WEB SERVER BERBASIS TCP
MULTITHREADING DAN SINGLE THREADING

Dosen pembimbing:
Beliau

Disusun oleh:

Farizsyach Razif Januar	1301220439
Imam Wijayanto	1301223117
Muhammad Hilal Abyan	1301223262

PROGRAM STUDI S1 INFORMATIKA
FAKULTAS INFORMATIKA
(2023/2024)

Daftar Isi

BAB 1
PENDAHULUAN

A. Tujuan Penulisan Laporan

Pada penulisan laporan ini tujuan yang akan dicapai yaitu:

a) Dapat membuat serta mengimplementasikan TCP socket dan mengaitkannya ke alamat dan port tertentu
b) Membuat program web server dapat menerima dan memparsing HTTP request yang dikirimkan
c) Membuat web server dapat mencari dan mengambil file (dari file system) yang diminta oleh client
d) Membuat web server dapat membuat HTTP response message yang terdiri dari header dan konten file yang diminta
e) Membuat web server dapat mengirimkan response message yang sudah dibuat ke browser (client) dan dapat ditampilkan dengan benar di sisi client
f) Membuat web server yang dapat mengirimkan pesan "404 *Not Found*" dan dapat ditampilkan dengan benar di sisi client.
g) Membuat web server dengan multithreading dan single threading.
h) Membuat client yang dapat meminta file ke server lalu menampilkan di terminal.

B. Landasan Teori

1. TCP

"Transmission Control Protocol (TCP) adalah salah satu protokol jaringan yang paling umum digunakan untuk mengontrol pengiriman data antar komputer di dalam jaringan. TCP beroperasi di lapisan transport dalam model referensi jaringan OSI (Open Systems Interconnection)" (Adya, 2023).

2. Socket Programing

"Pemrograman socket merupakan pemrograman yang digunakan untuk melakukan komunikasi proses (process-to-process) dalam sebuah jaringan. Selain komunikasi proses, dalam sebuah jaringan komputer juga melakukan komunikasi host (host-to-host). Dalam pengujian pemrograman socket dapat dilakukan dengan menggunakan jenis socket networking sock_stream pada client ke server dengan menggunakan alamat proses (IP dan Port) pada komputer yang sama dengan direktori yang sama dan yang berbeda serta dengan komputer lain yang berbeda." (Supriyanto, 2005).

BAB II
IMPLEMENTASI

1. Buat server_singlethread.py

jarkom-tubes > server_singlethread.py > handle_client

```python
from socket import *
import mimetypes


def handle_client(client_connection):
    request = client_connection.recv(1024).decode()
    print(request)
    # Memisahkan request per baris
    headers = request.split('\n')
    # Mengambil file yang diminta yang ada pada header
    # pada baris pertama setelah request method
    file_requested = headers[0].split()[1]
    # index.html sebagai default saat server dijalankan
    if file_requested == '/':
        file_requested = '/index.html'
    try:
        # Membuka file yang diminta oleh klien
        with open('./data/'+file_requested, 'rb') as file:
            content = file.read()

        # Mengambil content-type dari request klien
        content_type, _ = mimetypes.guess_type(file_requested)
        # Membuat response header dengan kode 200 OK dan tipe content
        response_header = f'HTTP/1.0 200 OK\nContent-Type: {content_type}\n\n'.encode()
        client_connection.send(response_header + content)
    except FileNotFoundError:
        # Error Handling jika file yang direquest tidak ditemukan
        response_header = 'HTTP/1.0 404 NOT FOUND\n\nFile Not Found'.encode()
        response_content = b''
        client_connection.send(response_header + response_content)
    client_connection.close()

def run_server(server_hostname, server_port):
    serverSocket = socket(AF_INET, SOCK_STREAM)
    serverSocket.bind((server_hostname, server_port))
    serverSocket.listen(1)
    print(f'[*] Listening on {server_hostname}:{server_port}...')
    while True:
        client_connection, client_address = serverSocket.accept()
        print(f"[*] Accepted connection from {client_address[0]}:{client_address[1]}")
        handle_client(client_connection)

def main():
    server_hostname = 'localhost'
    server_port = 7070

    # Memulai server dengan host dan port yang ditentukan
    run_server(server_hostname, server_port)

# Memanggil fungsi main
main()

```

2. Buat server_multithread.py

jarkom-tubes > server_multithread.py > run_server

```python
from socket import *
import threading
import mimetypes

def handle_client(client_connection):
    request = client_connection.recv(1024).decode()
    print(request)
    #Memisahkan header per baris
    headers = request.split('\n')
    #Mengambil file yang diminta yang ada pada header
    #pada baris pertama setelah request method
    file_requested = headers[0].split()[1]
    #index.html sebagai default saat server dijalankan
    if file_requested == '/':
        file_requested = '/index.html'
    try:
        #Membuka file yang diminta oleh klien
        with open('./data/'+file_requested, 'rb') as file:
            content = file.read()

        #Mengambil content-type dari request klien
        content_type, _ = mimetypes.guess_type(file_requested)
        #Membuat response header dengan kode 200 OK dan tipe content
        response_header = f'HTTP/1.0 200 OK\nContent-Type: {content_type}\n\n'.encode()
        client_connection.send(response_header+content)
    except FileNotFoundError:
        #Error Hnadling jika file yang direquest tidak ditemukan
        response_header = 'HTTP/1.0 404 NOT FOUND\n\nFile Not Found'.encode()
        response_content = b''
        client_connection.send(response_header+response_content)
    client_connection.close()

def run_server(server_hostname, server_port):
    serverSocket = socket(AF_INET, SOCK_STREAM)
    serverSocket.bind((server_hostname, server_port))
    serverSocket.listen(1)
    print(f'[*] Listening on {server_hostname}:{server_port}...')
    while True:
        client_connection, client_address = serverSocket.accept()
        print(f"[*] Accepted connection from {client_address[0]}:{client_address[1]}")
        client_handler = threading.Thread(target=handle_client, args=(client_connection,))
        client_handler.start()

def main():
    server_hostname = 'localhost'
    server_port = 6969
    run_server(server_hostname, server_port)

# Memanggil fungsi main
main()

```

3. Buat client.py

jarkom-tubes > client.py > ...

```python
from socket import *
import sys

def request_file(server_hostname, server_port, file_requested):
    # Create a TCP/IP socket
    client_socket = socket(AF_INET, SOCK_STREAM)
    client_socket.connect((server_hostname, server_port))

    # Formulate the HTTP GET request
    request = f'GET {file_requested} HTTP/1.0\r\nHost: {server_hostname}\r\n\r\n'
    client_socket.send(request.encode())

    # Receive the response from the server
    response = b''
    while True:
        chunk = client_socket.recv(1024)
        if not chunk:
            break
        response += chunk

    # Close the connection
    client_socket.close()

    # Split the response into headers and body
    header_end = response.find(b'\r\n\r\n')
    if header_end != -1:
        header = response[:header_end].decode()
        body = response[header_end + 4:]
    else:
        header = response.decode()
        body = b''

    # Print the headers and the body
    print(header)
    if body:
        print(body.decode())

if len(sys.argv) != 4:
    print(f"Usage: {sys.argv[0]} <serverhost> <serverport> <filerequested>")
    sys.exit(1)
server_hostname = sys.argv[1]  # Server hostname
server_port = int(sys.argv[2])  # Server port
file_requested = sys.argv[3]  # File requested

# Request the file from the server and print its contents to the terminal
request_file(server_hostname, server_port, file_requested)

```

4. Jalankan server single threading, hasil running pada localhost:7070
5. Jalankan server multithreading, hasil running pada localhost:6969
6. Hasil running pada client
7.