Dokumentasi Alur Kerja Program Client-Server

1. Pendahuluan

Program ini merupakan implementasi sederhana komunikasi dua arah antara client dan server menggunakan socket programming di Python. Program ini terdiri dari dua file utama, yaitu `server.py` dan `client.py`, yang berfungsi sebagai pihak server dan client.

2. Deskripsi Umum

Server akan menunggu koneksi dari client dan menerima pesan yang dikirimkan, kemudian membalas pesan tersebut. Client dapat mengirim pesan ke server dan menerima balasan dari server. Proses ini berlangsung hingga salah satu pihak mengirimkan pesan 'exit'.

3. Alur Kerja server.py

Berikut adalah langkah-langkah kerja server:

1. Import library socket.
2. Tentukan alamat IP dan port server.
3. Buat socket TCP menggunakan `socket.AF_INET` dan `socket.SOCK_STREAM`.
4. Bind socket ke alamat IP dan port.
5. Mulai mendengarkan koneksi dari client.
6. Terima koneksi client menggunakan `accept()`.
7. Masuk ke loop komunikasi dua arah:

- - Terima data dari client.
- - Tampilkan pesan client.
- - Minta input dari admin/server untuk membalas.
- - Kirim balasan ke client.
- - Jika balasan adalah 'exit', keluar dari loop.

8. Tutup koneksi dengan client.

4. Alur Kerja client.py

Berikut adalah langkah-langkah kerja client:

1. Import library socket.
2. Tentukan alamat IP dan port server yang akan dikoneksikan.
3. Buat socket TCP.

4. Koneksi ke server menggunakan `connect()`.
5. Masuk ke loop komunikasi dua arah:

- - Minta input dari user untuk dikirim ke server.
- - Kirim pesan ke server.
- - Jika pesan adalah 'exit', keluar dari loop.
- - Terima balasan dari server.
- - Tampilkan balasan dari server.

6. Tutup koneksi dengan server.

5. Penutup

Program client-server ini cocok digunakan sebagai dasar dalam mempelajari komunikasi jaringan menggunakan Python. Dengan pengembangan lebih lanjut, sistem ini bisa digunakan untuk membuat aplikasi chatting, sistem kendali jarak jauh, atau layanan berbasis jaringan lainnya.

LAMPIRAN KODE

server.py

```
import socket  # Import library socket

HOST = '127.0.0.1'  # Alamat IP lokal (localhost)
PORT = 65432        # Port yang digunakan (harus sama dengan client)

# Membuat socket TCP (SOCK_STREAM untuk TCP)
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Bind socket ke alamat dan port
server_socket.bind((HOST, PORT))

# Mulai mendengarkan koneksi yang masuk
server_socket.listen()

print(f"[SERVER] Menunggu koneksi di {HOST}:{PORT}...")

# Menerima koneksi dari client (blocking, menunggu hingga client connect)
conn, addr = server_socket.accept()
print(f"[SERVER] Terhubung dengan {addr}")

# Loop utama untuk komunikasi dua arah
```

```
while True:
    # Terima data dari client (maks 1024 byte)
    data = conn.recv(1024)
    if not data:
        # Jika data kosong, berarti client memutus koneksi
        print("[SERVER] Koneksi ditutup oleh client.")
        break
    print(f"[CLIENT] {data.decode()}")  # Tampilkan pesan dari client

    # Minta input dari server (admin) untuk membalas
    reply = input("[SERVER] Ketik balasan: ")
    conn.sendall(reply.encode())  # Kirim balasan ke client

    # Jika balasan adalah "exit", maka keluar dari loop
    if reply.lower() == 'exit':
        print("[SERVER] Keluar.")
        break

# Tutup koneksi dengan client
conn.close()
```

client.py

```
import socket  # Import library socket

HOST = '127.0.0.1'  # Alamat IP server (localhost)
PORT = 65432        # Port yang digunakan server

# Membuat socket TCP
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Koneksi ke server menggunakan IP dan port
client_socket.connect((HOST, PORT))

# Loop utama untuk komunikasi dua arah
while True:
    # Minta input dari user untuk dikirim ke server
    msg = input("[CLIENT] Ketik pesan: ")
    client_socket.sendall(msg.encode())  # Kirim pesan ke server

    # Jika pesan adalah "exit", keluar dari loop
    if msg.lower() == 'exit':
        print("[CLIENT] Keluar.")
```

```
        break

    # Terima balasan dari server (maks 1024 byte)
    data = client_socket.recv(1024)
    if not data:
        # Jika tidak ada data, berarti server menutup koneksi
        print("[CLIENT] Koneksi ditutup oleh server.")
        break
    print(f"[SERVER] {data.decode()}")  # Tampilkan balasan dari server

# Tutup koneksi dengan server
client_socket.close()
```