Nama : Ivan Fairuz Adinata

NRP : 5025221167

Kelas : Progjar D

LAPORAN TUGAS 4 PROGJAR D

1. Link GitHub : https://github.com/mlikiwe/progjar-d-2025-tugas4.git
2. Penjelasan Modifikasi untuk tiap file
    a. File http.py

        Modifikasi pada file ini berada pada penambahan method http_post() dan http_delete() dan modifikasi pada method proses() dan method http_get() untuk penganganan list direktori files. Pada method proses(), logika ditambahkan untuk menangani method GET, POST, dan DELETE. Penanganan data yang awalnya adalah string akan diubah menjadi byte untuk menangani file gambar atau jpg agar dapat di upload secara lengkap.

        Modifikasi utama terjadi pada method POST yang diimplementasikan pada method http_post(). Setelah di modifikasi, method ini akan mampu mem-parsing request body dengan tipe multipart/form-data, yang merupakan standar untuk unggahan file dari formulir web atau klien HTTP. Logika di dalamnya secara spesifik mencari boundary dari multipart, memisahkan setiap bagian dari request, menemukan bagian yang berisi data file (content_part_bytes), dan kemudian menyimpannya ke disk dalam mode tulis biner ('wb') seperti yang telah dijelaskan sebelumnya. Untuk DELETE, method http_delete() dibuat dengan memanggil fungsi os.remove untuk melakukan penghapusan pada file yang dipilih.

    b. File server_thread_pool_http.py dan server_process_pool_http.py

        Modifikasi pada kedua file ini berfokus pada cara kedua file ini dalam menerima sebuah data dari koneksi client lalu memprosesnya. Sebelumnya, kedua file ini hanya dapat menerima dan membaca data dalam potongan yang kecil sebesar 32 byte. Hal ini tidak memungkinkan untuk mengatasi request dalam bentuk POST yang memiliki head dan body.

        Setelah modifikasi dilakukan, method processTheClient() di dalam kedua file ini dapat membaca header (Content-Length) untuk mengetahui total ukuran body yang harus diterima. Setelah menerima bagian awal data, fungsi ini masuk ke dalam sebuah loop while untuk terus menerima data dari socket (connection.recv(4096)) hingga seluruh body (sesuai Content-Length) berhasil

diterima. Seluruh data akan diteruskan ke method proses yang berada pada httpserver sebelumnya.

c. File client.py

File client sebenarnya berfungsi untuk menguji fungsionalitas dari server dalam menangani request darinya. Di dalam file client akan terdapat pengujian sederhana, meliputi request GET, POST, dan DELETE. Di sini, akan dimanfaatkan library request dari python. Dengan library ini, client hanya perlu memanggil fungsi seperti requests.get(), requests.post(), dan requests.delete().

Secara fungsional, client.py yang baru mendefinisikan tiga fungsi utama: list_files, upload_file, dan delete_file, yang masing-masing memetakan ke metode GET, POST, dan DELETE pada server. Client atau script ini akan menjalankan fungsi-fungsi di atas secara berurutan untuk menguji request satu per satu. Request akan dimulai dengan list atau melihat daftar file, lalu akan mengunggah beberapa file, lalu melakukan list untuk kedua kalinya, kemudian dilanjutkan oleh penghapusan file, dan ditutup dengan list untuk mengetahui isi directory files setelah dilakukan penghapusan.

3. Screenshot Capture Wireshark Operasi Client dan Server

a. Operasi pada threadpool

No.	Time	Source	Destination	Protocol	Length	Info
4	0.000571723	172.16.16.102	172.16.16.101	HTTP	231	GET /files/ HTTP/1.1
6	0.014991890	172.16.16.101	172.16.16.102	HTTP	530	HTTP/1.1 200 OK (text/html)
16	0.024128509	172.16.16.102	172.16.16.101	HTTP	249	POST /upload HTTP/1.1
18	0.032701080	172.16.16.101	172.16.16.102	HTTP	228	HTTP/1.1 201 Created
32	0.048146313	172.16.16.102	172.16.16.101	HTTP	1435	POST /upload HTTP/1.1
34	0.055163498	172.16.16.101	172.16.16.102	HTTP	227	HTTP/1.1 201 Created
46	0.077438079	172.16.16.102	172.16.16.101	HTTP	3705	POST /upload HTTP/1.1
48	0.083646787	172.16.16.101	172.16.16.102	HTTP	230	HTTP/1.1 201 Created
70	0.096217128	172.16.16.102	172.16.16.101	HTTP	28892	POST /upload HTTP/1.1
72	0.105362747	172.16.16.101	172.16.16.102	HTTP	235	HTTP/1.1 201 Created
80	0.110964689	172.16.16.102	172.16.16.101	HTTP	231	GET /files/ HTTP/1.1
82	0.117276110	172.16.16.101	172.16.16.102	HTTP	585	HTTP/1.1 200 OK (text/html)
90	0.120889311	172.16.16.102	172.16.16.101	HTTP	265	DELETE /files/testfile.txt HTTP/1.1
92	0.127420491	172.16.16.101	172.16.16.102	HTTP	209	HTTP/1.1 200 OK
100	0.130386852	172.16.16.102	172.16.16.101	HTTP	231	GET /files/ HTTP/1.1
102	0.136774810	172.16.16.101	172.16.16.102	HTTP	530	HTTP/1.1 200 OK (text/html)

b. Operasi pada processpool

No.	Time	Source	Destination	Protocol	Length	Info
4	0.000385957	172.16.16.102	172.16.16.101	HTTP	231	GET /files/ HTTP/1.1
6	0.094217269	172.16.16.101	172.16.16.102	HTTP	530	HTTP/1.1 200 OK (text/html)
14	0.104768038	172.16.16.102	172.16.16.101	HTTP	249	POST /upload HTTP/1.1
16	0.134024236	172.16.16.101	172.16.16.102	HTTP	228	HTTP/1.1 201 Created
29	0.145027257	172.16.16.102	172.16.16.101	HTTP	1435	POST /upload HTTP/1.1
33	0.168859117	172.16.16.101	172.16.16.102	HTTP	227	HTTP/1.1 201 Created
43	0.180504323	172.16.16.102	172.16.16.101	HTTP	3705	POST /upload HTTP/1.1
47	0.206530882	172.16.16.101	172.16.16.102	HTTP	230	HTTP/1.1 201 Created
67	0.220029242	172.16.16.102	172.16.16.101	HTTP	28892	POST /upload HTTP/1.1
70	0.244275669	172.16.16.101	172.16.16.102	HTTP	235	HTTP/1.1 201 Created
76	0.249917164	172.16.16.102	172.16.16.101	HTTP	231	GET /files/ HTTP/1.1
80	0.275024250	172.16.16.101	172.16.16.102	HTTP	585	HTTP/1.1 200 OK (text/html)
86	0.277453302	172.16.16.102	172.16.16.101	HTTP	265	DELETE /files/testfile.txt HTTP/1.1
90	0.301944022	172.16.16.101	172.16.16.102	HTTP	209	HTTP/1.1 200 OK
96	0.305452747	172.16.16.102	172.16.16.101	HTTP	231	GET /files/ HTTP/1.1
100	0.329501131	172.16.16.101	172.16.16.102	HTTP	530	HTTP/1.1 200 OK (text/html)

c. Penjelasan

Dari kedua gambar di atas, baik dengan threadpool maupun processpool, dapat diketahui bahwa keduanya telah bekerja dengan benar. Hal itu ditunjukkan bahwa seluruh response berisikan HTTP 200 OK yang menunjukkan request dari client berhasil diproses. Urutan request dan response nya juga identik, di mana alur yang terjadi adalah satu kali request GET awal, diikuti oleh empat request POST untuk mengunggah file testfile.txt dan tiga file gambar, kemudian request GET kedua untuk memverifikasi hasil unggahan, satu request DELETE untuk menghapus testfile.txt, dan diakhiri dengan request GET terakhir untuk melihat kondisi akhir direktori. Perbedaan hanya terjadi pad alokasi port dan aspek waktu/timing. Perbedaan minor pada nilai waktu tersebut adalah hal yang wajar dan mencerminkan variasi alami dalam latensi jaringan dan waktu pemrosesan oleh thread atau process yang berbeda pada saat pengujian.