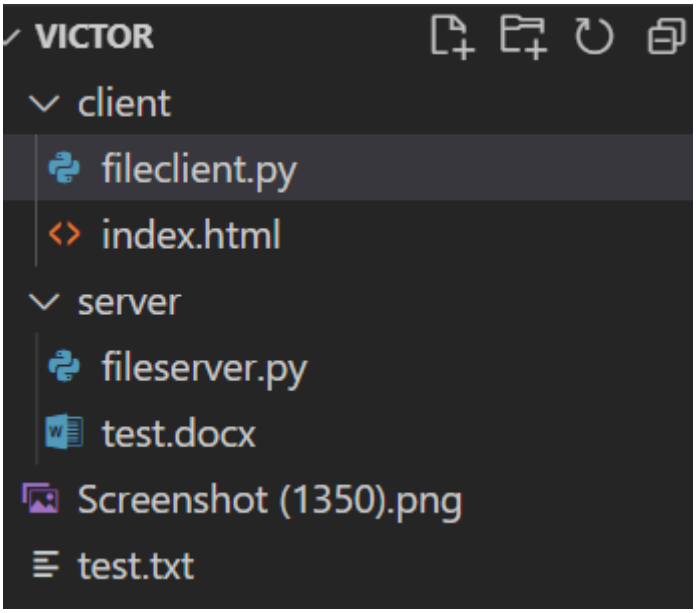
Nama : Victor Asido Tambunan
Matkul : Pemrogramman Jaringan
NPM : 51421502
Kelas : 4IA10
Tema : Socket Programming

Socket File Tranfer

Program ini merupakan implementasi dari konsep **socket programming** menggunakan bahasa Python, yang bertujuan untuk membangun aplikasi **client-server** yang mampu melakukan **pengiriman dan penerimaan file** secara langsung melalui jaringan. Program mendukung pengiriman beberapa file sekaligus, menampilkan progress bar saat transfer, serta mencatat log aktivitas untuk keperluan dokumentasi atau debugging.

Kerangka Program



Socket Programming

(Hanya fileserver dan client yang utama)

Program

fileserver.py

server > fileserver.py > ...

```python
import socket
import threading
import os
from datetime import datetime

PORT = 5050
SERVER = socket.gethostbyname(socket.gethostname())
ADDR = (SERVER, PORT)
CHUNKSIZE = 4096
SEPARATOR = "<SEPARATOR>"
LOGFILE = "server_log.txt"

Windsurf: Refactor | Explain | Generate Docstring | ×
def log_event(message):
    with open(LOGFILE, "a") as log:
        log.write(f"[{datetime.now()}] {message}\n")

Windsurf: Refactor | Explain | Generate Docstring | ×
def handle_client(conn, addr):
    print(f"[NEW CONNECTION] {addr}")
    try:
        cmd = conn.recv(1024).decode().strip()
        if cmd == "SEND":
            conn.send("OK".encode())
            filenames = conn.recv(1024).decode().strip().split(SEPARATOR)
            for _ in filenames:
                header = conn.recv(1024).decode()
                filename, filesize = header.split(SEPARATOR)
                filesize = int(filesize)
                with open(os.path.basename(filename), "wb") as f:
                    bytes_read = 0
                    while bytes_read < filesize:
                        chunk = conn.recv(min(CHUNKSIZE, filesize - bytes_read))
                        if not chunk:
                            break
                        f.write(chunk)
                        bytes_read += len(chunk)
                log_event(f"Received file: {filename} from {addr}")

        elif cmd == "RECEIVE":
            conn.send("OK".encode())
            filenames = conn.recv(1024).decode().strip().split(SEPARATOR)
            for filename in filenames:
                if os.path.exists(filename):
                    filesize = os.path.getsize(filename)
                    conn.send(f"{filename}{SEPARATOR}{filesize}".encode())
                    with open(filename, "rb") as f:
                        while (chunk := f.read(CHUNKSIZE)):
                            conn.sendall(chunk)
                    log_event(f"Sent file: {filename} to {addr}")
                else:
                    conn.send(f"ERROR: File {filename} not found.".encode())
    finally:
        conn.close()

Windsurf: Refactor | Explain | Generate Docstring | ×
def start_server():
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(ADDR)
    server.listen()
    print(f"[LISTENING] Server listening on {SERVER}:{PORT}")
    while True:
        conn, addr = server.accept()
        threading.Thread(target=handle_client, args=(conn, addr)).start()

if __name__ == "__main__":
    start_server()
```

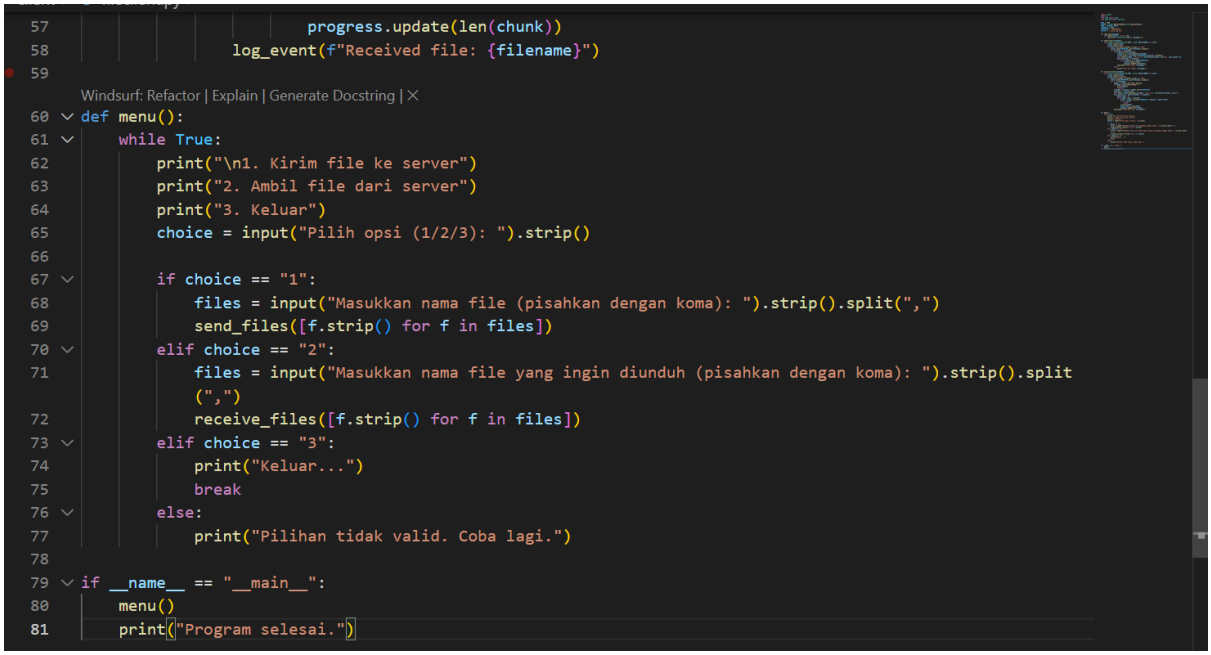
- Bertindak sebagai server yang selalu aktif menerima koneksi dari client.

- Menggunakan socket TCP (SOCK_STREAM) untuk koneksi yang andal.
- Dapat menerima beberapa file dari client atau mengirim file yang diminta client.
- Mencatat aktivitas seperti file yang diterima/dikirim ke dalam file server_log.txt.
- Menggunakan multithreading untuk melayani banyak client secara bersamaan.

fileclient.py

client > fileclient.py > ...

```python
import socket
import os
from tqdm import tqdm
from datetime import datetime

PORT = 5050
SERVER = socket.gethostbyname(socket.gethostname())
ADDR = (SERVER, PORT)
CHUNKSIZE = 4096
SEPARATOR = "<SEPARATOR>"
LOGFILE = "client_log.txt"

Windsurf: Refactor | Explain | Generate Docstring | ×
def log_event(message):
    with open(LOGFILE, "a") as log:
        log.write(f"[{datetime.now()}] {message}\n")

Windsurf: Refactor | Explain | Generate Docstring | ×
def send_files(filenames):
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as client:
        client.connect(ADDR)
        client.send(b"SEND")
        if client.recv(1024).decode().strip() == "OK":
            client.send(SEPARATOR.join(filenames).encode())
            for filename in filenames:
                if os.path.exists(filename):
                    filesize = os.path.getsize(filename)
                    client.send(f"{filename}{SEPARATOR}{filesize}".encode())
                    with open(filename, "rb") as f, tqdm(total=filesize, unit='B', unit_scale=True,
                    desc=filename) as progress:
                        while (chunk := f.read(CHUNKSIZE)):
                            client.sendall(chunk)
                            progress.update(len(chunk))
                    log_event(f"Sent file: {filename}")
                else:
                    print(f"File not found: {filename}")

Windsurf: Refactor | Explain | Generate Docstring | ×
def receive_files(filenames):
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as client:
        client.connect(ADDR)
        client.send(b"RECEIVE")
        if client.recv(1024).decode().strip() == "OK":
            client.send(SEPARATOR.join(filenames).encode())
            for _ in filenames:
                header = client.recv(1024).decode()
                if header.startswith("ERROR"):
                    print(header)
                    continue
                filename, filesize = header.split(SEPARATOR)
                filesize = int(filesize)
                with open(os.path.basename(filename), "wb") as f, tqdm(total=filesize, unit='B',
                unit_scale=True, desc=filename) as progress:
                    bytes_read = 0
                    while bytes_read < filesize:
                        chunk = client.recv(min(CHUNKSIZE, filesize - bytes_read))
                        if not chunk:
                            break
                        f.write(chunk)
                        bytes_read += len(chunk)
```



```
                            progress.update(len(chunk))
                    log_event(f"Received file: {filename}")

Windsurf: Refactor | Explain | Generate Docstring | ×
def menu():
    while True:
        print("\n1. Kirim file ke server")
        print("2. Ambil file dari server")
        print("3. Keluar")
        choice = input("Pilih opsi (1/2/3): ").strip()

        if choice == "1":
            files = input("Masukkan nama file (pisahkan dengan koma): ").strip().split(",")
            send_files([f.strip() for f in files])
        elif choice == "2":
            files = input("Masukkan nama file yang ingin diunduh (pisahkan dengan koma): ").strip().split
            (",")
            receive_files([f.strip() for f in files])
        elif choice == "3":
            print("Keluar...")
            break
        else:
            print("Pilihan tidak valid. Coba lagi.")

if __name__ == "__main__":
    menu()
    print("Program selesai.")
```

- Bertindak sebagai client dengan menu CLI interaktif.
- User dapat memilih untuk mengirim file, menerima file, atau keluar.
- Mendukung pengiriman dan penerimaan **multiple file** sekaligus (dipisahkan dengan koma).
- Menampilkan **progress bar** saat transfer file menggunakan modul tqdm.
- Mencatat aktivitas seperti file yang dikirim/didownload ke dalam client_log.txt.

Cara Kerja

1. Jalankan server (fileserver.py) terlebih dahulu.
2. Jalankan client (fileclient.py) di jendela terminal lain.
3. Di sisi client, user akan memilih aksi:
    - **Kirim file**: Client mengirim nama file dan isi file ke server.
    - **Terima file**: Client meminta file yang tersedia di server.
4. Server merespons sesuai perintah dan mencatat log-nya.
5. File akan dikirim/diterima secara berurutan dengan progress bar.

Output

Fileserver.py

```
PS C:\Users\victo\OneDrive\Desktop\Victor\server> python fileserver.py
[LISTENING] Server listening on 192.168.56.1:5050
```

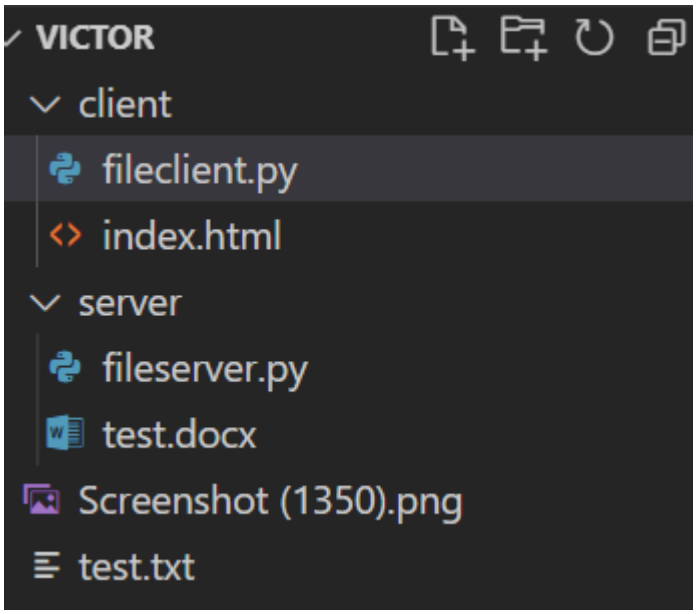
Aktivasi server yang menerima request

Fileclient.py

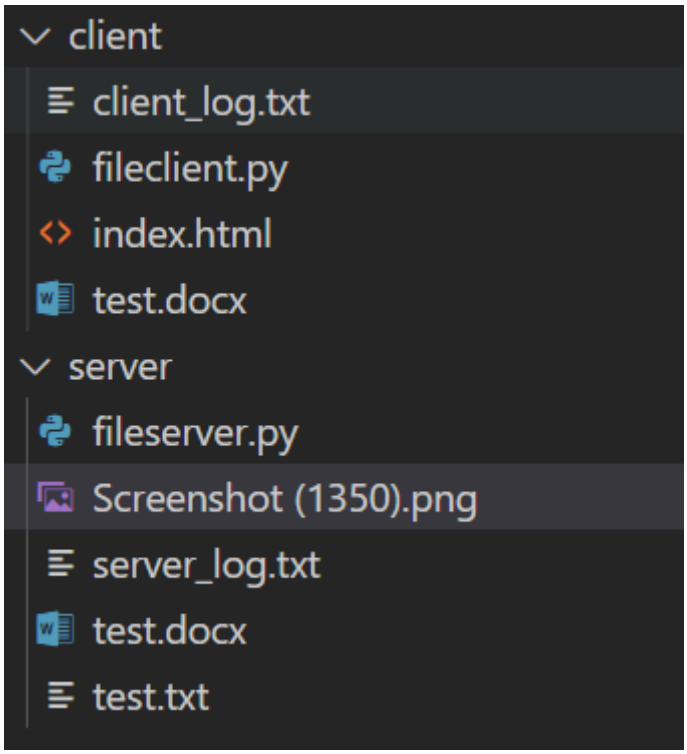
```
PS C:\Users\victo\OneDrive\Desktop\Victor\client> python fileclient.py

1. Kirim file ke server
2. Ambil file dari server
3. Keluar
Pilih opsi (1/2/3): 2
Masukkan nama file yang ingin diunduh (pisahkan dengan koma): test.docx,test.txt
test.docx: 100%|                                                                | 13.2k/13.2k [00:00<00:00, 13.2MB/s]
ERROR: File test.txt not found.
```

Mengambil file dari server test.docx

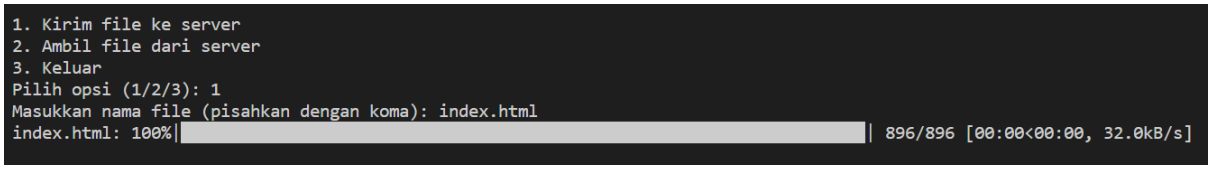
Before :



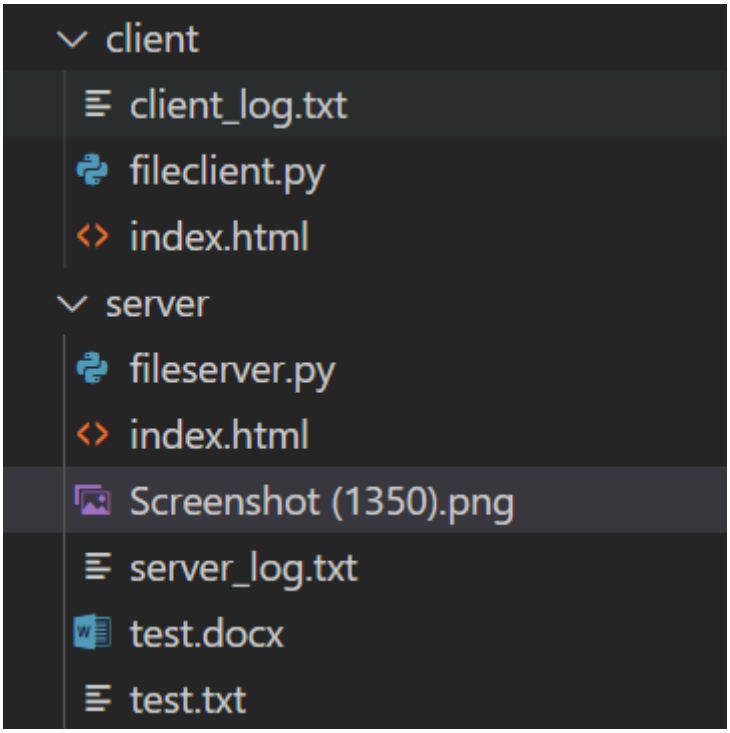
After :



File test.docx berpindah ke client



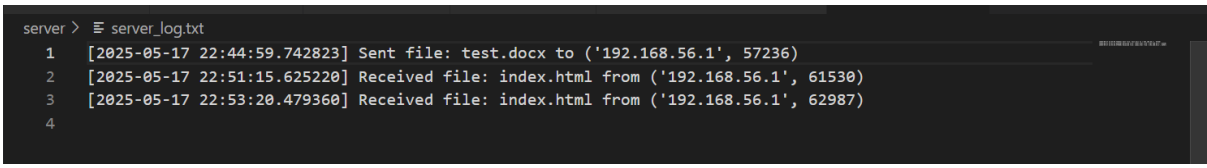
Mengirim file index.html keserver



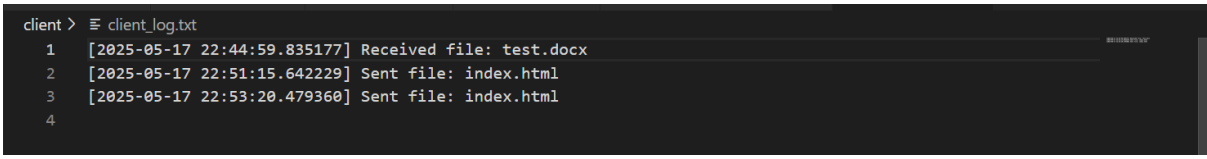
Server menerima file index.html

Aktivitas dicatat di:

- server_log.txt



- client_log.txt



Kesimpulan

Program ini secara keseluruhan membuktikan keberhasilan penerapan socket programming dalam membuat aplikasi client-server yang mendukung pengiriman dan penerimaan file secara efisien dan andal. Dengan fitur tambahan seperti menu interaktif berbasis CLI, dukungan untuk pengiriman beberapa file sekaligus, progress bar yang memberikan visualisasi proses transfer, serta sistem logging otomatis yang mencatat setiap aktivitas komunikasi, program ini menunjukkan praktik pemrograman jaringan yang tidak hanya fungsional tetapi juga mudah digunakan dan dipelihara. Fitur-fitur tersebut menjadikan aplikasi ini relevan sebagai fondasi untuk sistem transfer file yang lebih kompleks di masa depan, baik dalam skala lokal maupun jaringan yang lebih luas.