name
/ ... / progjar4a / files /
Name Last Modified
christo.txt 18 seconds ago
donalbebek... 2 months ago
file2.txt 9 minutes ago
file3.txt 9 minutes ago

jovyan@d56feff26f28: ~/wor X
(base) jovyan@d56feff26f28:~/work/progjar/progjar4a$ python3 file_server.py
WARNING:root:server berjalan di ip address ('0.0.0.0', 46666)
WARNING:root:connection from ('127.0.0.1', 44454)
WARNING:root:string diproses: upload christo.txt bmFtYWt1IGFkYWxhaCBjaHJpc3RvZm9ydXM=

WARNING:root:memproses request: upload
[]

jovyan@d56feff26f28: ~/wor X
(base) jovyan@d56feff26f28:~/work/progjar/progjar4a$ nc -vvv 127.16.16.101 46666
Connection to 127.16.16.101 46666 port [tcp/*] succeeded!
upload christo.txt bmFtYWt1IGFkYWxhaCBjaHJpc3RvZm9ydXM=
{"status": "OK", "data": "File christo.txt berhasil upload"}

^C
(base) jovyan@d56feff26f28:~/work/progjar/progjar4a$ cd files
(base) jovyan@d56feff26f28:~/work/progjar/progjar4a/files$ cat christo.txt
namaku adalah christoforus(base) jovyan@d56feff26f28:~/work/progjar/progjar4a/files$
```

Pengerjaan Tambahan

Sebelum melakukan client implementation di atas, perlu untuk melakukan beberapa perubahan di file `file_server.py` dan `file_protocol.py` sebagai berikut :

❖ Perubahan di `file_server.py`

Link Github : https://github.com/itozt/tugas3Progjar/blob/main/file_server.py

```
def run(self):
    """Method yang dijalankan saat thread dimulai"""
    while True:
        # Menerima data dari client (maksimal 32 bytes)
        data = self.connection.recv(4096)
        if data:
            # Decode data yang diterima
            d = data.decode()
            # Proses string menggunakan protokol file
            hasil = fp.proses_string(d)

def main():
    """Fungsi utama untuk menjalankan server"""
    # Buat instance server dengan IP 0.0.0.0 dan port 46666
    svr = Server(ipaddress='0.0.0.0',port=46666)
    # Mulai server
    svr.start()
```

Perubahan pada `self.connection.recv(4096)` :

Perintah klien (misalnya `UPLOAD nama_file sangat_panjang_base64...`) bisa jauh melebihi 32 byte. Jika buffer terlalu kecil, data akan terpotong atau terbagi ke banyak panggilan `recv()`, membuat logika penggabungan dan parsing jadi lebih rumit. Dengan 4096 byte, biasanya seluruh perintah atau sebagian besar payload kecil akan diterima dalam satu panggilan. Angka 4096 (4 KB) dipilih karena sejalan dengan ukuran halaman (page) di banyak sistem operasi dan ukuran buffer jaringan standar. Membaca 4 KB sekali panggil memberikan efisiensi lebih tinggi daripada hanya membaca 32 byte, sehingga mengurangi jumlah system call dan perpindahan konteks yang membebani kinerja.

Perubahan menjadi `port=46666` :

Kita melakukan komunikasi dengan menggunakan port 46666.

❖ Perubahan di `file_protocol.py`

Link Github : https://github.com/itozt/tugas3Progjar/blob/main/file_protocol.py

```
def __init__(self):
    self.file = FileInterface()

def proses_string(self,string_datamasuk=''):
    logging.warning(f"string diproses: {string_datamasuk}")
    c = shlex.split(string_datamasuk)
```

Perubahan `string_datamasuk.lower()` menjadi `string_datamasuk` saja :

Nama file sering kali sensitif huruf (misalnya `MyFile.JPG`). Jika seluruh string dipaksa menjadi huruf kecil, nama file bisa berubah menjadi `myfile.jpg`, yang mungkin tidak ada di sistem berkas—terutama di filesystem Linux yang sensitif huruf. Begitu juga dengan isi/konten dari suatu file setelah dilakukan encode base64 akan terdapat banyak karakter upper dan lower yang sensitive. Maka, kita perlu mempertahankan huruf besar-kecil pada parameter isi/konten file pada fungsi operasi `UPLOAD`.