TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

MẠNG MÁY TÍNH

23127470 - Ngô Tấn Tài
23127469 - Hồ Thái Sơn
23127256 - Trần Thị Xuân Tài

Thành phố Hồ Chí Minh, tháng 12, năm 2024

LỜI CẢM ƠN

Lời đầu tiên, chúng em xin gửi lời cảm ơn chân thành và sự tri ân sâu sắc đối với các thầy cô trong khoa Công nghệ thông tin của Trường Đại học Khoa học Tự nhiên, Đại học Quốc gia TP.HCM. Đặc biệt, chúng em xin phép được bày tỏ tình cảm và lòng biết ơn với thầy Lê Hà Minh - Người đã dìu dắt, từng bước hướng dẫn, giúp chúng em hiểu rõ mục tiêu đồ án mình cần làm gì, đưa lời khuyên trong quá trình thực hiện.

Tuy nhiên trong học tập, cũng như trong quá trình làm đồ án không thể tránh khỏi những thiếu sót, chúng em rất mong được sự góp ý quý báu của thầy cũng như các bạn để chúng em có cái nhìn trực quan và kết quả được hoàn thiện hơn cũng như có thêm nhiều kinh nghiệm hơn cho sau này.

Một lần nữa chúng em xin chân thành cảm ơn!

MỤC LỤC

Server

Server - Kết nối TCP

- Tổng quan luồng hoạt động:
  1. Server khởi tạo socket, lắng nghe các kết nối đến.
  2. Khi một client kết nối:
     - Chấp nhận kết nối và lấy thông tin của client.
     - Tạo một luồng mới để xử lý client.
  3. Server tiếp tục lắng nghe kết nối mới trong vòng lặp.
  4. Khi server nhận tín hiệu dừng (Ctrl+C), nó đóng socket và kết thúc chương trình.
- **Chức năng tổng quan**

- **Mục đích**: Khởi động một **server TCP** lắng nghe trên một địa chỉ IP (host) và cổng (port) cụ thể.
- **Cách hoạt động**:
  - Tạo một socket TCP.
  - Liên kết socket với địa chỉ IP và cổng.
  - Đưa server vào trạng thái lắng nghe, chờ các kết nối từ client.
  - Khi có kết nối, tạo một luồng (thread) mới để xử lý từng client.

- **Chi tiết của mã**

Khởi tạo socket

```
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

- Tạo một socket TCP/IP:
  - AF_INET: Chỉ định sử dụng giao thức IPv4.
  - SOCK_STREAM: Chỉ định sử dụng giao thức TCP (kết nối, luồng dữ liệu liên tục).

Gắn socket với địa chỉ IP và cổng

```
server_socket.bind((self.host, self.port))
```

- Liên kết socket với địa chỉ IP (self.host) và cổng (self.port)

Đưa socket vào trạng thái lắng nghe

```
server_socket.listen(5)
```

- Chuyển socket sang chế độ **lắng nghe** các kết nối đến.
- 5: Số lượng kết nối tối đa được xếp hàng chờ trong hàng đợi trước khi từ chối.

Chờ và chấp nhận kết nối

```
client_socket, addr = server_socket.accept()
```

- Khi một client yêu cầu kết nối, server:
  - Tạo một socket mới (client_socket) để giao tiếp với client.
  - Lấy thông tin địa chỉ IP và cổng của client (addr).

Xử lý kết nối trong một luồng riêng

- Tạo một luồng mới cho mỗi client:
  - target=self.handle_client: Chỉ định phương thức xử lý client.
  - args=(client_socket, addr): Truyền socket và địa chỉ client vào luồng.
- **Lợi ích**:
  - Server có thể xử lý nhiều client cùng lúc mà không bị chặn.
- Bắt sự kiện **Ctrl+C** (KeyboardInterrupt) để dừng server an toàn.
- Đảm bảo socket được đóng, giải phóng tài nguyên mạng ngay cả khi có lỗi xảy ra.

Server – Xử lí đa luồng

➢ **Luồng hoạt động cơ bản**

- **Khởi tạo server**: Tạo một socket lắng nghe kết nối từ các client.
- **Chấp nhận kết nối**: Khi có kết nối đến, server chấp nhận và lấy thông tin của client.
- **Xử lý kết nối trong luồng riêng**:
  - Mỗi kết nối client được xử lý trong một luồng riêng (thread).
  - Luồng này sẽ thực hiện các tác vụ như nhận/gửi dữ liệu, xác thực người dùng, hoặc xử lý yêu cầu.

➢ **Lợi ích của xử lý đa luồng**

- **Đồng thời xử lý nhiều kết nối**:
  - Mỗi client được xử lý trong một luồng riêng.
  - Server không bị chặn khi một client thực hiện tác vụ lâu.
- **Tăng khả năng mở rộng**:
  - Phục vụ nhiều client mà không làm gián đoạn hoạt động của các client khác.
- **Hiệu quả trong ứng dụng thời gian thực**:
  - Như dịch vụ chat, truyền tệp tin, hoặc game trực tuyến.

Server – Chức năng upload

➢ **Các bước chính trong chức năng upload:**

1. **Nhận yêu cầu upload**: Server chờ nhận lệnh từ client.
2. **Kiểm tra phiên bản và xác thực người dùng**: Trước khi nhận upload, server sẽ xác nhận rằng phiên bản của client tương thích và người dùng được xác thực.

3. **Nhận dữ liệu tệp tin**: Server nhận kích thước tệp tin và checksum từ client, đồng thời chuẩn bị sẵn sàng để nhận dữ liệu.
4. **Viết tệp tin**: Sau khi nhận đủ dữ liệu, server sẽ ghi tệp tin vào ổ đĩa, tính lại checksum và so sánh với checksum mà client gửi.
5. **Kết quả upload**: Gửi thông báo về trạng thái upload ("SUCCESS", "ERROR") đến client.

➢ **Cấu trúc logic**

- **Phần nhận lệnh "UPLOAD"**:
  - Khi nhận được lệnh "UPLOAD" từ client, server xác định người dùng hiện tại (dựa vào self.active_users[address]).
  - Lấy thông tin về thư mục lưu trữ của người dùng từ self.user_manager.get_user_storage(username).
  - Tách tên tệp tin từ lệnh nhận được.
  - Gửi xác nhận "FILENAME_OK" cho client để xác nhận rằng server đã nhận tên tệp tin.
- **Phần nhận kích thước tệp tin**:
  - Server nhận kích thước của tệp tin từ client dưới dạng chuỗi và chuyển đổi thành số nguyên.
  - Gửi lại xác nhận "READY" khi sẵn sàng nhận dữ liệu tệp tin.
- **Phần nhận checksum**:
  - Nhận thông báo checksum từ client, nếu có.
  - Lấy giá trị checksum từ thông báo.
  - Gửi lại "CHECKSUM_OK" để xác nhận checksum được chấp nhận.
- **Phần nhận dữ liệu tệp tin**:
  - Server tạo đường dẫn file đầy đủ (filepath) để lưu trữ tệp tin nhận được.
  - Sử dụng recv để nhận dữ liệu tệp tin từ client.
  - Viết dữ liệu vào tệp cho đến khi nhận đủ kích thước của tệp tin đã báo trước hoặc gặp sự cố.
- **Xử lý tình huống đóng máy chủ**:
  - Kiểm tra biến self.shutdown_event trước khi nhận thêm dữ liệu từ client để đảm bảo không tiếp tục nhận nếu server đã dừng.
  - Gửi thông báo "SERVER_SHUTDOWN" nếu server bị dừng.
- **Kiểm tra toàn vẹn tệp tin**:
  - Sau khi nhận đủ tệp tin, server sẽ tính lại checksum của tệp tin đã nhận (self.generate_file_checksum(filepath)).
  - So sánh checksum tính toán được với checksum mà client gửi. Nếu không khớp, xóa tệp tin và báo lỗi "ERROR|Checksum không khớp".
  - Nếu khớp, gửi "SUCCESS" cho client và ghi lại thông tin upload thành công.

➢ **Mục đích và lợi ích của chức năng upload**

- **Toàn vẹn dữ liệu**:
  - Sử dụng checksum để đảm bảo dữ liệu được truyền đi không bị hỏng hoặc thay đổi.
  - Xác thực tính toàn vẹn của tệp tin trước khi lưu trữ.
- **Tăng cường bảo mật**:
  - Chặn các hành vi cố gắng gửi tệp tin không hợp lệ hoặc bị thay đổi.
  - Đảm bảo server chỉ nhận các tệp tin hợp lệ từ client.
- **Khả năng phục hồi**:
  - Quản lý tiến trình upload, cho phép xử lý lại nếu bị gián đoạn.
  - Tạo điều kiện để server có thể xử lý upload tệp tin theo từng phần (chunk), giảm thiểu rủi ro trong quá trình truyền tải.
- **Hiệu suất**:
  - Tận dụng multi-threading để xử lý đồng thời nhiều kết nối từ các client.
  - Tạo điều kiện để server có thể phục vụ các yêu cầu upload từ nhiều client cùng một lúc mà không gặp phải sự chờ đợi.

=> Chức năng upload trong server sử dụng giao tiếp TCP socket với khả năng xử lý đa luồng để tiếp nhận và lưu trữ các tệp tin từ các client. Server đảm bảo tính toàn vẹn của tệp tin bằng cách sử dụng checksum để xác thực dữ liệu nhận được, đồng thời cung cấp khả năng phục hồi cho các upload bị gián đoạn. Đây là một tính năng quan trọng trong các ứng dụng cần phải quản lý các tệp tin từ xa như dịch vụ lưu trữ trực tuyến, hệ thống chia sẻ dữ liệu, hoặc các dịch vụ mạng riêng tư.

Server - Chức năng download

➢ **Các bước chính trong chức năng download:**

1. **Nhận yêu cầu download**: Server chờ nhận lệnh từ client.
2. **Kiểm tra tồn tại tệp tin**: Server kiểm tra xem tệp tin có tồn tại trong thư mục của người dùng hiện tại không.
3. **Kiểm tra trạng thái sẵn sàng**: Server gửi kích thước tệp tin cho client và chờ xác nhận "READY" từ client.
4. **Kiểm tra checksum**: Tính toán và gửi checksum của tệp tin cho client, sau đó chờ xác nhận "CHECKSUM_OK" từ client.
5. **Gửi tệp tin**: Gửi tệp tin theo từng chunk cho client, cập nhật trạng thái truyền tải và gửi thông báo "SUCCESS" khi hoàn thành.

➢ **Cấu trúc logic**

- **Phần nhận lệnh "DOWNLOAD"**:
  - Khi nhận được lệnh "DOWNLOAD" từ client, server xác định người dùng hiện tại (dựa vào self.active_users[address]).

- Lấy thông tin về thư mục lưu trữ của người dùng từ self.user_manager.get_user_storage(username).
- Tách tên tệp tin từ lệnh nhận được.
- Kiểm tra định dạng lệnh, nếu không hợp lệ thì gửi thông báo lỗi "ERROR|Định dạng lệnh không hợp lệ".

- **Phần kiểm tra tồn tại tệp tin**:
  - Server kiểm tra xem tệp tin có tồn tại trong thư mục lưu trữ của người dùng hay không.
  - Nếu không tìm thấy tệp tin, gửi thông báo "FILE_NOT_FOUND" đến client.
  - Nếu tìm thấy, lấy kích thước tệp tin bằng os.path.getsize(filepath) và gửi kích thước đó cho client.
- **Phần kiểm tra trạng thái sẵn sàng**:
  - Server gửi xác nhận kích thước cho client và chờ phản hồi từ client với thông điệp "READY" nếu sẵn sàng nhận tệp tin.
  - Nếu client không phản hồi "READY", server gửi thông báo "ERROR|Không sẵn sàng nhận tệp tin".
- **Phần kiểm tra checksum**:
  - Tính toán checksum của tệp tin (self.generate_file_checksum(filepath)) và gửi cho client.
  - Server chờ phản hồi từ client với thông điệp "CHECKSUM_OK". Nếu không nhận được phản hồi này, gửi lỗi "ERROR|Client không nhận được checksum".
- **Phần gửi tệp tin**:
  - Mở tệp tin và đọc dữ liệu với kích thước 4096 bytes (chunk).
  - Gửi từng chunk dữ liệu cho client cho đến khi toàn bộ tệp tin được gửi đi.
  - Sau khi gửi tệp tin thành công, server gửi thông báo "SUCCESS" cho client.

➢ **Mục đích và lợi ích của chức năng download**

- **Toàn vẹn dữ liệu**:
  - Sử dụng checksum để đảm bảo rằng tệp tin được truyền từ server đến client không bị hỏng.
  - Tạo điều kiện để client kiểm tra tính toàn vẹn của tệp tin trước khi lưu trữ trên máy của họ.

- **Bảo mật**:
  - Kiểm tra xem client có sẵn sàng nhận tệp tin trước khi gửi.
  - Bảo vệ chống lại các lỗi truyền tải không mong muốn bằng cách sử dụng xác nhận "READY" và "CHECKSUM_OK".
- **Khả năng phục hồi**:

- Xử lý việc client không nhận được checksum hoặc tệp tin bị lỗi trong quá trình gửi.
- Gửi lại thông báo lỗi và dừng quá trình truyền tải tệp tin nếu xảy ra sự cố.

- **Hiệu suất**:
  - Gửi tệp tin theo từng chunk giúp tiết kiệm băng thông mạng và giảm thiểu thời gian chờ cho client.
  - Tạo điều kiện để server phục vụ nhiều yêu cầu download từ nhiều client cùng một lúc mà không gặp phải sự chờ đợi.

⇨ Chức năng download trong server sử dụng giao tiếp TCP socket với khả năng xử lý đa luồng để gửi các tệp tin từ server đến client. Server đảm bảo tính toàn vẹn của dữ liệu bằng cách sử dụng checksum và xác nhận từ client trước khi gửi tệp tin. Đây là một tính năng quan trọng trong các ứng dụng chia sẻ dữ liệu từ xa như dịch vụ lưu trữ trực tuyến, hệ thống chia sẻ tệp tin, hoặc các ứng dụng mạng riêng tư.

Server – Xử lí lỗi

➢ Trong từng mục cụ thể nhóm đã đề cập chi tiết đến các phương thức xử lí lỗi chi tiết.

Client

Client – Kết nối đến server

➢ **Các bước chính trong phương thức connect()**:

1. **Thiết lập socket**: Tạo socket TCP và kết nối đến server.
2. **Gửi và nhận phiên bản giao thức**: Xác minh sự tương thích giữa phiên bản giao thức của client và server.
3. **Xử lý lỗi**: Nếu xảy ra lỗi, thông báo lỗi và ngắt kết nối.

➢ **Cấu trúc logic của phương thức connect():**

- **Thiết lập socket**:
  - Tạo một socket mới với giao thức TCP (AF_INET, SOCK_STREAM).
  - Sử dụng socket.connect() để kết nối đến server theo địa chỉ (host, port).
  - Nếu kết nối thành công, socket sẽ được thiết lập để gửi và nhận dữ liệu với server.
- **Gửi phiên bản giao thức**:
  - Gửi phiên bản giao thức của client (sử dụng self.PROTOCOL_VERSION) đến server thông qua lệnh "VERSION"trước khi thiết lập kết nối.

- Sau khi gửi, nhận phản hồi từ server để kiểm tra xem server có chấp nhận phiên bản giao thức của client hay không.
- Nếu server phản hồi "VERSION_OK", kết nối được thiết lập thành công.
- Nếu không, client sẽ báo lỗi và ngắt kết nối.

- **Xử lý lỗi**:
  - Nếu xảy ra bất kỳ lỗi nào trong quá trình kết nối, client sẽ đóng socket và trả về False đồng thời ném ra ngoại lệ với mô tả lỗi chi tiết.
  - Ngoại lệ có thể là do không thể thiết lập kết nối, không tương thích phiên bản giao thức, hoặc các sự cố khác liên quan đến giao tiếp mạng.

➢ **Mục đích của phương thức connect():**

- **Đảm bảo sự ổn định và an toàn**:
  - Kiểm tra phiên bản giao thức trước khi thiết lập kết nối giúp đảm bảo rằng cả client và server sử dụng phiên bản giao thức tương thích.
  - Bảo vệ client khỏi việc kết nối với server không hỗ trợ phiên bản giao thức của mình, tránh được các vấn đề không tương thích.
- **Tăng cường trải nghiệm người dùng**:
  - Việc kiểm tra phiên bản giúp người dùng không bị mắc kẹt với các kết nối không tương thích, giữ cho trải nghiệm sử dụng luôn ổn định và đáng tin cậy.
  - Cung cấp phản hồi tức thì khi kết nối không thành công giúp người dùng có thể hành động nhanh chóng nếu cần thiết (như khởi động lại ứng dụng hoặc kiểm tra cài đặt mạng).

⇨ Phương thức connect() trong lớp FileTransferClient cung cấp khả năng kết nối an toàn và hiệu quả với server qua TCP socket. Việc kiểm tra phiên bản giao thức trước khi kết nối giúp đảm bảo tính tương thích giữa client và server, ngăn chặn các sự cố không mong muốn. Phương thức này xử lý lỗi một cách rõ ràng, giúp người dùng nhận được thông báo chi tiết về lý do kết nối không thành công nếu xảy ra lỗi.

Client – Chức năng upload qua CLI

➢ **Quá trình upload tệp tin**:
- Nhận diện tệp tin từ đường dẫn mà người dùng cung cấp.
- Tính toán thông tin của tệp tin, như kích thước và tên.
- Sử dụng progress bar để hiển thị và cập nhật tiến độ tải lên.
- Gửi tệp tin đến server qua kết nối TCP được thiết lập bởi FileTransferClient.
- Xử lý bất kỳ lỗi nào có thể xảy ra và cung cấp phản hồi hợp lý cho người dùng.

➢ **Phương thức upload_file():**

- **Nhận diện tệp tin**:

- Lấy đường dẫn tệp tin từ người dùng thông qua input().
- Kiểm tra nếu tệp tin tồn tại tại đường dẫn cho trước. Nếu không, thông báo lỗi và dừng quá trình tải lên.

- **Tính toán thông tin tệp tin**:
  - Lấy kích thước tệp tin bằng os.path.getsize().
  - Lấy tên tệp tin từ đường dẫn để sử dụng trong quá trình tải lên.
- **Gửi tệp tin đến server**:
  - Khởi tạo một progress bar sử dụng tqdm để hiển thị tiến độ tải lên cho người dùng.
  - Gửi tệp tin đến server thông qua phương thức upload_file() của FileTransferClient và cập nhật tiến trình tải lên bằng callback progress_callback.
  - Sau khi hoàn tất, đóng progress bar và thông báo cho người dùng về việc tải lên thành công.
- **Xử lý lỗi**:
  - Nếu có lỗi xảy ra trong quá trình tải lên, thông báo lỗi chi tiết đến người dùng để họ có thể biết nguyên nhân và sửa chữa.

➢ **Mục đích của phương thức upload_file():**

- **Tăng cường trải nghiệm người dùng**:
  - Cung cấp một cách dễ dàng để người dùng tải lên các tệp tin từ hệ thống máy tính của họ đến server.
  - Hiển thị tiến độ tải lên giúp người dùng biết được thời gian còn lại và lượng dữ liệu đã tải lên.
  - Thông báo lỗi chi tiết giúp người dùng dễ dàng xác định và khắc phục các vấn đề xảy ra trong quá trình tải lên.
- **Tính linh hoạt**:
  - Lớp ClientCLI cho phép người dùng thực hiện tải lên tệp tin một cách chủ động và dễ dàng thông qua dòng lệnh, mà không cần tương tác trực tiếp với mã nguồn của FileTransferClient.
  - Hỗ trợ cả việc xử lý các tệp tin lớn nhờ vào cơ chế chia nhỏ dữ liệu và cập nhật tiến trình tải lên.
- **Hỗ trợ cho nhiều loại tệp tin**:
  - Người dùng có thể tải lên bất kỳ loại tệp tin nào, và hệ thống sẽ tự động quản lý và chuyển các tệp tin đó đến server mà không cần cấu hình đặc biệt.

⇨ Phương thức upload_file() là một phần quan trọng của **ClientCLI**, cho phép người dùng thực hiện việc tải lên tệp tin một cách dễ dàng và trực quan. Với sự tích hợp của tiến trình tải lên và hiển thị lỗi chi tiết, nó giúp nâng cao trải nghiệm người dùng khi sử dụng ứng dụng truyền tải tệp tin.

Client – Chức năng download qua CLI

➢ **Quá trình tải xuống tệp tin**:

1. Nhận diện tên tệp tin từ người dùng.
2. Tạo đường dẫn lưu trữ tệp tin tải xuống.
3. Gửi yêu cầu tải xuống tệp tin đến server thông qua FileTransferClient.
4. Sử dụng callback progress_callback để cập nhật tiến trình tải xuống.
5. Hiển thị tiến độ tải xuống bằng cách cập nhật giá trị cho progress bar.
6. Sau khi hoàn tất, thông báo cho người dùng về vị trí lưu trữ tệp tin đã tải xuống.

➢ **Phương thức download_file(filename):**

- **Nhận diện tệp tin cần tải xuống**:
  - Người dùng cung cấp tên tệp tin mà họ muốn tải xuống. Tên tệp tin này sẽ được sử dụng để xác định tệp tin trên server.
  - Đường dẫn lưu trữ cho tệp tin tải xuống được tạo bằng cách sử dụng thư mục làm việc hiện tại (os.getcwd()) và tên tệp tin.
- **Hiển thị tiến độ tải xuống**:
  - Sử dụng thư viện tqdm để tạo một thanh tiến trình trực quan cho việc tải xuống tệp tin. Thanh tiến trình này giúp người dùng theo dõi được trạng thái tải xuống và ước lượng thời gian còn lại để hoàn tất.
  - Cài đặt total=100 và desc="Tiến độ" cho thanh tiến trình, giúp người dùng dễ dàng nhận biết tốc độ và phần trăm tiến độ tải xuống.
- **Gọi hàm tải xuống từ FileTransferClient**:
  - self.client.download_file() là phương thức của FileTransferClient chịu trách nhiệm kết nối với server và thực hiện việc tải xuống tệp tin.
  - Nó nhận ba tham số: tên tệp tin cần tải, đường dẫn lưu trữ trên máy tính, và một callback (self.progress_callback) để cập nhật tiến trình tải xuống.
- **Xử lý sau khi tải xong**:
  - Sau khi quá trình tải xuống hoàn tất, đóng thanh tiến trình và thông báo cho người dùng về đường dẫn nơi tệp tin đã được lưu trữ trên máy tính.
- **Xử lý lỗi**:
  - Nếu có lỗi xảy ra trong quá trình tải xuống, thông báo chi tiết lỗi cho người dùng để họ có thể khắc phục. Lỗi có thể là do kết nối mạng bị gián đoạn, lỗi trong việc nhận dữ liệu từ server, hoặc do tệp tin không tồn tại trên server.

➢ **Mục đích của phương thức download_file():**

- **Tăng cường trải nghiệm người dùng**:
  - Người dùng có thể dễ dàng tải xuống các tệp tin từ server mà không cần phải tương tác sâu với mã nguồn. Điều này giúp người

dùng có thể sử dụng ứng dụng truyền tải tệp tin mà không cần kỹ thuật chuyên sâu.
    - Tiến độ tải xuống được hiển thị giúp người dùng biết được trạng thái hiện tại và đánh giá được thời gian còn lại cho việc tải xuống.
- **Hỗ trợ cho nhiều loại tệp tin**:
    - Tương tự như chức năng upload, phương thức này cho phép người dùng tải xuống bất kỳ loại tệp tin nào từ server, từ tài liệu đến hình ảnh, video, v.v.
- **Tích hợp với FileTransferClient**:
    - download_file() sử dụng phương thức download_file() của FileTransferClient để thực hiện kết nối và trao đổi dữ liệu với server. Điều này đảm bảo tính nhất quán và bảo mật trong quá trình tải xuống.
- **Tương tác với progress bar**:
    - Sử dụng tqdm để tạo một thanh tiến trình trực quan giúp người dùng biết rõ về quá trình tải xuống. Đây là một yếu tố quan trọng trong việc tạo ra một ứng dụng dễ sử dụng.

➢ **Xử lý lỗi**:

- Cung cấp phản hồi chi tiết nếu có lỗi xảy ra trong quá trình tải xuống, giúp người dùng dễ dàng nhận diện và khắc phục sự cố.

⇨ Chức năng download_file() trong **ClientCLI** không chỉ đơn thuần là tải xuống tệp tin mà còn tích hợp nhiều tính năng tiện ích để nâng cao trải nghiệm người dùng. Với việc sử dụng progress bar để hiển thị tiến độ, nó giúp người dùng biết rõ về tốc độ tải xuống và trạng thái của việc tải xuống tệp tin từ server.

Client – Thông báo kết qủa qua CLI

➢ **Hiển thị thông báo kết quả cho từng lệnh CLI:**

- **Thông báo tải lên (upload)**:
    - Trước khi bắt đầu tải lên, người dùng sẽ được thông báo về tên và kích thước của file sẽ được tải lên.
    - Sau khi hoàn tất tải lên, nếu thành công, sẽ có thông báo "Tải lên thành công!".
    - Nếu có lỗi trong quá trình tải lên, thông báo lỗi cụ thể sẽ hiển thị như "Lỗi tải lên: [lý do]".
- **Thông báo tải xuống (download)**:
    - Trước khi bắt đầu tải xuống, thông báo cho người dùng về tên file và đường dẫn lưu trữ.
    - Trong khi tải xuống, thanh tiến trình tqdm sẽ hiển thị trạng thái tiến độ tải xuống.
    - Sau khi hoàn tất, nếu thành công, sẽ có thông báo "Đã tải xuống thành công vào: [đường dẫn lưu trữ]".

- Nếu có lỗi, như kết nối bị ngắt hoặc lỗi tải xuống không hoàn thành, sẽ có thông báo "Lỗi tải xuống: [lý do]".

- **Danh sách file (list)**:
  - Nếu không có tệp nào được tìm thấy, sẽ có thông báo "Không có file nào".
  - Nếu có tệp, danh sách các tệp sẽ được hiển thị với tên và kích thước của từng tệp.
- **Lệnh khác**:
  - Khi người dùng nhập các lệnh không hợp lệ như "help", "list", "upload", "download" không đúng định dạng, sẽ hiển thị thông báo "Lệnh không hợp lệ! Gõ 'help' để xem hướng dẫn".

➢ **Cách thức hiển thị thông báo:**

- **Sử dụng print()**:
  - Mọi thông báo kết quả đều được in trực tiếp ra màn hình người dùng bằng hàm print(). Điều này giúp dễ dàng đọc và theo dõi thông báo trong khi người dùng đang làm việc với CLI.
  - Các thông báo được in với các dòng mới để làm rõ ràng từng thông báo và tránh làm cho người dùng bị choáng ngợp với quá nhiều thông tin trên một màn hình.
- **Xử lý lỗi**:
  - Nếu có lỗi xảy ra trong quá trình thực hiện một lệnh nào đó (như lỗi kết nối, lỗi tải lên hoặc tải xuống), sẽ có thông báo lỗi chi tiết giúp người dùng dễ dàng nhận diện và khắc phục sự cố.
  - Ví dụ, nếu tải lên hoặc tải xuống không thành công, thông báo lỗi chi tiết như "Lỗi tải lên: [lý do]" hoặc "Lỗi tải xuống: [lý do]" sẽ giúp người dùng hiểu rõ nguyên nhân của sự cố.

➢ Mục đích **về cách thức thông báo qua CLI:**

- **Tăng cường trải nghiệm người dùng**:
  - Việc cung cấp thông báo trực quan giúp người dùng có thể quản lý công việc và lỗi của mình hiệu quả hơn. Điều này làm tăng tính hữu dụng và dễ sử dụng của ứng dụng CLI.
  - Người dùng sẽ không phải đoán già đoán non hoặc tìm hiểu thêm về trạng thái của các lệnh của mình vì mọi thông tin cần thiết đều được thông báo ngay lập tức.
- **Cải thiện giao tiếp với người dùng**:
  - Thông báo chính xác và kịp thời sẽ giúp người dùng cảm thấy an toàn và yên tâm hơn khi sử dụng ứng dụng. Nó không chỉ cung cấp thông tin cần thiết mà còn giúp người dùng biết rằng ứng dụng đang phản hồi đúng theo lệnh mà họ đưa ra.
  - Điều này giúp xây dựng một giao diện người dùng thân thiện, dễ sử dụng và hiệu quả.

- **Bảo mật thông tin**:
  - Các thông báo cũng giúp người dùng nhận diện và phản hồi lại các sự cố về bảo mật có thể xảy ra trong quá trình sử dụng ứng dụng, như các lỗi trong kết nối mạng, sự cố về tài khoản người dùng, hoặc các lỗi hệ thống.

Đóng kết nối

➢ **Đóng kết nối:**

- **Khi nhận được yêu cầu tắt server**:
  - Phương thức shutdown của lớp sẽ được gọi khi server nhận được lệnh tắt, chẳng hạn như từ một sự kiện KeyboardInterrupt. Khi đó, server sẽ gửi thông điệp "SERVER_SHUTDOWN" tới tất cả các khách hàng đang kết nối.
  - Sau đó, server sẽ đóng tất cả các kết nối khách hàng hiện tại bằng cách gọi phương thức client_socket.close() cho từng client.
  - Các luồng xử lý cho từng client cũng sẽ được kết thúc với phương thức thread.join() để đảm bảo rằng mọi tài nguyên và kết nối được giải phóng đúng cách trước khi server thực sự đóng.
- **Khi có lỗi hoặc ngoại lệ**:
  - Phương thức shutdown sẽ vẫn hoạt động bình thường và đóng mọi tài nguyên liên quan. Nếu có lỗi xảy ra trong khi xử lý các lệnh hoặc trong khi giao tiếp với khách hàng, server sẽ in thông báo lỗi và đóng kết nối với client bằng cách gọi client_socket.close().
  - Điều này đảm bảo rằng không có tài nguyên nào bị lãng phí và không có khách hàng nào bị kẹt lại với một kết nối không còn hoạt động.

➢ **Quá trình đóng kết nối:**

- **Thông báo cho người dùng**:
  - Khi server bắt đầu đóng, nó sẽ thông báo cho tất cả các client đang kết nối rằng server đang chuẩn bị đóng. Điều này có thể được thực hiện bằng cách gửi một thông điệp tới tất cả các client qua send_message với thông điệp "SERVER_SHUTDOWN".
  - Server sẽ in thông báo cho người dùng về việc nó đang tắt các kết nối và các luồng xử lý cho từng client.
- **Xử lý các ngoại lệ**:
  - Nếu có lỗi khi cố gắng gửi thông điệp hoặc đóng kết nối với client, server sẽ in một thông báo lỗi chi tiết:
  - Việc này giúp người quản trị dễ dàng xác định nguyên nhân của sự cố và thực hiện các hành động khắc phục nếu cần.

➢ H**iệu quả của phương pháp đóng kết nối:**

- **Tính toàn vẹn của tài nguyên**:

- Phương pháp này đảm bảo rằng tất cả các tài nguyên liên quan đến kết nối khách hàng đều được giải phóng đúng cách. Các client được thông báo về việc tắt server trước khi bị ngắt kết nối, điều này giúp tránh mất dữ liệu hoặc sự không nhất quán trong giao tiếp.
- Các luồng xử lý của từng client cũng được dừng lại đúng lúc, điều này giúp tiết kiệm tài nguyên hệ thống và tránh tình trạng quá tải của server.

- **Hiện thực dễ hiểu và dễ bảo trì**:
  - Cách thức server đóng kết nối được thực hiện qua các phương thức shutdown, send_message, và receive_message, giúp cho việc bảo trì và mở rộng hệ thống dễ dàng hơn.
  - Các thông báo lỗi được in ra giúp dễ dàng chẩn đoán và khắc phục các sự cố khi đóng kết nối.
- **Tăng cường bảo mật**:
  - Việc đóng kết nối rõ ràng và kịp thời cũng làm tăng cường bảo mật cho hệ thống. Khi server tắt, tất cả các tài khoản người dùng và các kết nối không hợp lệ đều bị xóa, làm giảm nguy cơ tấn công sau đó từ các client không còn kết nối.

Hiển thị tiến độ tải file

1.CLI

- **Sử dụng Progress Bar**:
  - **Progress Bar**: Để hiển thị tiến độ tải file, có thể sử dụng thư viện tqdm để tạo ra một thanh tiến độ đồ họa.
  - **Cập nhật tiến độ**: Trong khi đọc dữ liệu từ file hoặc gửi dữ liệu đến server, cập nhật giá trị tiến độ của thanh tqdmđể phản ánh lượng dữ liệu đã tải. Mỗi lần gửi/nhận một khối dữ liệu, tiến độ sẽ được tăng lên tương ứng.
  - **Hiển thị trạng thái**: Thanh tiến độ sẽ hiển thị tỷ lệ phần trăm của file đã được gửi hoặc nhận, cho phép người dùng dễ dàng theo dõi quá trình tải.
- **Thông Báo Kết Quả**:
  - **Sau khi kết thúc tải lên**: Kiểm tra phản hồi từ server để xác định kết quả. Nếu phản hồi từ server là "SUCCESS", báo cáo "Tải lên thành công". Nếu không, hiển thị lý do tải lên thất bại.
  - **Sau khi kết thúc tải xuống**: Tương tự, kiểm tra phản hồi từ server sau khi hoàn thành tải xuống. Nếu phản hồi là "SUCCESS", báo cáo "Tải xuống thành công". Ngược lại, hiển thị lỗi nếu có.
- **Quản Lý Lỗi**:

- Cần có cơ chế quản lý lỗi tốt để bắt và hiển thị các ngoại lệ xảy ra trong quá trình tải lên hoặc tải xuống. Điều này giúp người dùng hiểu rõ hơn về nguyên nhân của lỗi.
- Các ngoại lệ phổ biến có thể bao gồm file không tồn tại, kích thước file không hợp lệ, hoặc kết nối bị gián đoạn trong quá trình truyền tải.

➢ **Hiển Thị Thông Tin Tiến Độ**:
- Cập nhật liên tục thông tin tiến độ lên màn hình để người dùng có thể biết được bao lâu nữa quá trình tải hoặc nhận sẽ hoàn thành.
- Sử dụng thông báo trực quan như "Đang gửi dữ liệu...", "Đang nhận dữ liệu..." để tăng cường trải nghiệm người dùng.

➢ **Báo Cáo Hiệu Quả và Lỗi**:
- Khi hoàn tất quá trình tải lên hoặc tải xuống, hãy gửi thông báo chi tiết về kết quả như:
  - Kích thước dữ liệu đã tải lên/xuống.
  - Tỷ lệ phần trăm thành công.
  - Thời gian hoàn tất (nếu có thể tính toán được).
- Báo cáo về bất kỳ lỗi nào xảy ra trong quá trình này giúp cải thiện việc chẩn đoán và khắc phục sự cố của người dùng.

2.GUI

Mô tả Ý tưởng:

➢ **Sử dụng Thanh Tiến Trình (Progress Bar)**:
- Thanh tiến trình (ttk.Progressbar) trong tkinter được dùng để hiển thị trực quan tiến độ của một quá trình (như tải lên/tải xuống file).
- **Biến liên kết**: Thanh này sử dụng biến DoubleVar làm tham chiếu để theo dõi tiến độ theo tỷ lệ phần trăm (0 - 100%).
- **Cập nhật tiến độ**: Mỗi khi một phần dữ liệu được gửi hoặc nhận, giá trị của biến progress_var sẽ được cập nhật, từ đó hiển thị mức độ hoàn thành của tiến trình.

➢ **Thiết lập giao diện người dùng**:
- Thanh tiến trình được thêm vào giao diện với phương pháp .pack() để mở rộng theo chiều ngang (fill=tk.X), giúp dễ dàng theo dõi trạng thái tải file.
- **Khoảng cách và thẩm mỹ**: Sử dụng pady=5 để tạo không gian giữa các thành phần giao diện, giúp giao diện thân thiện hơn.

➢ **Cập nhật trong thời gian thực**:
- Trong quá trình tải lên hoặc tải xuống file, chương trình sẽ tính toán tỷ lệ phần trăm dữ liệu đã xử lý so với tổng kích thước file.
- Giá trị này được cập nhật trực tiếp vào progress_var, làm thay đổi trạng thái thanh tiến trình.

➢ **Tương tác với người dùng**:

- Khi bắt đầu, thanh tiến trình hiển thị từ 0%, và khi quá trình hoàn tất, thanh sẽ đạt đến 100%.
- Thanh tiến trình cung cấp phản hồi trực quan giúp người dùng hiểu được trạng thái của quá trình mà không cần chờ đợi trong sự mơ hồ.

Chi Tiết về Thanh Tiến Trình và Hiển Thị Tiến Độ:

- Giới Thiệu:
  - Trong các ứng dụng tải lên hoặc tải xuống file, việc hiển thị tiến độ thời gian thực là một yếu tố quan trọng để cải thiện trải nghiệm người dùng. Thanh tiến trình giúp người dùng dễ dàng theo dõi trạng thái và dự đoán thời gian hoàn tất.
- **Cách Hoạt Động**:
  - **Thiết lập thanh tiến trình**: Thanh tiến trình được tạo thông qua lớp ttk.Progressbar của thư viện tkinter. Nó sử dụng biến DoubleVar để theo dõi và hiển thị trạng thái tiến trình theo tỷ lệ phần trăm.
  - **Cập nhật giá trị**: Trong quá trình tải file, giá trị phần trăm tiến trình được tính bằng công thức:

    Tiến trình (%)=(Dữ liệu đã xử lí/Tổng dữ liệu)×100.

    Giá trị này được gán vào biến progress_var, từ đó tự động cập nhật giao diện của thanh tiến trình.
- **Lợi Ích**:
  - **Trực quan**: Thanh tiến trình giúp người dùng dễ dàng theo dõi tiến độ mà không cần các thông tin phức tạp.
  - **Tăng cường trải nghiệm người dùng**: Người dùng cảm thấy yên tâm hơn khi họ biết rằng hệ thống vẫn đang hoạt động và có thể ước lượng thời gian hoàn tất.
  - **Phản hồi nhanh chóng**: Trong trường hợp có sự cố (như ngắt kết nối), thanh tiến trình ngừng di chuyển, cảnh báo người dùng về vấn đề.

Phát triển thêm

GUI

Mô tả Ý Tưởng:

- **Mục tiêu giao diện**:
  - Tạo giao diện đồ họa (GUI) cho ứng dụng chuyển tệp, bao gồm chức năng đăng nhập, đăng ký, tải lên và tải xuống file từ server.
  - Hiển thị danh sách file hiện có trên server và thông tin liên quan (kích thước, ngày sửa đổi).
  - Tích hợp thanh tiến trình để hiển thị trạng thái tải file.

- **Các thành phần chính trong giao diện**:
  - **Cửa sổ đăng nhập (LoginWindow)**:
    - Cho phép người dùng nhập tên đăng nhập và mật khẩu.
    - Cung cấp các nút "Đăng Nhập" và "Tạo Tài Khoản".
  - **Cửa sổ chính (FileTransferGUI)**:
    - Hiển thị danh sách tệp tin trên server.
    - Cho phép người dùng chọn tệp tin để tải xuống hoặc chọn file từ máy tính để tải lên.
    - Cung cấp thanh tiến trình để hiển thị trạng thái tải lên/tải xuống file.
    - Có thông báo trạng thái kết nối và các hoạt động đang thực hiện.
- **Chức năng giao diện**:
  - **Đăng nhập/Đăng ký**:
    - Người dùng nhập thông tin tài khoản và kết nối tới server.
    - Hiển thị thông báo thành công hoặc lỗi khi đăng nhập.
  - **Danh sách file**:
    - Hiển thị các file trên server với thông tin:
      - Tên file.
      - Kích thước (được định dạng thành các đơn vị như KB, MB).
      - Ngày sửa đổi (hiển thị thời gian hiện tại).
  - **Tải lên/Tải xuống file**:
    - Người dùng chọn file từ máy tính để tải lên server.
    - Hoặc chọn một file từ danh sách để tải xuống máy tính.
    - Thanh tiến trình hiển thị tỷ lệ phần trăm dữ liệu đã xử lý.
- **Thiết kế thanh tiến trình**:
  - Thanh tiến trình được sử dụng để hiển thị trạng thái của quá trình tải lên hoặc tải xuống file.
  - Cập nhật giá trị theo thời gian thực, giúp người dùng theo dõi tiến độ.

<u>Chi Tiết về Giao Diện GUI:</u>

- Giới thiệu: Ứng dụng chuyển file qua giao diện đồ họa (GUI) được thiết kế để hỗ trợ người dùng dễ dàng tải file lên hoặc xuống từ server. Giao diện thân thiện và trực quan, giúp nâng cao trải nghiệm sử dụng.
- **Cấu trúc giao diện:**

1. Cửa sổ đăng nhập (LoginWindow):

- **Mục đích**: Kết nối người dùng với server thông qua tên đăng nhập và mật khẩu.
- **Thành phần**:
  - Các ô nhập (Entry) cho tên đăng nhập và mật khẩu (ẩn ký tự).
  - Nút "Đăng Nhập" để xác thực tài khoản.

- Nút "Tạo Tài Khoản" để tạo tài khoản mới.

- **Hành vi**:
  - Hiển thị thông báo thành công hoặc lỗi đăng nhập/đăng ký.
  - Gọi chức năng chính sau khi đăng nhập thành công.

2. Cửa sổ chính (FileTransferGUI):

- **Mục đích**: Thực hiện tải lên, tải xuống, và quản lý file trên server.
- **Thành phần**:
  - **Danh sách file**:
    - Hiển thị các file từ server với tên, kích thước, và ngày sửa đổi.
    - Cho phép chọn file để tải xuống.
  - **Nút chức năng**:
    - "Tải Lên" để tải file từ máy lên server.
    - "Tải Xuống" để tải file từ server về máy.
  - **Thanh tiến trình**:
    - Hiển thị tiến độ tải lên/tải xuống file.
    - Giá trị tiến trình được tính bằng tỷ lệ phần trăm dữ liệu đã xử lý.
  - **Thanh trạng thái**:
    - Hiển thị trạng thái kết nối và các thông báo quan trọng.
- **Hành vi**:
  - Tự động làm mới danh sách file sau khi tải lên thành công.
  - Hiển thị thông báo khi gặp lỗi (như kết nối thất bại hoặc file không tồn tại).

➢ **Luồng hoạt động của ứng dụng:**

1. **Kết nối tới server**:
   - Người dùng mở ứng dụng và kết nối tới server.
   - Sau khi kết nối thành công, ứng dụng chuyển đến giao diện đăng nhập.
2. **Đăng nhập/Đăng ký**:
   - Người dùng nhập thông tin tài khoản.
   - Nếu xác thực thành công, giao diện chính được hiển thị.
3. **Quản lý file**:
   - Danh sách file từ server được tải và hiển thị.
   - Người dùng chọn tải lên hoặc tải xuống file.
   - Thanh tiến trình được cập nhật theo trạng thái.

Xác thực người dùng

Mô tả Chức Năng: UserManager là một lớp quản lý người dùng sử dụng SQLite làm cơ sở dữ liệu để lưu trữ thông tin người dùng, bao gồm tên đăng nhập, mật khẩu đã băm và thư mục lưu trữ cá nhân. Dưới đây là các phương thức chính trong lớp này:

- __init__(self, db_file='users.db'):
  - Khởi tạo cơ sở dữ liệu SQLite với tên file mặc định là 'users.db'.
  - Gọi phương thức _initialize_database() để kiểm tra và tạo bảng users nếu chưa tồn tại.
- _initialize_database(self):
  - Phương thức này tạo ra bảng users với các trường id, username, password, và storage_dir.
  - Cột id là khóa chính và cột username là duy nhất.
- _hash_password(self, password):
  - Mã hóa mật khẩu bằng thuật toán SHA-256 để bảo mật.
  - Trả về mật khẩu đã băm dưới dạng chuỗi hexdigest.
- add_user(self, username, password):
  - Thêm một người dùng mới vào cơ sở dữ liệu.
  - Tạo một thư mục lưu trữ cá nhân cho người dùng mới dựa trên tên người dùng.
  - Trả về True nếu thành công, False nếu người dùng đã tồn tại.
- verify_user(self, username, password):
  - Xác thực tên đăng nhập và mật khẩu của người dùng.
  - So sánh mật khẩu đã băm với mật khẩu lưu trữ trong cơ sở dữ liệu.
  - Trả về True nếu tên đăng nhập và mật khẩu khớp với nhau, False nếu không.
- user_exists(self, username):
  - Kiểm tra xem một tên người dùng có đã tồn tại trong cơ sở dữ liệu hay không.
  - Trả về True nếu người dùng tồn tại, False nếu không.
- get_user_storage(self, username):
  - Lấy đường dẫn lưu trữ cá nhân của người dùng.
  - Trả về đường dẫn nếu tên người dùng tồn tại trong cơ sở dữ liệu, None nếu không.

Chi Tiết về Xác Thực Người Dùng:

- Mô Tả Tổng Quan: UserManager cung cấp các phương thức để quản lý người dùng, bao gồm việc thêm người dùng, xác thực tên đăng nhập và mật khẩu, kiểm tra sự tồn tại của người dùng, và truy xuất thư mục lưu trữ cá nhân của người dùng. Các phương thức này đóng vai trò quan trọng trong bảo mật của ứng dụng, đảm bảo rằng chỉ những người dùng hợp lệ mới có thể đăng nhập và thực hiện các thao tác trên hệ thống.
- Cách Thức Hoạt Động:

  1. Tạo Người Dùng:

- Phương thức add_user(self, username, password):
  - Tạo đường dẫn lưu trữ: Người dùng mới có một thư mục riêng trong thư mục user_storage dựa trên tên người dùng của họ.
  - Băm mật khẩu: Mật khẩu của người dùng được băm bằng SHA-256 trước khi lưu trữ vào cơ sở dữ liệu.
  - Thêm vào cơ sở dữ liệu: Người dùng mới được thêm vào bảng users với thông tin username, hashed_password, và storage_dir.
  - Kiểm tra lỗi: Nếu người dùng đã tồn tại, phương thức trả về False, nếu không, trả về True.

    2. Xác Thực Người Dùng:
- Phương thức verify_user(self, username, password):
  - Băm mật khẩu: Mật khẩu từ người dùng nhập vào được băm trước khi so sánh với mật khẩu đã băm trong cơ sở dữ liệu.
  - So sánh: So sánh giá trị băm của mật khẩu nhập vào với mật khẩu đã lưu trữ.
  - Trả về: Trả về True nếu xác thực thành công, False nếu không.

    3. Kiểm Tra Người Dùng Tồn Tại:
- Phương thức user_exists(self, username):
  - Truy vấn cơ sở dữ liệu: Tìm kiếm tên người dùng trong bảng users.
  - Trả về: Trả về True nếu tên người dùng tồn tại, False nếu không.

    2.4. Lấy Thư Mục Lưu Trữ:
- Phương thức get_user_storage(self, username):
  - Truy vấn cơ sở dữ liệu: Lấy đường dẫn lưu trữ từ bảng users.
  - Trả về: Đường dẫn nếu tên người dùng tồn tại, None nếu không.

➢ Ưu Điểm:

1. Bảo mật:
   - Mật khẩu người dùng được băm trước khi lưu trữ trong cơ sở dữ liệu, đảm bảo rằng ngay cả khi cơ sở dữ liệu bị rò rỉ, mật khẩu không thể dễ dàng bị khai thác.
   - Thư mục lưu trữ cá nhân giúp bảo vệ dữ liệu của từng người dùng.
2. Hiệu suất:
   - Các thao tác thêm, tìm kiếm, và kiểm tra người dùng đều rất nhanh nhờ vào cơ sở dữ liệu SQLite.
   - Sử dụng SQLite giúp tránh được sự phức tạp của quản lý cơ sở dữ liệu lớn và đắt tiền như MySQL hoặc PostgreSQL.
3. Dễ sử dụng:
   - Giao diện đơn giản và dễ hiểu, giúp cho việc tích hợp vào các ứng dụng dễ dàng hơn.

Ghi nhật kí server

- **Thiết lập Logging:**
  - Tạo thư mục chứa log (logs) nếu chưa tồn tại.
  - Tên file log được đặt dựa trên ngày giờ hiện tại, sử dụng định dạng server_ddMMyyyy_HHMMSS.log.
  - Cấu hình logging để ghi log vào cả file và console với cấp độ log từ INFO trở lên.
  - Định dạng log bao gồm: thời gian (%(asctime)s), mức độ (%(levelname)s), và thông điệp (%(message)s).
- **Ghi Log ở Các Phần Quan Trọng:**
  - **Khởi động server:** Ghi thông tin về việc server được khởi tạo, địa chỉ IP local và public.
  - **Kết nối client:** Ghi nhật ký khi có client kết nối hoặc khi kết nối bị ngắt.
  - **Quản lý người dùng:** Ghi log cho các hành động như đăng nhập, đăng ký, hay xác thực thất bại.
  - **Quản lý tệp tin:** Ghi log khi client thực hiện thao tác tải lên, tải xuống, hoặc liệt kê tệp tin.
  - **Đóng server:** Ghi log khi server ngừng hoạt động hoặc gặp sự cố bất ngờ.
- **Xử lý và Lưu Log:**
  - Tất cả các sự kiện, lỗi, và thông tin quan trọng được lưu trữ trong file log, giúp quản trị viên dễ dàng theo dõi hoạt động và xử lý sự cố.

⇨ Tổng quan ghi Nhật Ký Server

1. **Mô tả hệ thống ghi nhật ký:**

Hệ thống ghi nhật ký được thiết kế để đảm bảo việc theo dõi toàn bộ hoạt động của server, từ khởi động, quản lý kết nối, đến xử lý các lệnh từ client. Các log này được lưu trữ có tổ chức và có thể truy xuất dễ dàng.

2. Quy trình hoạt động:

- **Khởi động server:**
  - Tạo log về địa chỉ IP (local và public).
  - Lưu trạng thái khởi tạo thành công.
- **Xử lý client:**
  - Log thông tin khi có kết nối mới.
  - Ghi lại các hành động quan trọng như đăng nhập, đăng ký tài khoản, hoặc lỗi xác thực.
  - Log chi tiết khi thực hiện các lệnh như tải lên/tải xuống tệp tin.
- **Kết thúc server:**

- Log các bước đóng server, bao gồm thông báo cho client và dọn dẹp kết nối.
- Đảm bảo không có dữ liệu log bị mất bằng cách flush toàn bộ buffer.

3. Điểm nổi bật trong ghi nhật ký:

- **Log đa kênh:** Nhật ký được gửi đồng thời đến cả console và file, phù hợp cho việc giám sát trực tiếp lẫn phân tích sau này.
- **Chi tiết và dễ hiểu:** Log bao gồm thông tin chi tiết như địa chỉ IP, tên người dùng, lỗi, và trạng thái các thao tác.
- **Tính an toàn:** Các lỗi nghiêm trọng được ghi lại để hỗ trợ khắc phục sự cố hiệu quả.

4. Lợi ích của hệ thống ghi nhật ký:

- **Dễ dàng giám sát:** Quản trị viên có thể theo dõi tình trạng server theo thời gian thực hoặc qua các file log.
- **Hỗ trợ xử lý sự cố:** Log chi tiết giúp xác định nguyên nhân lỗi nhanh chóng.
- **Tăng tính bảo mật:** Theo dõi các lần đăng nhập thất bại hoặc hành động đáng ngờ từ client.