

Tugas 3 Tugas 3

- Pada file server protocol
(<https://github.com/rm77/progjar/tree/master/progjar4a>),
- Tambahkan kemampuan
 - Upload file
 - Content file yang diupload harus diencode dulu dengan format base64
 - Hapus file
- Update-lah spesifikasi protokol (PROTOKOL.txt) yang telah ada pada contoh dengan kemampuan yang baru ditambahkan tersebut, berikan penjelasan tambahan dalam satu paragraf
- Buatlah client implementation dari operasi tambahan tersebut. Jalankan operasi client server untuk kemampuan tersebut, berikanlah screenshot seperlunya, dan penjelasan dalam paragraf

GITHUB REPO :

NABILAH ATIKA RAHMA - 5025221005

<https://github.com/raxnabel/network-programming-5025221005>

PROTOKOL.txt

```
1 FILE SERVER
2 TUJUAN: melayani client dalam request file server
3
4 ATURAN PROTOKOL:
5 - client harus mengirimkan request dalam bentuk string
6 - string harus dalam format
7 REQUEST spasi PARAMETER
8 - PARAMETER dapat berkembang menjadi PARAMETER1 spasi PARAMETER2 dan seterusnya
9
10 REQUEST YANG DILAYANI:
11 - Informasi umum:
12 * Jika request tidak dikenali akan menghasilkan pesan
13 - status: ERROR
14 - data: request tidak dikenali
15 * Semua result akan diberikan dalam bentuk JSON dan diakhiri
16 dengan character ascii code #13#10#13#10 atau "\r\n\r\n"
17
18 LIST
19 * TUJUAN: untuk mendapatkan daftar seluruh file yang dilayani oleh file server
20 * PARAMETER: tidak ada
21 * RESULT:
22 - BERHASIL:
23 - status: OK
24 - data: list file
25 - GAGAL:
26 - status: ERROR
27 - data: pesan kesalahan
28
29 GET
30 * TUJUAN: untuk mendapatkan isi file dengan menyebutkan nama file dalam parameter
31 * PARAMETER:
32 - PARAMETER1 : nama file
33 * RESULT:
34 - BERHASIL:
35 - status: OK
36 - data_namafile : nama file yang diminta
37 - data_file : isi file yang diminta (dalam bentuk base64)
38 - GAGAL:
39 - status: ERROR
40 - data: pesan kesalahan
41
42 UPLOAD
43 * TUJUAN: untuk mengupload file dengan menyebutkan nama file dan isi file dalam parameter
44 * PARAMETER:
45 - PARAMETER1 : nama file
46 - PARAMETER2 : isi file
47 * RESULT:
48 - BERHASIL:
49 - status: OK
50 - data : UPLOAD (data_namafile) Success
51 - GAGAL:
52 - status: ERROR
53 - data: pesan kesalahan
54
55 DELETE
56 * TUJUAN: untuk menghapus file dengan menyebutkan nama file dalam parameter
57 * PARAMETER:
58 - PARAMETER1 : nama file
59 * RESULT:
60 - BERHASIL:
61 - status: OK
62 - data : DELETE (data_namafile) Success
63 - GAGAL:
64 - status: ERROR
65 - data: pesan kesalahan
```

file_client_cli.py

```
1 import socket
2 import json
3 import base64
4 import logging
5
6 server_address = ('172.16.16.101', 7777)
7
8 def send_command(command_str=""):
9     global server_address
10    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
11    sock.connect(server_address)
12    logging.warning(f"connecting to {server_address}")
13    try:
14        logging.warning(f"sending message")
15        sock.sendall(command_str.encode())
16        data_received = ""
17        while True:
18            data = sock.recv(1024)
19            if data:
20                data_received += data.decode()
21                if "\n\n\r\n" in data_received:
22                    break
23            else:
24                break
25        hasil = json.loads(data_received)
26        logging.warning("data received from server:")
27        return hasil
28    except:
29        logging.warning("error during data receiving")
30        return False
31
32 def remote_list():
33     command_str = "LIST"
34     hasil = send_command(command_str)
35     if hasil['status'] == 'OK':
36         print("Daftar file:")
37         for mfile in hasil['data']:
38             print(f"- {mfile}")
39         return True
40     else:
41         print("Gagal")
42         return False
43
44 def remote_get(filename=""):
45     command_str = f"GET {filename}"
46     hasil = send_command(command_str)
47     if hasil['status'] == 'OK':
48         namafile = hasil['data_namafile']
49         isifile = base64.b64decode(hasil['data_file'])
50         with open(namafile, 'wb') as fp:
51             fp.write(isifile)
52         return True
53     else:
54         print("Gagal")
55         return False
56
57 def remote_upload(filename=""):
58     try:
59         with open(filename, 'rb') as fp:
60             isifile = base64.b64encode(fp.read()).decode()
61             command_str = f"UPLOAD {filename} {isifile}"
62             hasil = send_command(command_str)
63             if hasil['status'] == 'OK':
64                 print(hasil['data'])
65                 return True
66             else:
67                 print("Gagal")
68                 return False
69     except FileNotFoundError:
70         print(f"File {filename} tidak ditemukan")
71         return False
72
73 def remote_delete(filename=""):
74     command_str = f"DELETE {filename}"
75     hasil = send_command(command_str)
76     if hasil['status'] == 'OK':
77         print(hasil['data'])
78         return True
79     else:
80         print("Gagal")
81         return False
82
83 if __name__ == '__main__':
84     server_address = ('172.16.16.101', 7777)
85     print("==== Tes LIST ===")
86     remote_list()
87
88     print("\n==== Tes GET ===")
89     remote_get('donalbebek.jpg')
90
91     print("\n==== Tes LIST setelah GET ===")
92     remote_list()
93
94     print("\n==== Tes UPLOAD ===")
95     remote_upload('tugas3.txt')
96
97     print("\n==== Tes LIST setelah UPLOAD ===")
98     remote_list()
99
100    print("\n==== Tes DELETE ===")
101    remote_delete('tugas3.txt')
102
103    print("\n==== Tes LIST setelah DELETE ===")
104    remote_list()
```

file_interface.py

```
1  import os
2  import json
3  import base64
4  from glob import glob
5
6  class FileInterface:
7      def __init__(self):
8          os.chdir('files/')
9
10     def list(self, params=[]):
11         try:
12             filelist = glob('*.*)')
13             return dict(status='OK', data=filelist)
14         except Exception as e:
15             return dict(status='ERROR', data=str(e))
16
17     def get(self, params=[]):
18         try:
19             filename = params[0]
20             if not filename:
21                 return None
22             with open(filename, 'rb') as fp:
23                 isifile = base64.b64encode(fp.read()).decode()
24             return dict(status='OK', data_namafile=filename, data_file=isifile)
25         except Exception as e:
26             return dict(status='ERROR', data=str(e))
27
28     def upload(self, params=[]):
29         try:
30             filename = params[0]
31             file_content = base64.b64decode(params[1])
32             with open(filename, 'wb') as fp:
33                 fp.write(file_content)
34             return dict(status='OK', data=f"UPLOAD {filename} Success")
35         except Exception as e:
36             return dict(status='ERROR', data=str(e))
37
38     def delete(self, params=[]):
39         try:
40             filename = params[0]
41             if not filename:
42                 return dict(status='ERROR', data='Filename empty')
43             os.remove(filename)
44             return dict(status='OK', data=f"DELETE {filename} Success")
45         except Exception as e:
46             return dict(status='ERROR', data=str(e))
```

file_protocol.py

```
1 import json
2 import logging
3 import shlex
4
5 from file_interface import FileInterface
6
7 """
8 * class FileProtocol bertugas untuk memproses
9 data yang masuk, dan menerjemahkannya apakah sesuai dengan
10 protokol/aturan yang dibuat
11
12 * data yang masuk dari client adalah dalam bentuk bytes yang
13 pada akhirnya akan diproses dalam bentuk string
14
15 * class FileProtocol akan memproses data yang masuk dalam bentuk
16 string
17 """
18
19
20
21 class FileProtocol:
22     def __init__(self):
23         self.file = FileInterface()
24     def proses_string(self, string_datamasuk=''):
25         logging.warning(f"string diproses: {string_datamasuk}")
26         c = shlex.split(string_datamasuk)
27         if len(c) > 0:
28             c[0] = c[0].lower()
29         if len(c) > 1:
30             c[1] = c[1].lower()
31         try:
32             c_request = c[0].strip()
33             logging.warning(f"memproses request: {c_request}")
34             params = [x for x in c[1:]]
35             cl = getattr(self.file, c_request)(params)
36             return json.dumps(cl)
37         except Exception:
38             return json.dumps(dict(status='ERROR', data='request tidak dikenali'))
39
40
41 if __name__ == '__main__':
42     #contoh pemakaian
43     fp = FileProtocol()
44     print(fp.proses_string("LIST"))
45     print(fp.proses_string("GET pokijan.jpg"))
```

file_server.py

```
1  from socket import *
2  import socket
3  import threading
4  import logging
5  import time
6  import sys
7
8
9  from file_protocol import FileProtocol
10 fp = FileProtocol()
11
12
13 class ProcessTheClient(threading.Thread):
14     def __init__(self, connection, address):
15         self.connection = connection
16         self.address = address
17         threading.Thread.__init__(self)
18
19     def run(self):
20         while True:
21             data = self.connection.recv(1024)
22             if data:
23                 d = data.decode()
24                 hasil = fp.proses_string(d)
25                 hasil=hasil+"\r\n\r\n"
26                 self.connection.sendall(hasil.encode())
27             else:
28                 break
29         self.connection.close()
30
31
32 class Server(threading.Thread):
33     def __init__(self, ipaddress='0.0.0.0', port=7777):
34         self.ipinfo=(ipaddress,port)
35         self.the_clients = []
36         self.my_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
37         self.my_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
38         threading.Thread.__init__(self)
39
40     def run(self):
41         logging.warning(f"server berjalan di ip address {self.ipinfo}")
42         self.my_socket.bind(self.ipinfo)
43         self.my_socket.listen(1)
44         while True:
45             self.connection, self.client_address = self.my_socket.accept()
46             logging.warning(f"connection from {self.client_address}")
47
48             clt = ProcessTheClient(self.connection, self.client_address)
49             clt.start()
50             self.the_clients.append(clt)
51
52
53 def main():
54     svr = Server(ipaddress='0.0.0.0', port=7777)
55     svr.start()
56
57
58 if __name__ == "__main__":
59     main()
```

PROTOKOL.txt

```
1 FILE SERVER
2 TUJUAN: melayani client dalam request file server
3
4 ATURAN PROTOKOL:
5 - client harus mengirimkan request dalam bentuk string
6 - string harus dalam format
7   REQUEST spasi PARAMETER
8 - PARAMETER dapat berkembang menjadi PARAMETER1 spasi PARAMETER2 dan seterusnya
9
10 REQUEST YANG DILAYANI:
11 - Informasi umum:
12   * Jika request tidak dikenali akan menghasilkan pesan
13     - status: ERROR
14     - data: request tidak dikenali
15   * Semua result akan diberikan dalam bentuk JSON dan diakhiri
16     dengan character ascii code #11#10#13#10 atau "\r\n\r\n"
17
18 LIST
19 * TUJUAN: untuk mendapatkan daftar seluruh file yang dilayani oleh file server
20 * PARAMETER: tidak ada
21 * RESULT:
22 - BERHASIL:
23   - status: OK
24   - data: list file
25 - GAGAL:
26   - status: ERROR
27   - data: pesan kesalahan
28
29 GET
30 * TUJUAN: untuk mendapatkan isi file dengan menyebutkan nama file dalam parameter
31 * PARAMETER:
32   - PARAMETER1 : nama file
33 * RESULT:
34 - BERHASIL:
35   - status: OK
36   - data_namafile : nama file yang diminta
37   - data_file : isi file yang diminta (dalam bentuk base64)
38 - GAGAL:
39   - status: ERROR
40   - data: pesan kesalahan
41
42 UPLOAD
43 * TUJUAN: untuk mengupload file dengan menyebutkan nama file dan isi file dalam parameter
44 * PARAMETER:
45   - PARAMETER1 : nama file
46   - PARAMETER2 : isi file
47 * RESULT:
48 - BERHASIL:
49   - status: OK
50   - data : UPLOAD (data_namafile) Success
51 - GAGAL:
52   - status: ERROR
53   - data: pesan kesalahan
54
55 DELETE
56 * TUJUAN: untuk menghapus file dengan menyebutkan nama file dalam parameter
57 * PARAMETER:
58   - PARAMETER1 : nama file
59 * RESULT:
60 - BERHASIL:
61   - status: OK
62   - data : DELETE (data_namafile) Success
63 - GAGAL:
64   - status: ERROR
65   - data: pesan kesalahan
```

Pada **file_client_cli.py**, file ini berperan sebagai command-line yang memungkinkan user berinteraksi dengan server. Dengan menggunakan socket TCP, client ini mengimplementasikan 4 fungsi: LIST untuk melihat daftar file di server, GET untuk mengunduh file dalam format base64, UPLOAD untuk mengirim file ke server setelah melakukan encoding base64, dan DELETE untuk menghapus file di server. Setiap permintaan dikirim sebagai string kemudian client akan menunggu dan memproses respons dari server yang dikembalikan. Disini juga menangani error handling dan logging untuk memudahkan debugging ketika masalah koneksi atau pemrosesan data.

-> Pada file ini ditambahkan dua fungsi baru yaitu `remote_upload()` dan `remote_delete()` yang memungkinkan client mengirim perintah dan data file ke server untuk diunggah, serta menghapus file yang ada di server. Selain itu, ukuran buffer pada fungsi `send_command()` diubah dari 16 byte menjadi 1024 byte agar dapat menerima respon yang lebih besar, seperti data base64 dari file.

Pada **file_interface.py**, sebagai inti operasi file system di sisi server, file ini mengelola semua interaksi langsung dengan sistem file melalui class `FileInterface`. Class ini secara otomatis mengubah working directory ke folder 'files/' saat di inialisasi, memastikan semua operasi file terjadi di lokasi yang terkontrol. Empat method utamanya (`list`, `get`, `upload`, `delete`) masing-masing menangani operasi file system yang sesuai dengan error handling komprehensif. Untuk transfer file yang aman, class ini melakukan encoding/decoding konten file menggunakan base64 sebelum dikirim melalui jaringan. Setiap operasi selalu mengembalikan respons JSON yang konsisten berisi status operasi (OK/ERROR) dan data terkait, memudahkan proses komunikasi antara client dan server.

-> File ini dimodifikasi dengan menambahkan dua method baru dalam kelas `FileInterface`, yaitu `upload()` yang berfungsi menyimpan file dari data base64 ke file fisik di server, dan `delete()` yang menghapus file jika ada. Metode `upload()` menggunakan penamaan file dari parameter dan menuliskannya ke disk, sedangkan `delete()` akan mengecek keberadaan file sebelum menghapus.

Pada **file_protocol.py**, file ini berfungsi sebagai penerjemah antara permintaan jaringan dan operasi filesystem. Class FileProtocol menggunakan teknik refleksi (getattr) untuk memetakan string command dari client ke method yang sesuai di FileInterface. Proses parsing menggunakan shlex.split() memungkinkan penanganan parameter yang lebih robust, termasuk yang mengandung spasi. File ini juga melakukan normalisasi command menjadi lowercase untuk memastikan case-insensitive dan menambahkan logging untuk memantau setiap permintaan yang diproses. Jika command tidak valid atau tidak dikenali, sistem akan otomatis merespons dengan pesan error dalam format JSON yang konsisten, menjaga keandalan sistem secara keseluruhan.

-> Perubahan pada file ini sangat minimal, hanya menambahkan pemrosesan command menjadi lowercase dengan `command = command.lower()` agar perintah seperti "UPLOAD" atau "Delete" tetap dikenali meskipun pengguna mengetiknya dalam format huruf besar atau campuran.

Pada **file_server.py**, merupakan implementasi server multithread yang terus-menerus mendengarkan koneksi di port 7777. Setiap koneksi baru akan ditangani oleh thread terpisah (ProcessTheClient) yang bertanggung jawab penuh untuk komunikasi dengan client tersebut. Server ini menggunakan pola REUSEADDR untuk menghindari masalah port binding dan mengimplementasikan threading untuk menangani banyak client secara paralel. Setiap thread akan membaca data dari socket, meneruskannya ke File Protocol untuk diproses, kemudian mengirimkan kembali respons ke client. Desain ini memungkinkan server bekerja secara non-blocking dan scalable, meskipun dalam versi saat ini masih memiliki keterbatasan dalam hal manajemen thread dan beban tinggi.

-> Modifikasi di file ini meliputi penambahan dukungan perintah baru upload dan delete dalam blok pengecekan perintah. Selain itu, ukuran buffer pada `client_socket.recv()` ditingkatkan dari 32 byte menjadi 1024 byte untuk menerima data file yang lebih besar dari client, terutama saat proses upload.

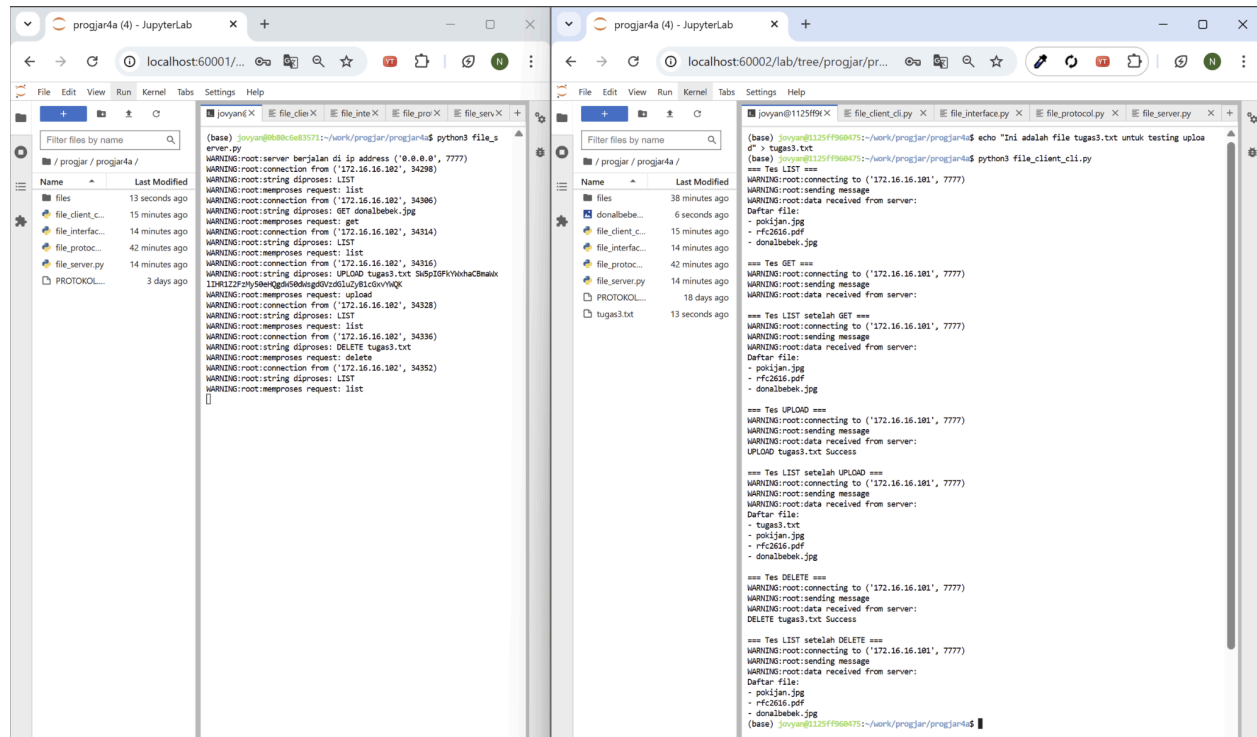
Pada PROTOKOL

File server ini dirancang untuk melayani permintaan dari client terkait pengelolaan file. Update terbaru pada pengerjaan tugas ini adalah adanya 2 permintaan UPLOAD dan juga DELETE.

Permintaan **UPLOAD** digunakan untuk mengunggah file baru ke server. Client perlu menyertakan dua parameter, yaitu nama file dan isi file dalam bentuk string. Jika berhasil, server akan merespons dengan status "OK" dan pesan bahwa file berhasil diunggah. Jika gagal, akan diberikan status "ERROR" dan keterangan kesalahannya.

Permintaan **DELETE** digunakan untuk menghapus file tertentu dari server. Client hanya perlu menyebutkan nama file sebagai parameter. Jika berhasil, server akan memberikan status "OK" disertai pesan bahwa file berhasil dihapus. Sebaliknya, jika file tidak ditemukan atau terjadi masalah lain, server akan membalas dengan status "ERROR" dan pesan kesalahan.

SS Server dan Client



The image displays two side-by-side screenshots of a JupyterLab interface, showing the development and testing of a network programming project. The left screenshot shows the file explorer on the left and the terminal output on the right. The terminal output shows the server script running, with messages indicating connections from '172.16.16.101' and successful file uploads and deletions. The right screenshot shows the file explorer on the left and the terminal output on the right. The terminal output shows the client script running, with messages indicating connections to '172.16.16.101' and successful file uploads and deletions.

Server kini memiliki empat protokol yang dapat digunakan oleh klien untuk mengajukan permintaan, yaitu LIST, GET, UPLOAD, dan DELETE. Dari keempat protokol tersebut, UPLOAD dan DELETE merupakan tambahan baru. Protokol UPLOAD digunakan untuk mengunggah file dari client ke server, sedangkan protokol DELETE berfungsi untuk menghapus file yang ada di server.

Protokol UPLOAD memerlukan dua parameter: nama file dan isi file yang telah diencode dalam format Base64. Client akan mengirimkan perintah UPLOAD diikuti nama file dan isi file yang telah dikodekan. Setelah data diterima oleh server, file didekode dari Base64 dan disimpan ke dalam sistem file server menggunakan fungsi `upload()` pada interface.

Protokol DELETE hanya membutuhkan satu parameter, yaitu nama file yang akan dihapus. Server akan mengecek keberadaan file terlebih dahulu sebelum menghapusnya menggunakan fungsi `delete()` pada interface. Protokol ini berguna untuk mengelola file yang tidak lagi dibutuhkan secara langsung dari sisi client.