

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**MÔN: LẬP TRÌNH MẠNG**

**BÁO CÁO BÀI TẬP GIỮA KỲ**

**MÃ MÔN HỌC: NPRO430980**

**SINH VIÊN: Nguyễn Đinh Như Quỳnh – 22162038**

**Lê Trường Khoa - 22162017**

**GIẢNG VIÊN HƯỚNG DẪN: Nguyễn Đăng Quang**

**Thành phố Hồ Chí Minh, tháng 03 năm 2025**

**1. Phân công nhiệm vụ từng thành viên**

- Trường Khoa: Viết chương trình SSH Client (Client.c), kết nối tới Server, gửi thư mục và file qua giao thức SSH tự xây dựng

- Như Quỳnh: Viết chương trình SSH Server (Server.c), xử lý kết nối SSH, xác thực Public-key, nhận file/thư mục

1. **Setup môi trường thực thi trên các máy tính**

# Cập nhật danh sách gói

sudo apt update

# Cài đặt các gói cần thiết

sudo apt install -y build-essential gcc make libpthread-stubs0-dev git

# Tạo thư mục cho mã nguồn

mkdir -p ~/secure\_copy

# Tạo thư mục cho file gửi và nhận

mkdir -p ~/secure\_copy/files\_to\_send

mkdir -p ~/received\_files

# Cấp quyền thích hợp

chmod 755 ~/secure\_copy

chmod 755 ~/secure\_copy/files\_to\_send

chmod 755 ~/received\_files

# Tạo thư mục cho mã nguồn

mkdir -p ~/secure\_copy

# Tạo thư mục cho file gửi và nhận

mkdir -p ~/secure\_copy/files\_to\_send

mkdir -p ~/received\_files

# Cấp quyền thích hợp

chmod 755 ~/secure\_copy

chmod 755 ~/secure\_copy/files\_to\_send

chmod 755 ~/received\_files

1. **Giải pháp thực hiện theo tiếp cận Top-down refinement**

**Bước 1: Xác định bài toán tổng quát**

- SSH Client: Kết nối đến SSH Server và gửi file/thư mục.

- SSH Server: Nhận file/thư mục từ client và lưu vào thư mục đích.

- Xác thực bằng Public-key hoặc mật khẩu.

- Hiển thị thông tin log về quá trình truyền file.

**Bước 2: Chia nhỏ hệ thống thành các module**

- SSH Server:

+ Lắng nghe kết nối từ SSH Client.

+ Xác thực kết nối bằng Public-key/mật khẩu.

+ Nhận file từ Client qua SCP và lưu vào thư mục đích.

+ Ghi log quá trình nhận file.

- SSH Client:

+ Kết nối đến SSH Server thông qua SSH.

+ Xác thực bằng Public-key/mật khẩu.

+ Gửi file/thư mục qua SCP.

+ Hiển thị trạng thái quá trình gửi file.

**Bước 3: Chi tiết hóa từng module**

*- Module kết nối SSH (Client và Server):*

+ SSH Client:

• Tạo một phiên SSH (ssh\_new()), thiết lập thông tin máy chủ (ssh\_options\_set()).

• Thực hiện xác thực (ssh\_userauth\_publickey\_auto() hoặc ssh\_userauth\_password()).

• Nếu kết nối thành công, gửi lệnh SCP.

+ SSH Server:

* Lắng nghe kết nối từ Client (ssh\_bind\_listen()).
* Nhận thông tin xác thực.
* Kiểm tra khóa công khai hoặc mật khẩu.
* Nếu hợp lệ, cho phép Client gửi dữ liệu.

*- Module gửi và nhận file qua SCP:*

+ SSH Client:

* Duyệt thư mục local (opendir()).
* Gửi từng file qua SCP (ssh\_scp\_push\_file()).
* Ghi log trạng thái gửi file.

+ SSH Server:

* Nhận file từ SCP (ssh\_scp\_pull\_request()).
* Lưu vào thư mục đích (fopen() + fwrite()).
* Hiển thị trạng thái nhận file.

*- Module ghi log và hiển thị thông tin:*

* Ghi log thời gian gửi/nhận file (get\_current\_time()).
* Lưu log vào file server.
* Hiển thị thông báo gửi file thành công/thất bại.

1. **Các bước thực hiện code**

## Phần 1: scp\_server

### 1. Khởi động và khởi tạo

1. **Phân tích tham số dòng lệnh**
   * Đọc thư mục đích từ tham số 1
   * Đọc số port từ tham số 2 (mặc định: 2222)
2. **Hiển thị thông tin khởi động**
   * Banner "SCP Server khởi động"
   * Thời gian hiện tại
   * Tên máy chủ
   * Port lắng nghe
   * Thư mục đích
3. **Khởi tạo socket TCP**
   * Tạo socket với AF\_INET (IPv4), SOCK\_STREAM (TCP)
   * Thiết lập cờ SO\_REUSEADDR để tái sử dụng port
   * Gắn socket với địa chỉ INADDR\_ANY và port đã chỉ định
4. **Bắt đầu lắng nghe kết nối**
   * Gọi hàm listen() với backlog = 5
   * Hiển thị thông báo "Đang lắng nghe kết nối tại port..."

### 2. Xử lý kết nối client

1. **Chấp nhận kết nối mới**
   * Gọi accept() để chờ và chấp nhận kết nối
   * Lưu thông tin địa chỉ client
2. **Tạo luồng xử lý riêng**
   * Tạo cấu trúc thread\_args chứa thông tin kết nối
   * Khởi tạo thread mới với hàm handle\_client()
   * Tách thread (detach) để chạy độc lập
3. **Ghi log kết nối**
   * Hiển thị "Kết nối mới đến từ: [IP]"
   * Hiển thị thông báo xác thực thành công

### 3. Nhận file từ client

1. **Nhận thông tin số lượng file**
   * Gọi recv() để nhận số lượng file từ client
   * Ghi log "Đang nhận [số] file từ client"
2. **Vòng lặp nhận từng file**
   * Nhận thông tin file (tên, kích thước, quyền truy cập)
   * Tạo file đích tại thư mục nhận với tên tương ứng
   * Ghi log "Đang nhận file: [tên] ([kích thước])"
3. **Nhận dữ liệu file**
   * Vòng lặp đọc từng khối dữ liệu qua socket
   * Ghi dữ liệu vào file đích
   * Cập nhật biến đếm số byte đã nhận
4. **Hoàn thành nhận file**
   * Thiết lập quyền truy cập cho file nhận được
   * Đóng file
   * Ghi log "Đã nhận file: [tên] - [kích thước] ([thời gian])"
   * Tăng biến đếm tổng kích thước
5. **Thống kê kết quả**
   * Hiển thị thông báo hoàn tất
   * Hiển thị số file đã nhận và tổng kích thước

### 4. Kết thúc kết nối

1. **Đóng socket client**
   * Gọi close(client\_socket)
2. **Ghi log kết thúc kết nối**
   * Hiển thị "Kết nối kết thúc - Đã nhận [số] file thành công" hoặc "Kết nối kết thúc" nếu có lỗi

## Phần 2: scp\_client

### 1. Khởi động và khởi tạo

1. **Phân tích tham số dòng lệnh**
   * Đọc thư mục nguồn từ tham số 1
   * Đọc hostname (IP đích) từ tham số 2
   * Đọc username từ tham số 3
   * Đọc thư mục đích từ tham số 4
   * Đọc port từ tham số 5 (mặc định: 2222)
   * Đọc đường dẫn khóa từ tham số 6 (mặc định: /home/kali/.ssh/id\_ed25519\_client)
2. **Hiển thị thông tin khởi động**
   * Banner "SCP Client"
   * Thời gian hiện tại
   * Tên máy cục bộ
   * Thư mục nguồn
   * Máy đích và port
   * Tên người dùng
   * Thư mục đích
   * Đường dẫn khóa SSH
3. **Khởi tạo socket TCP**
   * Tạo socket với AF\_INET (IPv4), SOCK\_STREAM (TCP)
4. **Phân giải tên miền**
   * Chuyển đổi hostname thành địa chỉ IP
   * Sử dụng inet\_pton() hoặc gethostbyname() nếu cần

### 2. Kết nối đến server

1. **Thiết lập kết nối**
   * Gọi connect() đến địa chỉ và port của server
   * Hiển thị "Đang kết nối đến [hostname]:[port]..."
2. **Báo cáo trạng thái kết nối**
   * Hiển thị "Kết nối thành công!"
3. **Giả lập xác thực SSH**
   * Hiển thị "Đang xác thực với khóa [đường dẫn khóa]..."
   * Hiển thị "Xác thực thành công!"

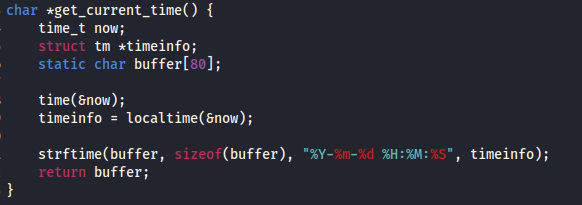
### 3. Gửi file

1. **Đếm số file trong thư mục nguồn**
   * Duyệt qua thư mục nguồn để đếm số file (không tính thư mục con)
2. **Gửi thông tin số lượng file**
   * Gửi số nguyên biểu thị số file sẽ gửi
   * Hiển thị "Chuẩn bị gửi [số] file từ [thư mục]"
3. **Vòng lặp gửi từng file**
   * Duyệt qua từng file trong thư mục nguồn
   * Bỏ qua thư mục con và các file đặc biệt (".", "..")
4. **Cho mỗi file**
   * Mở file để đọc ở chế độ nhị phân
   * Xác định kích thước file và quyền truy cập
   * Tạo cấu trúc FileInfo chứa tên, kích thước, quyền
   * Gửi thông tin FileInfo đến server
   * Hiển thị "Đang gửi file: [tên] ([kích thước])"
5. **Đọc và gửi nội dung file**
   * Vòng lặp đọc từng khối dữ liệu từ file
   * Gửi dữ liệu đến server thông qua socket
   * Cập nhật thanh tiến độ
6. **Kết thúc gửi file**
   * Hiển thị "Đã gửi file: [tên]"
   * Tăng biến đếm số file đã gửi
   * Tăng biến đếm tổng kích thước

### 4. Kết thúc kết nối

1. **Hiển thị thống kê**
   * Số file đã gửi và tổng kích thước
2. **Đóng kết nối**
   * Gọi close(client\_socket) để đóng socket
   * Hiển thị "Đã gửi [số] file thành công" hoặc thông báo lỗi
3. **Giải thích**

**\* Client:**



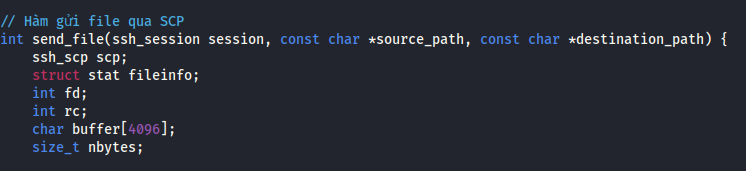
- **Hàm get\_current\_time():** Trả về thời gian hiện tại dưới dạng chuỗi "YYYY-MM-DD HH:MM:SS" để hiển thị trong log.

- Cách hoạt động:

+ Lấy thời gian hiện tại bằng time(&now).

+ Chuyển đổi thời gian thành cấu trúc tm bằng localtime(&now).

+ Định dạng thành chuỗi sử dụng strftime().



- **Hàm send\_file():** Gửi một file từ máy client lên server thông qua SCP.

- Cách hoạt động:

+ Mở file nguồn bằng open().

+ Lấy thông tin file bằng stat().

+ Tạo phiên SCP bằng ssh\_scp\_new().

+ Khởi tạo SCP bằng ssh\_scp\_init().

+ Gửi yêu cầu upload file bằng ssh\_scp\_push\_file().

+ Đọc file và gửi dữ liệu bằng read() và ssh\_scp\_write().

+ Đóng tài nguyên (fd, scp).

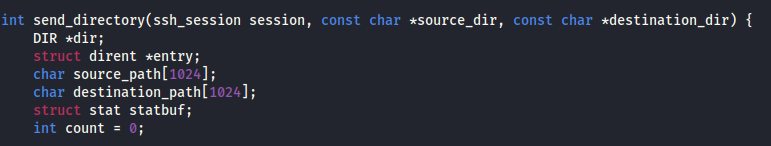
- Xử lý lỗi:

+ Nếu file không mở được => báo lỗi "Lỗi khi mở file".

+ Nếu ssh\_scp\_new() hoặc ssh\_scp\_init() thất bại => báo lỗi "Lỗi khi tạo phiên SCP".

+ Nếu ssh\_scp\_push\_file() thất bại => báo lỗi "Lỗi khi tạo request SCP".

+ Nếu ssh\_scp\_write() lỗi => báo lỗi "Lỗi khi gửi dữ liệu".



**- Hàm send\_directory():** Gửi toàn bộ file trong thư mục qua SCP.

- Cách hoạt động:

+ Mở thư mục nguồn bằng opendir().

+ Duyệt qua từng file/thư mục bằng readdir().

+ Bỏ qua . và ...

+ Lấy thông tin file/thư mục bằng stat().

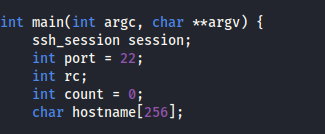
+ Nếu là file => gọi send\_file() để gửi.

+ Nếu là thư mục con => bỏ qua (hiện chưa có xử lý đệ quy).

- Xử lý lỗi:

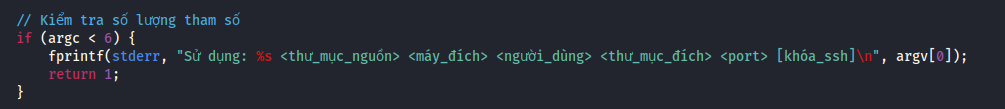
+ Nếu opendir() lỗi => báo "Lỗi khi mở thư mục".

+ Nếu stat() lỗi => báo "Lỗi khi lấy thông tin".



**- Hàm main():** nhận tham số dòng lệnh, hiển thị thông tin kết nối, khởi tạo kết nối SSH, gọi send\_directory() để gửi dữ liệu.

- Xử lý tham số dòng lệnh:



+ argv[1]: Thư mục nguồn.

+ argv[2]: Máy đích (server).

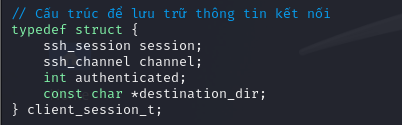
+ argv[3]: Tài khoản SSH.

+ argv[4]: Thư mục đích trên server.

+ argv[5]: Cổng SSH.

+ argv[6]: (Tuỳ chọn) Khóa SSH.

**\* Server:**



**- Cấu trúc dữ liệu client\_session\_t:** Lưu thông tin phiên làm việc của Client

+ session: Phiên SSH đang kết nối.

+ channel: Kênh giao tiếp SSH (dùng để nhận dữ liệu).

+ authenticated: Trạng thái xác thực (1 nếu xác thực thành công).

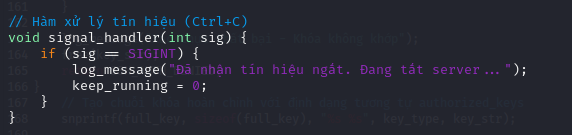
+ destination\_dir: Thư mục đích trên server để lưu file.



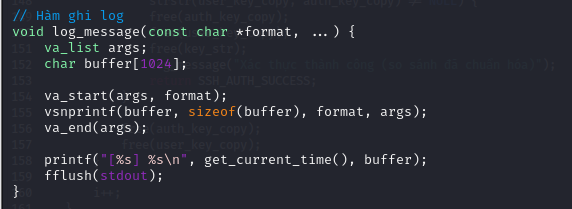
**- Chứa danh sách các khóa SSH công khai được phép kết nối:**

+ Hỗ trợ nhiều loại khóa, ví dụ RSA và Ed25519.

+ Khi một Client gửi khóa công khai, server sẽ so sánh với danh sách này.



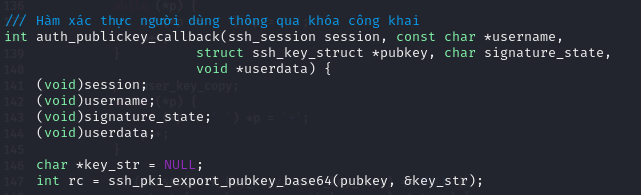
**- Xử lý tín hiệu ngắt (Ctrl+C):** Khi nhấn Ctrl+C, server sẽ nhận được tín hiệu SIGINT. Biến keep\_running = 0 để thoát vòng lặp chính, giúp tắt server an toàn.



- **Hàm ghi log (log\_message):**

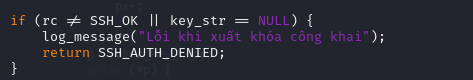
+ In ra log thông tin server với timestamp.

+ Sử dụng va\_list để hỗ trợ format string giống printf.



**- Hàm xác thực Public-key (auth\_publickey\_callback):** Được gọi khi Client gửi khóa SSH để xác thực.





- **Lấy khóa công khai của Client và kiểm tra hợp lệ**:

+ Xuất khóa công khai từ pubkey dưới dạng chuỗi Base64 để dễ so sánh.

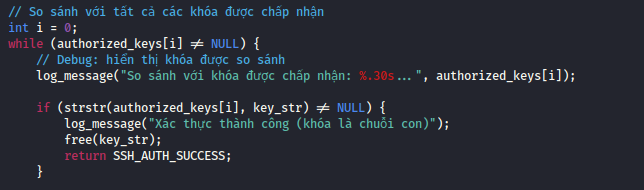
+ Nếu thất bại, trả về SSH\_AUTH\_DENIED.



**- Xác định loại khóa (RSA, Ed25519, ECDSA):**

+ Phân loại khóa dựa trên kiểu RSA, Ed25519, hoặc ECDSA.

+ Tạo chuỗi khóa đầy đủ để so sánh với danh sách authorized\_keys.

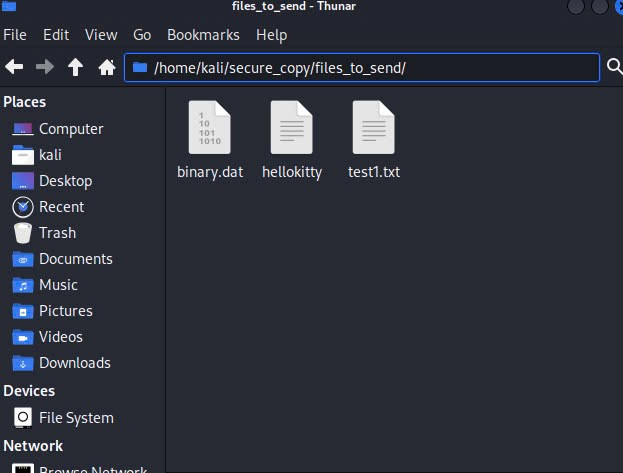


**- So sánh với danh sách khóa được chấp nhận:**

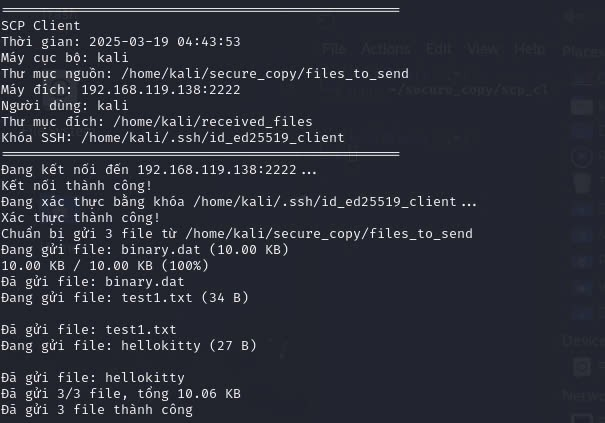
+ Kiểm tra xem khóa của Client có khớp với bất kỳ khóa nào trong danh sách không.

+ Nếu khớp, trả về SSH\_AUTH\_SUCCESS, ngược lại SSH\_AUTH\_DENIED.

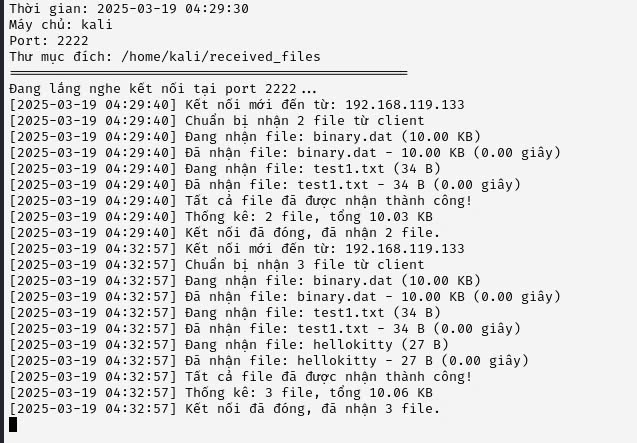
1. **Kết quả biên dịch và chạy chương trình**

****

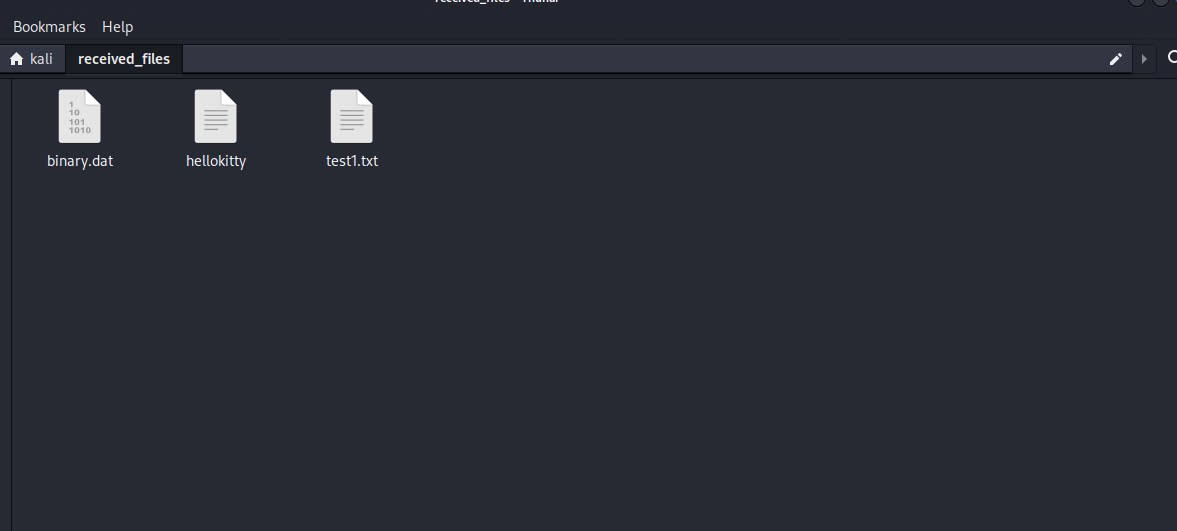
File có trong thư mục ở máy client.



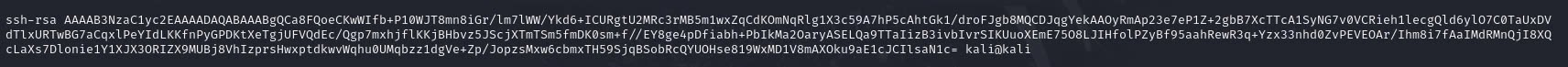
Màn hình máy client gửi file xác nhận kết nối



Màn hình hiển thị thành công bên máy server khi đã kết nối và gửi file.



Các file đã nhận bên máy server.



Khóa RSA của client



Đây là khóa ed 25519