Đại học Quốc gia
Thành phố Hồ Chí Minh

Trường Đại học Khoa học Tự nhiên

khoa công nghệ thông tin

Báo cáo đồ án Socket

CSC10008 – MẠNG MÁY TÍNH

Ngô Nguyễn Thế Khoa 23127065
Hình Diễm Xuân 23127524

Ngày 25 tháng 6 năm 2024

Mục lục

1 Thông tin nhóm

- **Môn:** Mạng máy tính
- **Lớp học phần:** 23CLC09
- **Giảng viên hướng dẫn:** Lê Hà Minh
- **Thành viên:**

STT	Họ và tên	MSSV	Email	Vai trò
1	Ngô Nguyễn Thế Khoa	23127065	nntkhoa23@clc.fitus.edu.vn	Trưởng nhóm
2	Hình Diễm Xuân	23127524	hdxuan23@clc.fitus.edu.vn	Thành viên

- **Công cụ hỗ trợ:**
  - **Git**, **GitHub**
  - **Canva**
  - **Google Sheets**

2 Bảng đánh giá mức độ hoàn thành

STT	Yêu cầu	Tiến độ
Phần I		
1	Client có thể nhận được danh sách các file từ Server và ctrl-c	100%
2	Client có thể nhận lần lượt từng file thành công từ Server. Server có thể gửi file thành công tới Client	100%
3	Hiển thị percent download file và phát hiện những file cần download tiếp theo	100%
Phần II		
4	Client có thể nhận được danh sách các file từ Server và ctrl-c	100%
5	2s quét file input.txt 1 lần	100%
6	Hiển thị phần trăm download files	100%
7	Client có thể nhận files thành công từ Server. Tập tin sau khi download phải đúng và đủ dung lượng	100%
8	Độ ưu tiên CRITICAL, HIGH, NORMAL	100%
Tổng kết báo cáo *Hoàn thành 100% yêu cầu đồ án* *Không xảy ra lỗi khi vận hành chương trình*		

3 Nội dung

3.1 Thông tin chung về đồ án:

1. **Tên đồ án:** Lập trình Socket
2. **Môi trường lập trình:** Visual Studio Code, Windows
3. **Ngôn ngữ lập trình:** Python
4. **Danh sách thư viện chuẩn được sử dụng (standard libs)**
    - `argparse` - parse command line arguments
    - `datetime` - parse datetime with custom format
    - `enum` - usage of enum type
    - `json` - data serialization
    - `os` - file operations
    - `pathlib` - usage of filesystem path
    - `re` - usage of regex
    - `socket` - creating sockets
    - `sys` - rendering progress bar (pure progress bar)
    - `threading` - handling multiple clients
    - `time` - time operations
    - `typing` - usage of generic type
    - `unittest` - testing functions
5. **Danh sách thư viện bổ sung được sử dụng (external libs)**
    - `customtkinter` - making GUI
    - `python-dotenv` - load configs from `.env` files
    - `rich` - rendering more beautiful console progress bar
    - `watchdog` - watching file changes
6. **Giao thức trao đổi giữa Client và Server:** Giao thức TCP

3.2 Một số lưu ý

- Cần cấu hình file `.env` trước khi chạy chương trình (cấu hình cho Server và Client)

- Tải thư viện bổ sung bằng lệnh `pip install -r requirements.txt` (trong thư mục `Source`)
- Chạy Server trước khi chạy bất kì Client nào
- Cần chạy một Socket Server khác với địa chỉ đã được dùng trước đó, nếu không sẽ không thể tạo được một Socket Server mới

3.3 Cấu trúc đồ án

(Source): Thư mục chứa mã nguồn của chương trình

(app): Thư mục chứa mã nguồn của Server và Client

(client): Thư mục ứng dụng hoạt động (có thể tùy chỉnh)

(downloads): Thư mục chứa các file đã tải về (đường dẫn mặc định)

input.txt: File chứa danh sách file cần tải về (đường dẫn mặc định)

(server): Thư mục ứng dụng hoạt động (có thể tùy chỉnh)

(resources): Thư mục chứa các file có thể tải về (đường dẫn mặc định)

resources.json: File chứa danh sách file có thể tải về (tự động tạo và cập nhật)

(classes): Thư mục chứa các class hỗ trợ cho Server và Client

(shared): Thư mục chứa các biến chung cho Server và Client

(tests): Thư mục chứa các chương trình dùng để test hàm

(utils): Thư mục chứa các hàm hỗ trợ cho Server và Client

.env: File cấu hình cho Server và Client

requirements.txt: File chứa danh sách thư viện cần cài đặt

server.py: Mã nguồn của Server

client.py: Mã nguồn của Client

Report.pdf: Báo cáo đồ án

3.4 Nguyên lí hoạt động - Kịch bản giao tiếp:

- **Giao thức kết nối:** TCP
- **Kịch bản giao tiếp:**

  Client:

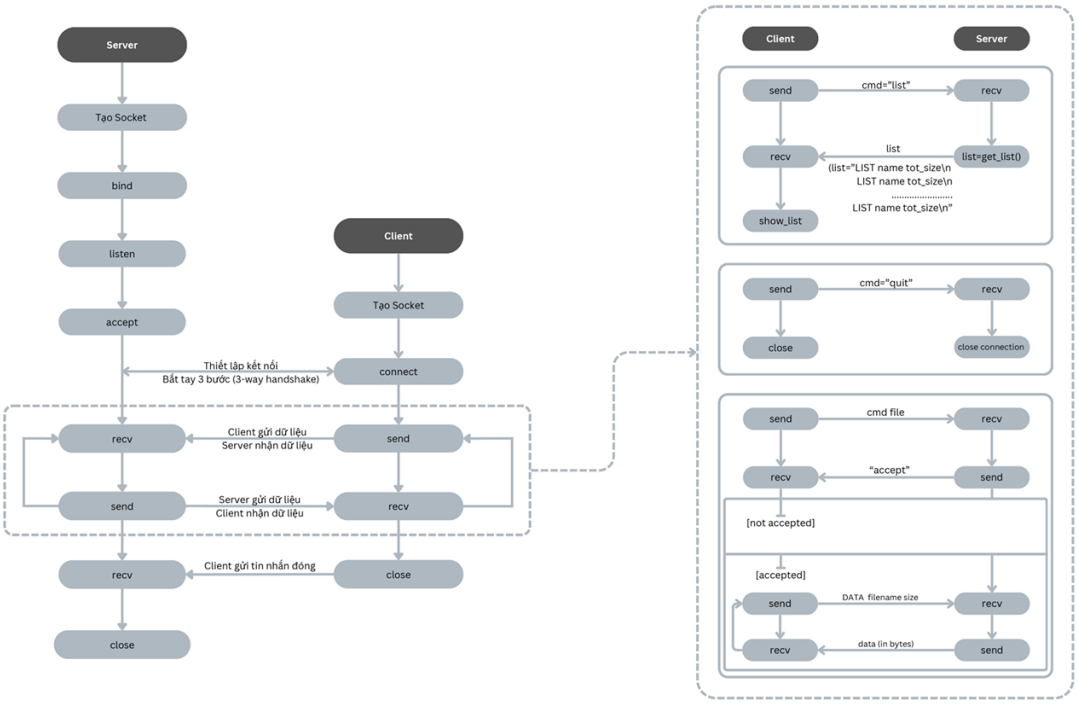
  - `help`: Lệnh hiển thị danh sách các lệnh có thể sử dụng

- `quit`: Lệnh thoát chương trình, được gửi khi có tín hiệu kết thúc phiên chạy (Ctrl + C hoặc lệnh `quit`)
- `list`: Lệnh hiển thị danh sách các file có thể tải về từ Server
  1. Chờ nhận dữ liệu từ Server và hiển thị danh sách file
     * Danh sách các file được gửi dưới dạng
       '`LIST {sep} {filename} {sep} {file_sz} {sep}`'
     * Nếu trong quá trình nhận dữ liệu, nhận được lệnh `terminate` từ Server ⇒ Kết thúc phiên chạy
- `file`: Lệnh tải file được gửi từ Client để thiết lập quá trình tải file
  1. Chờ nhận lệnh `accept` từ Server
  2. Nếu nhận được lệnh `accept`, tiến hành tải file bằng cách gửi thông điệp dưới dạng
     '`DATA {sep} {filename} {sep} {chunk_sz} {sep}`'
  3. Chờ nhận dữ liệu từ Server và lưu vào file
     * Nếu trong quá trình nhận dữ liệu, nhận được lệnh `done` từ Server ⇒ Tải file thành công
     * Nếu trong quá trình nhận dữ liệu, nhận được lệnh `terminate` từ Server ⇒ Kết thúc phiên chạy

Server:

- `done`: Lệnh thông báo tải file thành công
- `invalid`: Lệnh thông báo lệnh nhận được từ Client không hợp lệ
- `terminate`: Lệnh ngắt kết nối đến tất cả Client (được gửi đi khi Server nhận tín hiệu kết thúc phiên chạy)
- `accept`: Lệnh thông báo Server hoàn tất thiết lập tải file từ Client
  1. Chờ nhận thông điệp từ Client và tiến hành tải file
     * Gửi dữ liệu file dưới dạng `bytes` cho Client
     * Trong quá trình nhận dữ liệu, nếu nhận được lệnh `quit` từ Client ⇒ Kết thúc phiên chạy của Client Socket tương ứng bên Server

- **Kiểu dữ liệu của thông điệp:** kiểu `string`, gửi bằng lệnh `send, sendall` và nhận bằng lệnh `recv` theo FORMAT được quy ước
- **Sơ đồ minh họa một phần quá trình giao tiếp giữa Client và Server sử dụng giao thức TCP:**



Hình 1: Sơ đồ minh họa kịch bản giao tiếp

4 Các tính năng chương trình

4.1 Các tính năng cơ bản

1. Server phục vụ đồng thời nhiều Client cùng lúc.
2. Client nhận được danh sách tài nguyên từ Server.
3. Thêm file mới vào danh sách tải và thêm vào độ ưu tiên tải nếu muốn (Phần 2).
4. Duyệt file mỗi khi tải xong từng file (Phần 1) và duyệt file mỗi 2 giây (Phần 2).
5. Mỗi Client chỉ dùng duy nhất một kết nối để download tuần tự các phần nhỏ xen kẽ của từng file từ server (Phần 2).
6. Hiển thị thanh tiến độ tải.
7. Khi Client bấm 'Ctrl + C' thì chương trình client đóng kết nối đến Server và kết thúc chương trình.
8. Khi Server bấm 'Ctrl + C' thì kết thúc phiên chạy Server và đóng kết nối tới các Client.

4.2 Các tính năng bổ sung

1. Giao diện Console cơ bản cho Server.

2. Giao diện Console cơ bản và nâng cấp cho Client.
3. Giao diện người dùng đồ họa (GUI) cho Server và Client.
4. Tự động cập nhật danh sách tài nguyên trên Server nếu có bất kì sự thay đổi nào (đổi tên file, xóa file, ...).
5. Sử dụng `command line arguments` để chạy bất kì phiên bản nào (sử dụng GUI/-Console, chạy phần 1) của Server và Client một cách thuận tiện.
   - `py server.py --part1`: Chạy Server với yêu cầu phần 1.
   - `py client.py --part1`: Chạy Client với yêu cầu phần 1.
   - `py server.py --gui`: Chạy Server với giao diện người dùng đồ họa.
   - `py client.py --gui`: Chạy Client với giao diện người dùng đồ họa.
   - `py client.py --rich`: Chạy Client với giao diện console nâng cấp.
   - Các lệnh trên có thể kết hợp với nhau mà không gây xung đột.

4.3 Các trường hợp đặc biệt khi chạy chương trình

1. Khởi tạo một Socket Server mới với địa chỉ đã được sử dụng sẽ không thành công.
2. Trong quá trình tải file, nếu Client bấm 'Ctrl + C' thì kết thúc phiên chạy của Client đó và Server cũng sẽ ngắt kết nối với Client đó.
3. Trong quá trình tải file, nếu Server bấm 'Ctrl + C' thì kết thúc phiên chạy của Server và ngắt kết nối với tất cả Client, đồng thời các file đang tải cũng sẽ dừng tải và Client kết thúc chương trình.

5 Hướng dẫn sử dụng tính năng chương trình

Chuẩn bị: Cấu hình file `.env` trước khi chạy chương trình (cấu hình cho Server và Client) và tải thư viện bổ sung bằng lệnh `pip install -r requirements.txt` (trong thư mục `Source`).

- **Server:**
  - `HOST`: Địa chỉ IP của Server
  - `PORT`: Cổng kết nối của Server
  - `SERVER_RESOURCES_PATH`: Đường dẫn chứa tài nguyên có thể tải về
- **Client:**
  - `HOST`: Địa chỉ IP của Server
  - `PORT`: Cổng kết nối của Server
  - `CLIENT_DOWNLOADS_PATH`: Đường dẫn chứa tài nguyên đã tải về
  - `CLIENT_REQUEST_INPUT`: Đường dẫn file chứa danh sách file cần tải về

Bước 1:

- Chạy file `server.py` bằng lệnh `py server.py`, lúc này một Socket Server sẽ chạy và chờ các kết nối từ Client.
- Chạy file `client.py` bằng lệnh `py client.py` khởi tạo một Socket Client và tự động kết nối đến Server.

Bước 2: Sau khi Client kết nối thành công, Server sẽ hiển thị địa chỉ của Client đã kết nối và gửi đi danh sách tài nguyên có thể tải về cho Client.

Bước 3: Nhập những file muốn tải vào `input.txt` trong đường dẫn của ứng dụng Client (mặc định ở `Source/app/client/input.txt`). Client sẽ tự động đọc (sau khi tải xong file đối với Phần 1 và mỗi 2s đối với Phần 2) và tải file trong danh sách theo mức độ ưu tiên được quy ước từ trước.

Ngoài ra còn có thể nhập lệnh bằng `console` (Client) với quy tắc giao tiếp:

- `"help", "h"`: Hiển thị danh sách các lệnh có thể dùng.
- `"quit", "q"`: Ngắt kết nối với Server và kết thúc chương trình.
- `"list", "l"`: Gửi yêu cầu lấy danh sách các file có thể tải về từ Server sau đó nhận dữ liệu và hiển thị danh sách này.
- `"file", "f"`: Tiến hành tải file từ Server dựa trên dữ liệu trong file `input.txt`.

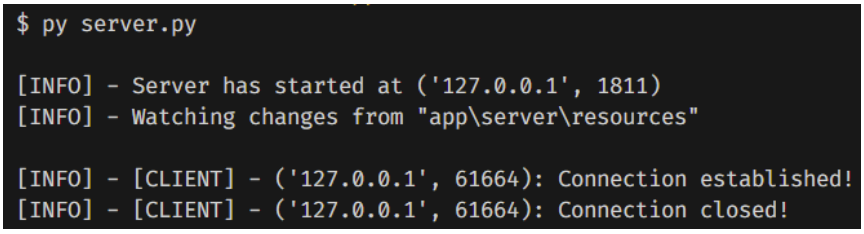
Bước 4: Ngắt kết nối bằng cách `Ctrl+C` (có thể thực hiện ở cả Server và Client) hoặc nhập lệnh `"q"` (chỉ ở Client).

6 Bảng phân công việc

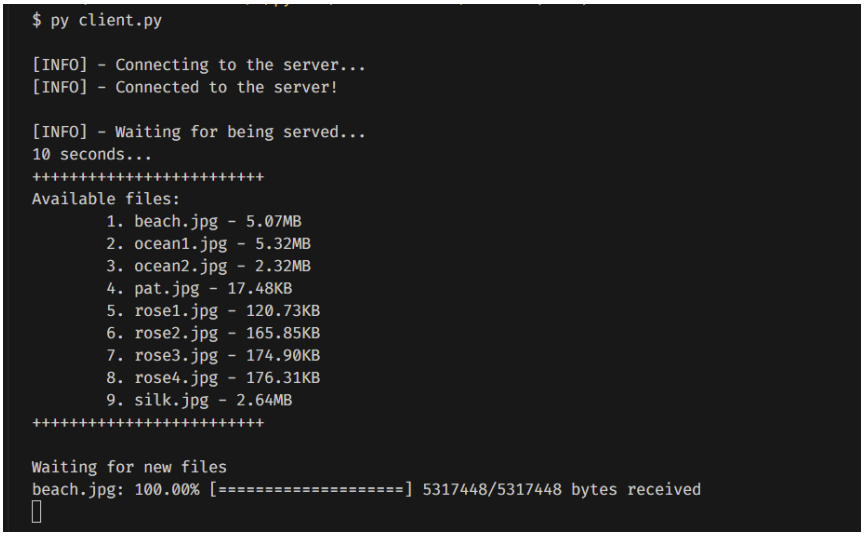
STT	Công việc	Người thực hiện
1	Viết báo cáo	Khoa, Xuân
2	Kiểm thử phần mềm (kiểm tra các tính năng và rà soát lỗi)	Xuân
3	Thiết kế mẫu giao diện	Xuân
4	Tích hợp command line interface	Xuân
5	Vẽ biểu đồ kịch bản giao tiếp giữa Client và Server	Xuân
6	Lấy dữ liệu, kích thước, đường dẫn của các tài nguyên (Server)	Xuân
7	Lấy danh sách tài nguyên và hiển thị danh sách file hiện có bên Server	Khoa
8	Lấy danh sách file cần tải từ input.txt (Client)	Xuân
9	Trích xuất thông tin từ file input.txt, xử lí độ ưu tiên tải (CRIT, HIGH, NORM)	Xuân
10	Tạo các thành phần (khung, thanh cuộn, khung chữ, check box, ...) cho giao diện đồ họa	Khoa
11	Tạo thanh tiến trình, bảng điều khiển, cấu hình khung cho giao diện	Khoa
12	Bộ đếm ngược thời gian chờ kết nối đến Server	Khoa
13	Theo dõi sự thay đổi trong thư mục 'resources' (thêm mới, chỉnh sửa file tài nguyên) và tiến hành cập nhật file `resources.json` (file chứa thông tin về các tài nguyên như tên file, dung lượng và đường dẫn)	Khoa
14	Gửi tín hiệu để xác định trạng thái kết nối (Server và Client)	Khoa
15	Xử lí thoát và dừng khi có tín hiệu tương ứng	Khoa
16	Xử lí tải file	Khoa
17	Xử lí các exception (ngoại lệ)	Khoa
18	Client nhận dữ liệu từ Server theo từng chunk	Khoa
19	Tạo giao diện (gui) cho Server và Client	Khoa
20	Server xử lí tín hiệu từ Client	Khoa

7 Ảnh chụp màn hình

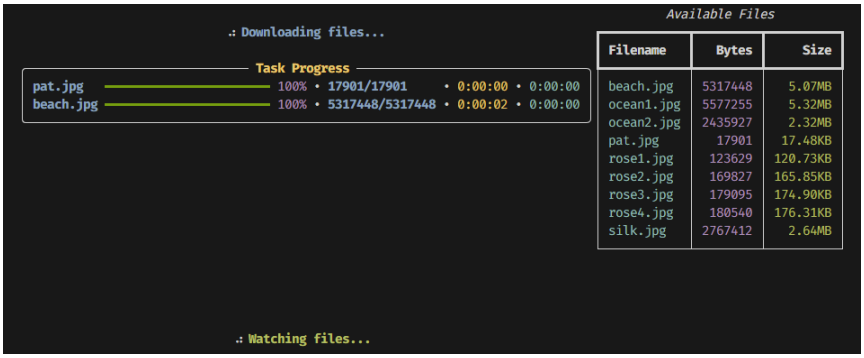
7.1 Console UI



Hình 2: Giao diện Server (Console)

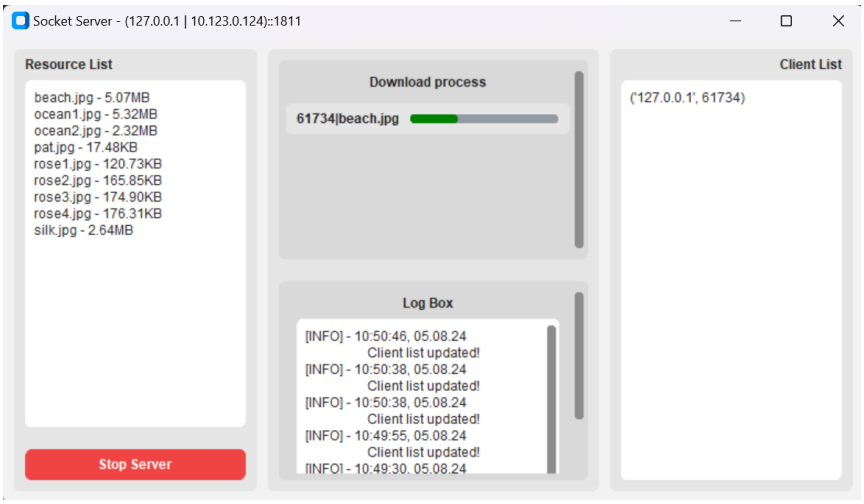


Hình 3: Giao diện Client (Console)

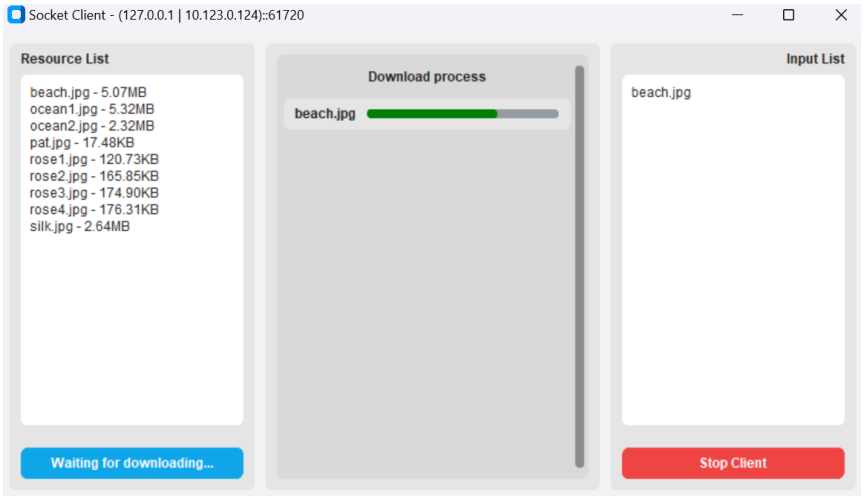


Hình 4: Giao diện Client (Rich)

7.2 Graphical User Interface (GUI)



Hình 5: Giao diện Server (GUI)



Hình 6: Giao diện Client (GUI)

8 Nguồn tham khảo

1. Python Threading
2. Python Socket Programming
3. Keyboard interrupt sockets and threads
4. Socket FTP