Server & Client Report

Sajjin Nijjar
Sept. 27, 2023

Requirements

Tasks	Status
Server and client use domain sockets	Implemented
Server has error handles	Implemented
Client takes one or more files using CMD	Implemented
Clients sends server file name, file size and data for each file	Implemented
Server saves files in a directory (server_storage)	Implemented
Server handles duplicates	Implemented
Server runs until user press ctrl+c	Implemented
The client runs until all files sent	Implemented

Platforms

Server & Client has been tested on:

- Ubuntu 22.04.3 LTS

Language

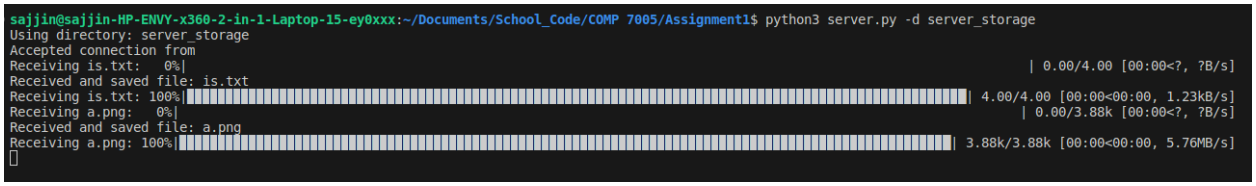
- Python 3.10.12

Design

Functions

Server:

Input on CMD: `python3 server.py -d directory_path`



Main():

Main is to establish a connection

```
    server_socket = socket.socket(socket.AF_UNIX, socket.SOCK_STREAM)
    server_socket.bind(socket_path)
    server_socket.listen(1)
while True:
    client_socket, client_address = server_socket.accept()
    print(f"Accepted connection from {client_address}")
    receive_files(client_socket)
```

receive _files():

Receive files is to get the file name and size and send them to the next functions

```
        while True:

            received = connection.recv(BUFFER_SIZE).decode()
            file_name, file_size = received.split(SEPARATOR)
            file_name = os.path.basename(file_name)
            file_size = int(file_size)

            file_path = duplicate_files()

            write_file()
```

duplicate_files():

Duplicate files checks the file path and see if there are any of the same file and if so it will add by 1 to the name of the file e.g. file.txt - file3.txt (already exists) file.txt is sent again its new name will be file4.txt

```
original_file_name = file_name
    file_counter = 1
    while True:
        file_path = os.path.join(getfilepath, storage_directory, file_name)
        if not os.path.exists(file_path):
```

```
            break
        base_name, extension = split file name to name and extension
        file_name = f"{base_name}_{file_counter}{extension}"
        file_counter += 1
    file_path = os.path.join(getfilepath, storage_directory, file_name)
    return file_path
```

write_file()

Write file simply writes the file at the file path and receives the data for the file sent and writes it all in the bytes.

```
    with open(file_path, 'wb') as file:
        received_data = 0
        while received_data < file_size:
            data = connection.recv(file_size)
            if not data:
                break
            file.write(data)
            received_data += len(data)
        print(f"Received and saved file: {file_name}")
```

Client:

Input on CMD: `python3 client.py test.txt test.png test.gif`

Main():

Main sets up the connection.

```
client_socket = socket.socket(socket.AF_UNIX, socket.SOCK_STREAM)
client_socket.connect(socket_path)
file_names = sys.argv[1:]
send_files( SEPARATOR, file_names, client_socket)
```

Send_files():

Send file first sends the filename and file size. And then reads the files contents in bytes and sends it to the server.

```
for file_name in file_names:
    file_size = os.path.getsize(file_name)

client_socket.send(f"{file_name}{SEPARATOR}{file_size}".encode())

            with open(file_name, 'rb') as file:
                while True:
                    data = file.read(file_size)
                    if not data:
                        break
                    client_socket.send(data)
```

Testing

- I have tested by sending single files 3 files of the same type and 3 files of different types and sizes.
- I tested large files and files with no data.
- Make an error occur on the server and make sure the server does not turn off.
- I have also tested by turning off the server while it was receiving data.

Problems

- The only issue that I have found while testing is when sending 3 or bigger GB files VS code crashes but files still send fine.