Bài tập thực hành

Môn học: TH Lập Trình Mạng

Chương 5: I/O Multiplexing

1. MỤC TIÊU

Bài thực hành đưa ra với mục đích giúp sinh viên hiểu về cơ chế I/O Multiplexing. Sinh viên sẽ có thể sử dụng một số hàm như select(), pselect(), poll() để cho tiến trình kiểm soát nhiều đầu vào cùng lúc (standard input, nhiều socket, ...). Cụ thể, bài thực hành này sẽ giúp sinh viên phát triển một ứng dụng chat thời gian thực, đồng thời hiểu sâu hơn về lập trình socket cơ bản và các kỹ thuật nâng cao trong việc xử lý nhiều client sử dụng I/O multiplexing.

2. YÊU CẦU

- Kiến thức cơ bản về lập trình C
- Kiến thức cơ bản về Mạng máy tính
- Máy tính cài đặt Hệ điều hành Linux (khuyến khích distro Ubuntu)
- Sử dụng các hàm trong bộ thư viện liên quan lập trình mạng

3. BÀI THỰC HÀNH

Phần 1: Lập Trình Socket TCP - Thiết Lập Cơ Bản

1.1: Triển Khai Server TCP

- Mục Tiêu: Triển khai một server TCP để quản lý phòng chat và xử lý nhiều kết nối client.

- Các Bước:

1. Tạo socket server sử dụng `socket()`.

2. Gán socket server tới một địa chỉ IP và cổng cụ thể bằng `bind()`.

3. Sử dụng `listen()` để chấp nhận các kết nối client đến.

4. Chấp nhận nhiều kết nối client sử dụng `accept()`.

5. Phát các tin nhắn nhận được từ một client đến tất cả các client đang kết nối khác.

2.2: Triển Khai Client TCP

- Mục Tiêu: Triển khai một client TCP kết nối tới server phòng chat và tương tác với các client khác thông qua server.

- Các Bước:

1. Tạo socket client sử dụng `socket()`.

2. Kết nối tới server sử dụng `connect()`.

3. Gửi các tin nhắn do người dùng nhập tới server.

4. Liên tục nhận và hiển thị các tin nhắn từ các client khác, được chuyển qua server.

Kết Quả Mong Đợi:

- Server có thể xử lý nhiều client, phát tin nhắn từ một client tới tất cả các client khác trong phòng chat.

- Mỗi client có thể gửi tin nhắn và nhìn thấy tin nhắn của các client khác trong thời gian thực.

Phần 2: Logic Server và Phát Tin Nhắn

2.1: Phát Tin Nhắn Đến Tất Cả Các Client

- Mục Tiêu: Triển khai logic chính của server để nhận tin nhắn từ một client và phát chúng đến tất cả các client khác đang kết nối.

- Các Bước:

1. Khi server nhận được tin nhắn từ một client, nó sẽ chuyển tiếp tin nhắn đó tới tất cả các client khác.

2. Sử dụng hàm `send()` để phát tin nhắn tới mỗi socket client.

Nhiệm Vụ 2: Xử Lý Client Ngắt Kết Nối

- Mục Tiêu: Đảm bảo server có thể xử lý việc client ngắt kết nối một cách êm đẹp.

- Các Bước:

1. Phát hiện khi một client ngắt kết nối và xóa kết nối của họ khỏi danh sách client hoạt động.

2. Thông báo tới các client còn lại rằng một người dùng đã rời khỏi phòng chat (tùy chọn).

Phần 3: I/O Multiplexing để Xử Lý Nhiều Client

3.1: Sử Dụng `select()` để Xử Lý Nhiều Client

- Mục Tiêu: Chỉnh sửa server để xử lý nhiều client đồng thời sử dụng `select()` để thực hiện I/O multiplexing.

- Các Bước:

1. Sử dụng select() để giám sát socket server cho các kết nối đến và socket client cho các tin nhắn đến.

2. Khi một client gửi tin nhắn, server sẽ đọc nó và phát tới các client khác.

3. Xử lý các kết nối client mới và ngắt kết nối client.

3.2: Sử Dụng `pselect()` để Thực Hiện I/O Multiplexing An Toàn Với Tín Hiệu

- Mục Tiêu: Minh họa cách `pselect()` có thể được sử dụng để xử lý nhiều client trong khi xử lý tín hiệu an toàn.

- Các Bước:

1. Thiết lập xử lý tín hiệu (ví dụ: SIGINT để kết thúc server) bằng `pselect()`.

2. Đóng server an toàn khi nhận tín hiệu, đảm bảo rằng tất cả các client đều ngắt kết nối một cách đúng đắn.

3.3: Sử Dụng `poll()` để Xử Lý Nhiều Client

- Mục Tiêu: Chỉnh sửa server để sử dụng `poll()` trong việc quản lý nhiều kết nối client.

- Các Bước:

1. Sử dụng `poll()` để giám sát socket server và các socket client.

2. Thêm và xóa socket client ra khỏi mảng `pollfd` một cách động khi các client kết nối và ngắt kết nối.

3. Đảm bảo server có thể xử lý các kết nối mới và tin nhắn của client một cách hiệu quả.

Ví Dụ: Hàm quảng bá tin nhắn đến tất cả client đang kết nối

```
void broadcast_message(int sender_fd, int *client_sockets, int
num_clients, char *message) {
    for (int i = 0; i < num_clients; i++) {
        int client_fd = client_sockets[i];
        if (client_fd != sender_fd && client_fd > 0) {
            send(client_fd, message, strlen(message), 0);
        }
    }
}
```

Phần 4: Tính Năng Phòng Chat và Trải Nghiệm Người Dùng

4.1: Thêm Tên Người Dùng

- Mục Tiêu: Cho phép các client đặt tên người dùng sẽ được hiển thị cùng với tin nhắn của họ.

- Các Bước:

1. Khi kết nối, yêu cầu mỗi client cung cấp tên người dùng.

2. Gán tên người dùng vào mỗi tin nhắn mà client gửi (ví dụ: `Alice: Hello!`).

4.2: Cải Thiện Giao Diện Người Dùng

- Mục Tiêu: Nâng cao giao diện cho client để hiển thị tin nhắn.

- Các Bước:

  1. Hiển thị rõ ràng các tin nhắn đến kèm theo tên người gửi.

  2. Liên tục yêu cầu người dùng nhập tin nhắn, trong khi cũng hiển thị các tin nhắn đến từ các client khác.

4.3: Thông Báo Phòng Chat

- Mục Tiêu: Thông báo cho tất cả các client khi có người dùng mới tham gia hoặc rời khỏi phòng chat.

- Các Bước:

  1. Phát tin nhắn tới tất cả client khi có người dùng mới tham gia (ví dụ: `Alice đã tham gia phòng chat`).

  2. Tương tự, thông báo khi một người dùng ngắt kết nối.

Yêu Cầu Nộp Bài:

- Mã Nguồn: Nộp mã nguồn đầy đủ cho cả chương trình client và server.

- Trả lời trên Assignment: Viết một đoạn giải thích:

  - Cách mà các cơ chế I/O multiplexing (`select()`,`pselect()`, `poll()`) được sử dụng để quản lý nhiều client.

  - Cách mà chức năng phòng chat (phát tin nhắn, tên người dùng, v.v.) được triển khai.

  - Những khó khăn gặp phải trong quá trình triển khai và cách khắc phục.