

Modul

Pembelajaran: Implementasi FTP dengan Python

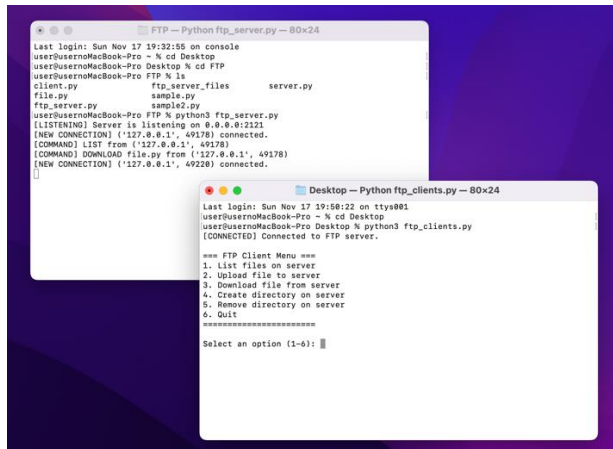
A. Tujuan Pembelajaran

Setelah mempelajari modul ini, mahasiswa diharapkan dapat:

1. Memahami konsep dasar File Transfer Protocol (FTP) dan aplikasinya dalam jaringan komputer.
2. Mampu mengimplementasikan sistem FTP sederhana menggunakan Python.
3. Memahami aspek keamanan dalam implementasi FTP.
4. Melakukan analisis masalah dan optimasi dalam implementasi server-client FTP.

B. Pengantar FTP

1.1. Apa itu FTP?



The image shows two terminal windows. The top window, titled 'FTP - Python ftp_server.py -- 80x24', displays the server's operation. It shows the user navigating to the 'FTP' directory, running 'python3 ftp_server.py', and the server listening on 0.0.0.0:2121. A client connects from 127.0.0.1:49178, and the server successfully downloads 'file.py'. The bottom window, titled 'Desktop - Python ftp_clients.py -- 80x24', shows the client's operation. It shows the user running 'python3 ftp_clients.py', which connects to the server. A menu is displayed with options: 1. List files on server, 2. Upload file to server, 3. Download file from server, 4. Create directory on server, 5. Remove directory on server, 6. Quit. The user selects option 3 to download the file.

- File Transfer Protocol (FTP) adalah protokol standar untuk mentransfer file antara client dan server melalui jaringan.
- FTP digunakan untuk mengunggah (upload) atau mengunduh (download) file.

1.2. Fungsi Utama FTP

- **Upload:** Mengirim file dari client ke server.
- **Download:** Mengambil file dari server ke client.
- **List:** Melihat daftar file di server.
- **Delete/Rename:** Menghapus atau mengganti nama file (opsional).

1.3. Kelebihan dan Kekurangan FTP

- **Kelebihan:** Sederhana, mendukung transfer file besar.
- **Kekurangan:** Protokol asli FTP tidak aman (data dikirim dalam teks biasa)

C. Implementasi Dasar FTP

2.1. Struktur Sistem

- **Server:** Menangani koneksi client, menerima permintaan, dan mengelola file.
- **Client:** Mengirim perintah ke server dan menampilkan respons.

/ftp-server

```
|— /files          # Direktori utama untuk file yang diunggah dan diunduh
|  |— file1.txt
|  |— file2.jpg
|  |— subfolder    # Contoh subdirektori
|     |— document.pdf
|— server.py       # File program server
```

/ftp-client

```
|— /downloads      # Tempat penyimpanan file hasil download dari server
|  |— file1.txt
|  |— file2.jpg
|— /uploads        # Tempat penyimpanan file untuk diunggah (opsional)
|  |— example.txt
|  |— notes.pdf
|— client.py       # File program client
```

2.2. Kode Server FTP

Berikut adalah ftp server menggunakan Python:

Buatlah file dengan nama ftp_server.py dengan baris code sebagai berikut:

```
import socket  
  
import os  
  
import threading
```

- **Socket** : Modul untuk membuat koneksi jaringan.
- **os** : Digunakan untuk operasi file dan direktori.
- **Threading** : Memungkinkan server menangani banyak client secara bersamaan dengan membuat thread baru untuk setiap koneksi.

Konfigurasi Server

```
HOST = '127.0.0.1'  
PORT = 2121  
BASE_DIR = "ftp_server_files" # Direktori penyimpanan file server  
  
# Buat direktori jika belum ada  
if not os.path.exists(BASE_DIR):  
    os.makedirs(BASE_DIR)
```

- **HOST**: Alamat IP server. 127.0.0.1 berarti server berjalan secara lokal.
- **PORT**: Port yang digunakan server untuk mendengarkan koneksi.
- **BASE_DIR**: Direktori tempat server menyimpan file. Semua operasi file terbatas pada direktori ini.
- **os.makedirs(BASE_DIR)**: Membuat direktori BASE_DIR jika belum ada. Ini memastikan server memiliki lokasi penyimpanan yang valid.

Fungsi Utama: `handle_client`

```
def handle_client(client_socket, addr):  
    print(f'[NEW CONNECTION] {addr} connected.')  
    while True:  
        try:  
            command = client_socket.recv(1024).decode().strip()  
            if not command:  
                break
```

- `handle_client`: Fungsi untuk menangani koneksi dengan satu client.
- `client_socket.recv(1024)`: Menerima data dari client hingga 1024 byte.
- `command.strip()`: Menghapus spasi tambahan atau karakter newline dari perintah yang diterima.
- Jika perintah kosong (misalnya client terputus), loop dihentikan.

Perintah `LIST`

```
if command == "LIST":  
    files = "\n".join(os.listdir(BASE_DIR))  
    client_socket.send(files.encode())
```

- `os.listdir(BASE_DIR)`: Mengambil daftar file dan direktori di `BASE_DIR`.
- `"\n".join()`: Menggabungkan nama file menjadi string dengan setiap file berada di baris baru.
- `client_socket.send(files.encode())`: Mengirim daftar file ke client.

Perintah `UPLOAD`

```
elif command.startswith("UPLOAD"):  
    _, filename = command.split(maxsplit=1)  
    filepath = os.path.join(BASE_DIR, filename)  
    with open(filepath, 'wb') as f:  
        while True:  
            data = client_socket.recv(1024)  
            if data == b'DONE':  
                break  
            f.write(data)  
    client_socket.send(f'[SUCCESS] File {filename} uploaded.'.encode())
```

- `command.startswith("UPLOAD")`: Memastikan perintah diawali dengan "UPLOAD".

- `command.split(maxsplit=1)`: Membagi perintah menjadi "UPLOAD" dan nama file.
- `os.path.join(BASE_DIR, filename)`: Membuat path lengkap untuk file di direktori server.
- `open(filepath, 'wb')`: Membuka file dalam mode tulis biner.
- **Loop `recv`**: Menerima data file dalam potongan kecil (chunks) hingga sinyal `DONE` diterima.
- `f.write(data)`: Menyimpan setiap chunk ke file.
- Setelah file selesai diunggah, server mengirim pesan sukses.

Perintah `DOWNLOAD`

```
elif command.startswith("DOWNLOAD"):
    _, filename = command.split(maxsplit=1)
    filepath = os.path.join(BASE_DIR, filename)
    if os.path.exists(filepath):
        with open(filepath, 'rb') as f:
            while (data := f.read(1024)):
                client_socket.send(data)
            client_socket.send(b'DONE')
    else:
        client_socket.send("[ERROR] File not found.".encode())
```

- `command.startswith("DOWNLOAD")`: Memastikan perintah diawali dengan "DOWNLOAD".
- `os.path.exists(filepath)`: Memeriksa apakah file yang diminta ada di direktori server.
- `open(filepath, 'rb')`: Membuka file dalam mode baca biner.
- **Loop `f.read(1024)`**: Membaca file dalam potongan kecil (chunks) dan mengirimnya ke client.
- Setelah file selesai dikirim, server mengirim sinyal `DONE`.
- Jika file tidak ditemukan, server mengirim pesan error.

Perintah `QUIT`

```
elif command == "QUIT":
    client_socket.send("[DISCONNECTED] Goodbye!".encode())
    break
```

- `command == "QUIT"`: Memastikan perintah adalah "QUIT".
- `client_socket.send()`: Mengirim pesan perpisahan ke client.
- `break`: Menghentikan loop, menutup koneksi dengan client.

Perintah Tidak Valid

```
else:  
    client_socket.send("[ERROR] Invalid command.".encode())
```

Jika perintah tidak dikenal, server mengirim pesan error ke client.

Fungsi Utama: `start_server`

```
def start_server():  
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
    server_socket.bind((HOST, PORT))  
    server_socket.listen()  
    print(f'[STARTED] Server running on {HOST}:{PORT}')  
    while True:  
        client_socket, addr = server_socket.accept()  
        thread = threading.Thread(target=handle_client, args=(client_socket, addr))  
        thread.start()
```

- `socket.socket(socket.AF_INET, socket.SOCK_STREAM)`: Membuat socket TCP.
- `server_socket.bind((HOST, PORT))`: Mengikat socket ke alamat dan port tertentu.
- `server_socket.listen()`: Membuat server mendengarkan koneksi masuk.
- `server_socket.accept()`: Menerima koneksi baru dari client. Mengembalikan socket client dan alamat client.
- `threading.Thread`: Membuat thread baru untuk setiap koneksi client.
- `thread.start()`: Menjalankan thread, yang memanggil `handle_client`.

D. Alur Eksekusi Server

1. **Start Server**: Server berjalan pada `HOST` dan `PORT`.
2. **Accept Client**: Ketika client terhubung, server membuat thread baru untuk menangani koneksi.
3. **Handle Commands**:
 - `LIST`: Mengirim daftar file di direktori server.
 - `UPLOAD`: Menerima file dari client dan menyimpannya.
 - `DOWNLOAD`: Mengirim file ke client.
 - `QUIT`: Mengakhiri koneksi client.
 - **Invalid Command**: Mengirim pesan error.
4. **Multi-threading**: Setiap koneksi client berjalan di thread terpisah, memungkinkan banyak client dilayani bersamaan.

E. FTP CLIENT

Buatlah file `ftp_client.py`

```
import socket
import os
```

- **socket**: Digunakan untuk membuat koneksi TCP antara client dan server.
- **os**: Digunakan untuk operasi file dan direktori lokal, seperti memeriksa keberadaan file atau direktori.

Konfigurasi Client

```
SERVER_HOST = '127.0.0.1' # Alamat server
SERVER_PORT = 2121
BUFFER_SIZE = 1024
```

- **SERVER_HOST**: Alamat IP server. `127.0.0.1` menunjukkan server berjalan di mesin lokal (localhost).
- **SERVER_PORT**: Port server yang digunakan untuk koneksi.
- **BUFFER_SIZE**: Ukuran buffer (dalam byte) untuk data yang dikirim/diterima. Diatur ke 1024 byte.

```
def print_menu():
    """Menampilkan menu opsi kepada pengguna."""
    print("\n=== FTP Client Menu ===")
    print("1. List files on server")
    print("2. Upload file to server")
    print("3. Download file from server")
    print("4. Create directory on server")
    print("5. Remove directory on server")
    print("6. Quit")
    print("=====\n")
```

- Menampilkan menu opsi yang tersedia untuk pengguna.
- Setiap nomor dalam menu mewakili fitur FTP (daftar file, upload, download, dll.).

Fungsi Utama: `start_client()`

1. Membuka Koneksi dengan Server

```
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect((SERVER_HOST, SERVER_PORT))
print("[CONNECTED] Connected to FTP server.")
```

- `socket.AF_INET`: Menentukan bahwa koneksi menggunakan IPv4.
- `socket.SOCK_STREAM`: Menentukan protokol TCP.
- `connect((SERVER_HOST, SERVER_PORT))`: Menghubungkan client ke server pada IP dan port yang ditentukan.

2. Loop Utama

```
while True:
    print_menu()
    try:
        choice = input("Select an option (1-6): ").strip()
```

- `while True`: Membuat loop terus berjalan hingga client memilih opsi `Quit`.
- `input("Select an option (1-6): ")`: Meminta pengguna untuk memilih fitur yang diinginkan.

3. Fitur-Fitur dalam Menu

a. List Files on Server

```
if choice == '1': # List files
    client_socket.send("LIST".encode())
    response = client_socket.recv(BUFFER_SIZE).decode()
    print("\nFiles on server:")
    print(response)
```

- `client_socket.send("LIST".encode())`: Mengirim perintah `LIST` ke server.
- `client_socket.recv(BUFFER_SIZE).decode()`: Menerima daftar file dari server dan menampilkan hasilnya.

b. Upload File to Server

```
elif choice == '2': # Upload file
    filepath = input("Enter the path of the file to upload: ").strip()
    if os.path.exists(filepath):
        filename = os.path.basename(filepath)
        client_socket.send(f"UPLOAD {filename}".encode())
        with open(filepath, 'rb') as f:
            while (data := f.read(BUFFER_SIZE)):
                client_socket.send(data)
        client_socket.send(b'DONE')
        response = client_socket.recv(BUFFER_SIZE).decode()
        print(response)
    else:
        print("File not found. Please check the path.")
```

- `os.path.exists(filepath)`: Memeriksa apakah file ada.
- `os.path.basename(filepath)`: Mendapatkan nama file dari path lengkap.
- **Mengirim file:**
 - Membaca file dalam potongan kecil (`BUFFER_SIZE` bytes) menggunakan loop.
 - Mengirimkan potongan file ke server.
- `client_socket.send(b'DONE')`: Memberitahu server bahwa pengiriman file selesai.
- **Error Handling:** Memberi peringatan jika file tidak ditemukan.

c. Download File from Server

```
elif choice == '3': # Download file
    filename = input("Enter the name of the file to download: ").strip()
    client_socket.send(f"DOWNLOAD {filename}".encode())
    save_path = os.path.join(os.getcwd(), filename)
    with open(save_path, 'wb') as f:
        while True:
            data = client_socket.recv(BUFFER_SIZE)
            if data == b'DONE':
                break
            f.write(data)
    print(f"File {filename} downloaded to {save_path}.")
```

- `client_socket.send(f"DOWNLOAD {filename}".encode())`: Mengirim perintah `DOWNLOAD` beserta nama file ke server.
- `os.path.join(os.getcwd(), filename)`: Menentukan lokasi penyimpanan file di direktori kerja saat ini.

- **Menyimpan file:**

- Menerima potongan data dari server dan menuliskannya ke file lokal.
- Berhenti ketika server mengirim sinyal `DONE`.

d. Create Directory on Server

```
elif choice == '4': # Create directory
    dirname = input("Enter the name of the directory to create: ").strip()
    client_socket.send(f"MKDIR {dirname}".encode())
    response = client_socket.recv(BUFFER_SIZE).decode()
    print(response)
```

- `client_socket.send(f"MKDIR {dirname}".encode())`: Mengirim perintah `MKDIR` dengan nama direktori yang akan dibuat ke server.
- `response`: Menampilkan hasil operasi dari server (sukses atau gagal).

e. Remove Directory on Server

```
elif choice == '5': # Remove directory
    dirname = input("Enter the name of the directory to remove: ").strip()
    client_socket.send(f"RMDIR {dirname}".encode())
    response = client_socket.recv(BUFFER_SIZE).decode()
    print(response)
```

- `client_socket.send(f"RMDIR {dirname}".encode())`: Mengirim perintah `RMDIR` ke server.
- `response`: Menampilkan pesan respons server.

```
elif choice == '6': # Quit
    client_socket.send("QUIT".encode())
    response = client_socket.recv(BUFFER_SIZE).decode()
    print(response)
    break
```

- `client_socket.send("QUIT".encode())`: Mengirim perintah `QUIT` ke server.
- `break`: Mengakhiri loop, menutup koneksi.

4. Menangani Kesalahan

```
except Exception as e:  
    print(f'[ERROR] {e}')
```

Jika ada kesalahan (misalnya koneksi terputus), program akan menangkapnya dan mencetak pesan error.

5. Menutup Koneksi

```
client_socket.close()  
  
print("[DISCONNECTED] Client closed connection.")
```

- `client_socket.close()`: Menutup koneksi dengan server.
- **Pesan Penutup**: Memberitahukan pengguna bahwa koneksi telah ditutup.

F. Alur dari program client:

1. **Inisialisasi:**
 - Membuat socket.
 - Terhubung ke server FTP di `SERVER_HOST` dan `SERVER_PORT`.
2. **Menu Utama:**
 - Menampilkan pilihan:
 1. **List files**: Menampilkan daftar file di server.
 2. **Upload file**: Mengirim file dari client ke server.
 3. **Download file**: Mengunduh file dari server ke client.
 4. **Create directory**: Membuat direktori di server.
 5. **Remove directory**: Menghapus direktori di server.
 6. **Quit**: Menutup koneksi.
3. **Pemilihan Opsi:**
 - Mengirim perintah sesuai opsi ke server.
 - Jika diperlukan, membaca/mengirim file secara bertahap menggunakan buffer.
 - Menampilkan respons server (sukses/gagal).
4. **Keluar:**
 - Mengirim perintah `QUIT`.
 - Menutup koneksi socket.
 - Menampilkan pesan bahwa client telah terputus.

Alur di atas berjalan dalam loop hingga pengguna memilih opsi `Quit`.

G. Implementasikan Fungsi

- **List files:** Menampilkan daftar file di server.
- **Upload file:** Mengirim file dari client ke server.

Select an option (1-6): 2

Enter the path of the file to upload: C:\Users\User\Documents\example.txt

- **Download file:** Mengunduh file dari server ke client.

Select an option (1-6): 3

Enter the name of the file to download: file1.txt

- **Create directory:** Membuat direktori di server.

Select an option (1-6): 4

Enter the name of the directory to create: new_folder

- **Remove directory:** Menghapus direktori di server.

Select an option (1-6): 5

Enter the name of the directory to remove: old_folder

Dokumentasikan dalam bentuk laporan praktikum pdf dan submit di sunan.