

**LỜI NÓI ĐẦU**

Trong thời đại ngày nay, sự liên kết giữa các ứng dụng và dịch vụ trên mạng trở nên ngày càng quan trọng. Việc truyền dữ liệu hiệu quả là một yếu tố chính để đảm bảo sự kết nối này diễn ra mượt mà và ổn định. Trong bối cảnh này, việc sử dụng Socket là một giải pháp phổ biến, cho phép truyền tải dữ liệu giữa các ứng dụng một cách linh hoạt và hiệu quả.

Đề tài mà chúng tôi trình bày trong bài dự án này tập trung vào việc áp dụng Socket trong ngôn ngữ lập trình C để truyền dữ liệu đến một web server. Mục tiêu chính là tìm hiểu cách thiết lập kết nối, truyền tải dữ liệu và quản lý giao tiếp giữa một chương trình C và một web server. Qua đó, chúng tôi hy vọng trình bày một cách rõ ràng và chi tiết về quy trình này, đồng thời cung cấp cái nhìn sâu sắc về cơ chế truyền dữ liệu sử dụng Socket.

Bài dự án sẽ bắt đầu bằng một giới thiệu về lý do chọn lựa Socket và mục tiêu nghiên cứu. Sau đó, chúng tôi sẽ trình bày về kiến thức cơ bản về Socket, kiến trúc của chương trình C và web server, cũng như quy trình kết nối và truyền dữ liệu. Cuối cùng, chúng tôi sẽ đánh giá hiệu suất và ứng dụng thực tế của giải pháp mà chúng tôi đã triển khai.

Hy vọng rằng bài dự án sẽ mang lại cái nhìn sâu sắc và hữu ích về việc sử dụng Socket trong truyền dữ liệu giữa chương trình C và web server, đồng thời khuyến khích sự phát triển và tối ưu hóa trong lĩnh vực này.

**MỤC LỤC**

**Phần 1: CƠ SỞ LÝ THUYẾT**

1. Sơ lược về cơ sở lập trình........................................................................................4
   1. Khái niệm về cơ sở lập trình..............................................................................4
   2. Tổng quan về cơ sở lập trình..............................................................................4
2. Lập trình Socket trên Linux.....................................................................................5
   1. Sơ lược về Socket..............................................................................................5
   2. Mô hình Client/Server.......................................................................................7
      1. Viết chương trình phía server.................................................................7
      2. Viết chương trình phía client..................................................................7
3. Tổng quan về công nghệ chính...............................................................................8
   1. Ngôn ngữ lập trình C........................................................................................8
   2. Môi trường lập trình Linux...............................................................................9

**Phần 2: CHƯƠNG TRÌNH DEMO**

1. File server.c ..........................................................................................................10
2. File client.c ...........................................................................................................12

**Phần 3: KẾT LUẬN** ......................................................................................................14

**PHẦN 1: CƠ SỞ LẬP TRÌNH**

1. Sơ lược về Cơ Sở Lập Trình
   1. Khái niệm về Cơ Sở Lập Trình

Cơ sở lập trình là nền tảng quan trọng đối với việc xây dựng và phát triển ứng dụng máy tính. Nó đặc trưng cho bộ khung và quy tắc cơ bản mà các nhà phát triển sử dụng để thiết kế, triển khai, và duy trì phần mềm. Khái niệm về cơ sở lập trình bao gồm cả các nguyên tắc thiết kế, quy ước đặt tên, quản lý mã nguồn, và quy trình kiểm thử, tạo nên một nền tảng ổn định và dễ bảo trì.

Trong ngữ cảnh của đề tài này, cơ sở lập trình đóng vai trò quan trọng trong việc xây dựng chương trình C. Cơ sở lập trình này cung cấp môi trường phát triển mạnh mẽ và đồng nhất, giúp quản lý mã nguồn một cách hiệu quả và tối ưu hóa hiệu suất của chương trình.

* 1. Tổng quan về Cơ Sở Lập Trình

Tổng quan về cơ sở lập trình bao gồm việc hiểu rõ về ngôn ngữ lập trình, thư viện, và công cụ hỗ trợ. Trong môi trường C, nhà phát triển sử dụng các nguyên tắc cơ bản như quản lý bộ nhớ, điều kiện biên, và cấu trúc dữ liệu để xây dựng ứng dụng có hiệu suất cao.

Ngoài ra, cơ sở lập trình còn liên quan đến quá trình triển khai và quản lý mã nguồn trên hệ thống kiểm soát phiên bản (version control). Quy trình này đảm bảo tính nhất quán và theo dõi sự phát triển của mã nguồn từ thời điểm xây dựng đến thời điểm triển khai.

Tổng quan về cơ sở lập trình là bước quan trọng giúp hiểu rõ hơn về môi trường phát triển, từ đó tối ưu hóa quy trình phát triển ứng dụng và đảm bảo tính ổn định và bảo trì trong thời gian dài.

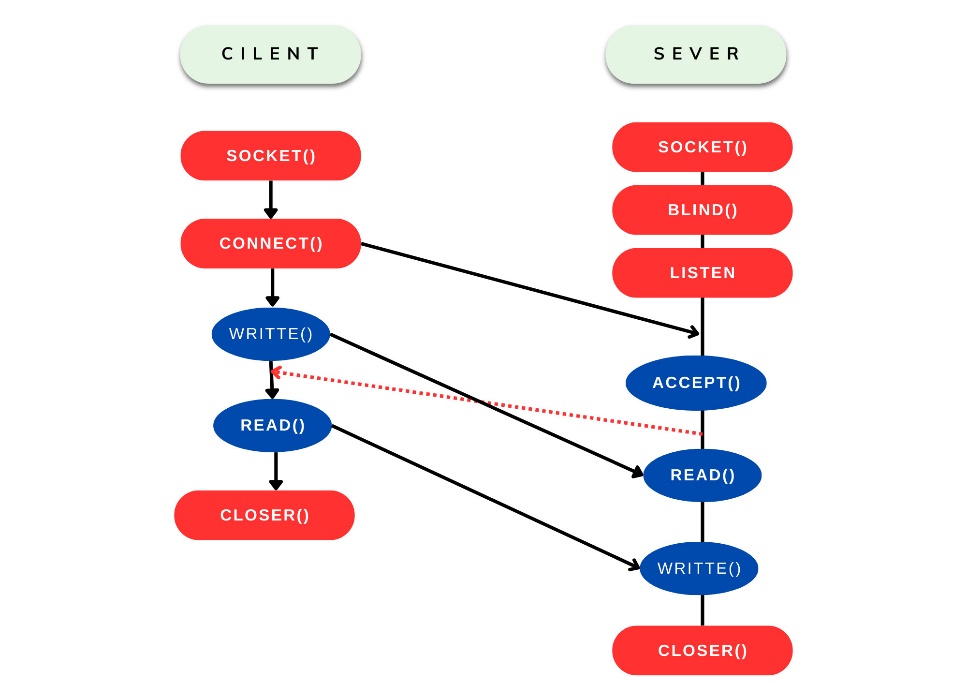
Bằng việc tìm hiểu sâu về cơ sở lập trình, chúng ta có thể xây dựng và duy trì hệ thống một cách có hệ thống và hiệu quả, điều này là quan trọng đặc biệt khi triển khai chương trình C và kết nối nó với một web server thông qua Socket.

1. Lập trình Socket trên Linux
   1. Sơ lược về Socket

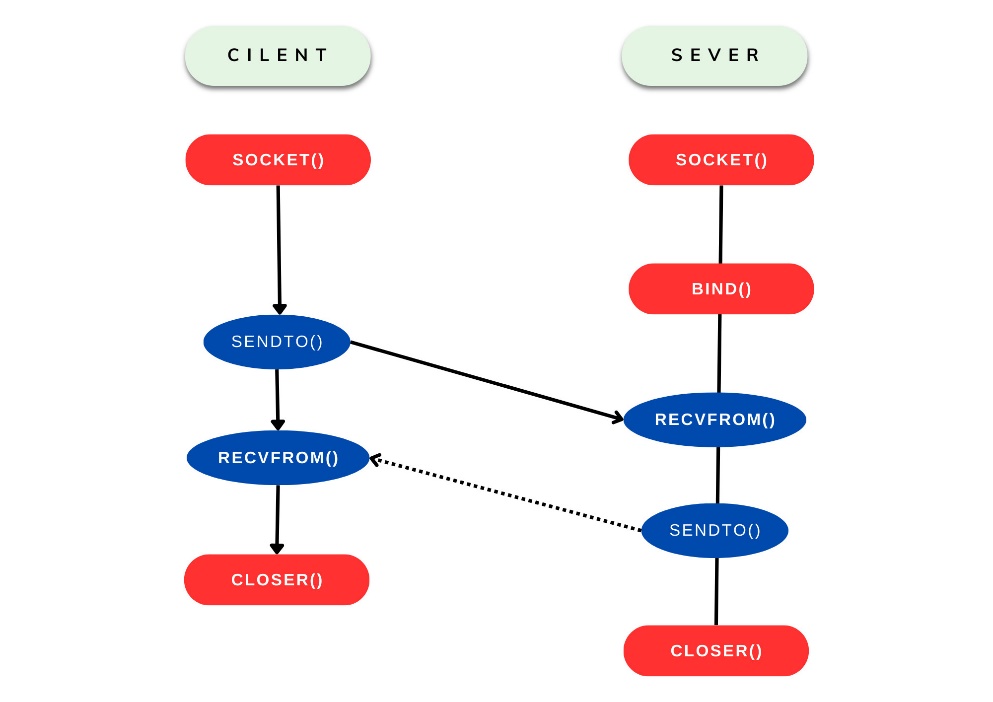
SocKet là một giao diện lập trình (API – Application Program Interface) ứng dụng mạng thông qua giao diện này có thể lập trình điều khiển việc truyền thông giữa 2 máy sử dụng các giao thức mức thấp như TCP,UDP… .Có thể tưởng tượng nó như một thiết bị truyền thông 2 chiều tương tự như tệp tin, chúng ta gửi/nhận dữ liệu giữa 2 máy, tương tự như việc đọc ghi trên tệp tin.

Có 2 loại socket được sử dụng rộng rãi là: stream sockets và datagram sockets.

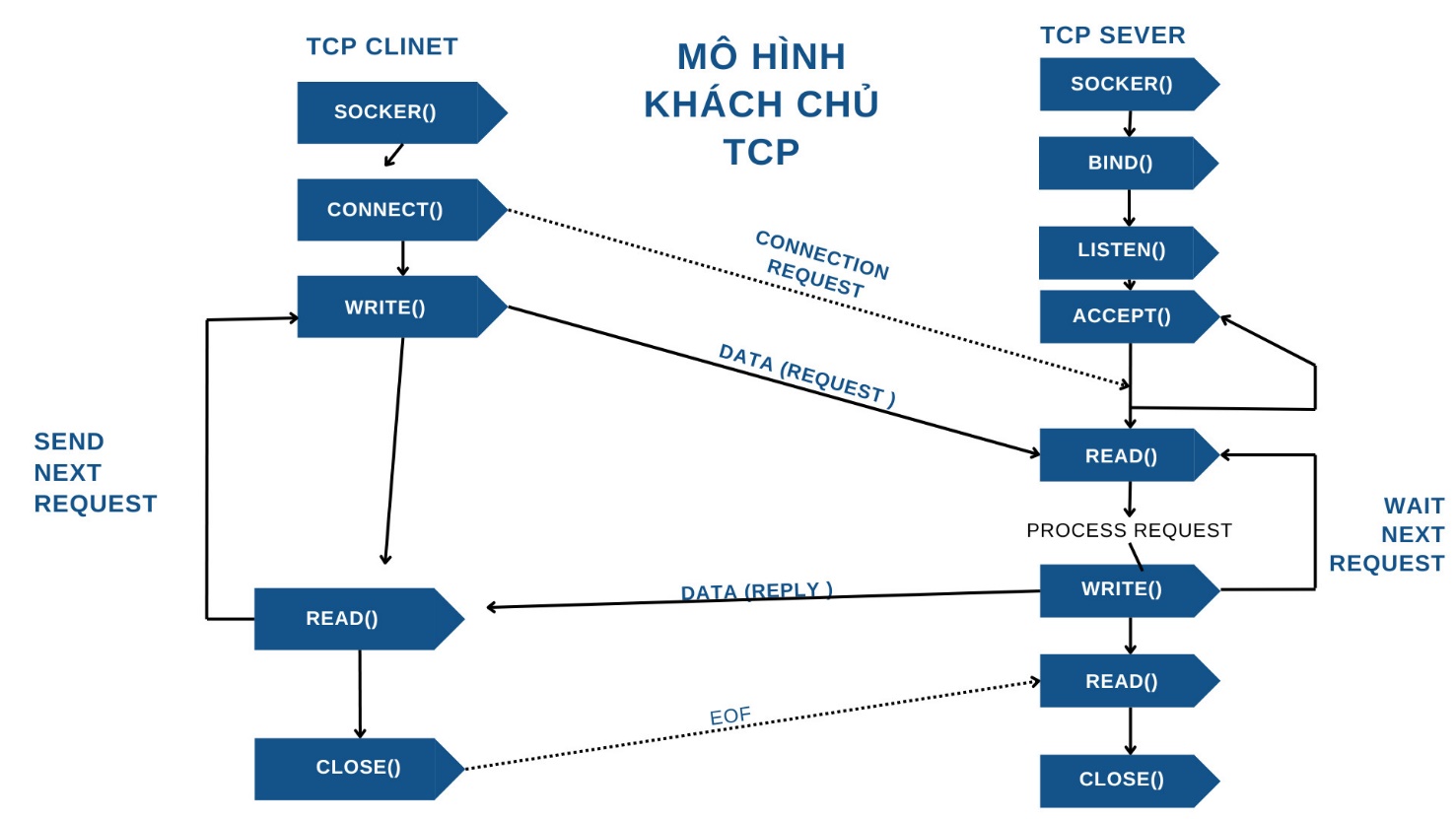
* Stream sockets: Dựa trên giao thức TCP (Tranmission Control Protocol), là giao thức hướng luồng (stream oriented). Việc truyền dữ liệu chỉ thực hiện giữa 2 tiến trình đã thiết lập kết nối. Giao thức này đảm bảo dữ liệu được truyền đến nơi nhận một cách đáng tin cậy, đúng thứ tự nhờ vào cơ chế quản lý luồng lưu thông trên mạng và cơ chế chống tắc nghẽn



* Datagram sockets: Dựa trên giao thức UDP (User Datagram Protocol), là giao thức hướng thông điệp (message oriented). Việc truyền dữ liệu không yêu cầu có sự thiết lập kết nối giữa tiến quá trình. Ngược lại với giao thức TCP thì dữ liệu được truyền theo giao thức UDP không được tin cậy, có thể không đúng trình tự và lặp lại. Tuy nhiên vì nó không yêu cầu thiết lập kết nối không phải có những cơ chế phức tạp nên tốc độ nhanh… ứng dụng cho các ứng dụng truyền dữ liệu nhanh như chat, game….



Mô hình khách chủ TCP:



* 1. Mô hình Client/Server
* Client – Server là mô hình phổ biến cho giao tiếp giữa 2 tiến trình/2 máy, trong đó, tiến trìnhclient kết nối đến tiến trình server để yêu cầu trao đổi dữ liệu.
* Client cần biết về sự tồn tại và địa chỉ của tiến trình server, nhưng server không cần biết về sựtồn tại và địa chỉ của client cho đến khi kết nối được thiết lập.
* Mỗi khi kết nối được thiết lập, cả 2 bên đều có thể trao đổi (gửi và nhận dữ liệu).
* Để thiết lập một kết nối cho cả 2 phía, cần xây dựng một socket.
* Có thể hiểu socket như một điểm đầu cuối của kênh kết nối giữa 2 tiến trình.
* Các hệ thống (linux, windows) đều cung cấp các hàm hệ thống để thực hiện thiết lập một socket.

2.2.1 Viết chương trình phía Server

* Tạo socket với hàm socket (int family, int type, int protocol) các tham số trong đó theo thứ tự là họ giao thức, kiểu socket, kiểu giao thức.
* Gán địa chỉ cho socket bind (int sockfd, const struct sockaddr\*sockaddr, socklen\_t addrlen các tham số lần lượt là mô tả socket vừa tạo, con trỏ trỏ đến địa chỉ socket, độ lớn địa chỉ
* Chỉ định socket lắng nghe kết nối listen (int sockfd, int backlog) trong đó sockfd là mô tả socket vừa tạo, backlog là số lượng tối đa các kết nối đang chờ
* Chờ/chấp nhận kết nối accept (int sockfd, struct sockaddr\*cliaddr, socklen\_t\*addrlen) lần lượt có các tham số là mô tả socket vừa tạo, con trỏ tới cấu trúc địa chỉ socket của tiến trình kết nối đến, độ lớn cấu trúc địa chỉ
* Thiết lập kết nối với máy chủ TCP connect (int sockfd, const struct sockaddr\*servaddr, socklen\_t addrlen).
  + 1. Viết chương trình phía client
* Tạo socket với hàm socket (int family, int type, int protocol) các tham số trong đó theo thứ tự là họ giao thức, kiểu socket, kiểu giao thức.
* Connect tới địa chỉ server với hàm connect (sock, (struct sockaddr\*)&serv\_addr, sizeof(serv\_addr).
* Đọc dữ liệu từ server và ghi vào biến buffer read (sock, buffer, 1024).

1. Tổng quan về công nghệ chính
   1. Ngôn ngữ lập trình C



Ngôn ngữ C được gọi là “ngôn ngữ mẹ” của ngành IT, bởi chúng được sử dụng như cơ sở, nền tảng chính dành cho những ngôn ngữ khác như: C++, C# & Java. Do đó, khi học nhuần nhuyễn ngôn ngữ này thì việc chinh phục những ngôn ngữ còn lại sẽ dễ dàng hơn rất nhiều.

Ngôn ngữ lập trình C được sử dụng để viết ra những ngôn ngữ lập trình khác như: JVMs, Kernels, C++, C#,… nhằm cung cấp chính xác các khái niệm cốt lõi, xử lý tập tin,… cực kỳ tốt. Đồng thời ngôn ngữ lập trình C còn có khả năng tạo ra hệ điều hành, các thiết bị phần cứng, hạt nhân, trình điều khiển,… trên máy tính – thiết bị điện tử vô cùng linh hoạt.

Tại sao nên sử dụng ngôn ngữ C?



* Ngôn ngữ C có tính hoạt động độc lập, đảm bảo tính ứng dụng linh hoạt cho người dùng nhờ vào khả năng thực thi nhanh chóng bởi các câu lệnh điều khiển & nhiều khái niệm khác. Bên cạnh đó, ngôn ngữ này còn có cấu trúc chia thành nhiều mô-đun nhỏ. Người dùng có thể viết riêng biệt & tạo thành một chương trình C độc lập phục vụ cho quá trình kiểm tra, gỡ lỗi & bảo trì website, ứng dụng, hệ thống,…..
* Trên hết tính năng vận hành của ngôn ngữ C giúp chúng có thể tự mở rộng thông qua các hàm khác nhau. Đáp ứng tối ưu nhu cầu sử dụng của người dùng tại nhiều hệ thống, chương trình, ứng dụng,…
* Hiện nay có rất nhiều phần mềm lập trình ra đời nhằm đáp ứng nhu cầu cho các lập trình viên. Ngoài ra, có nhiều phần mềm giúp việc học và lập trình ngôn ngữ C trở nên dễ dàng hơn, có thể kể đến như: Visual Studio Code, CLion, Visual Studio,…
  1. Môi trường lập trình Linux

Linux, là một hệ điều hành mã nguồn mở, đã trở thành một trong những môi trường phát triển ưa chuộng cho nhiều nhà phát triển. Với tính ổn định, độ bảo mật cao, và khả năng tương tác mạnh mẽ, Linux cung cấp một nền tảng lý tưởng để phát triển ứng dụng.

Môi trường lập trình Linux không chỉ hỗ trợ nhiều ngôn ngữ lập trình, mà còn cung cấp các công cụ mạnh mẽ như GCC (GNU Compiler Collection), Make, và các tiện ích quản lý mã nguồn như Git. Sự linh hoạt của Linux giúp nhà phát triển dễ dàng triển khai và quản lý chương trình C trên nền hệ điều hành này.

Việc kết hợp ngôn ngữ lập trình C và môi trường lập trình Linux trong nghiên cứu của chúng tôi không chỉ đảm bảo sự hiệu quả của chương trình mà còn mang lại trải nghiệm phát triển tích cực và linh hoạt trong quá trình triển khai và kiểm thử.

**PHẦN 2: CHƯƠNG TRÌNH DEMO**

1. **File server.c**
2. Khai báo biến và khởi tạo thông số cho server:

* **char \*ip = "127.0.0.1";**: Chuỗi địa chỉ IP mà server sẽ sử dụng.
* **int port = 8080;**: Số cổng mà server sẽ lắng nghe kết nối.
* **int e;**: Biến sử dụng để kiểm tra lỗi trả về từ các hàm

1. Khai báo các biến cần thiết cho socket và kết nối:

* **int sockfd, new\_sock;**: Số thực hiện nhiệm vụ socket cho server và socket mới sau khi chấp nhận kết nối.
* **struct sockaddr\_in server\_addr, new\_addr;**: Struct chứa thông tin về địa chỉ của server và client.
* **socklen\_t addr\_size;**: Độ dài của địa chỉ socket.

1. Tạo socket cho server và kiểm tra lỗi

* **sockfd = socket(AF\_INET, SOCK\_STREAM, 0);**: Tạo một socket sử dụng giao thức AF\_INET (IPv4) và SOCK\_STREAM (TCP).
* Kiểm tra nếu có lỗi trong quá trình tạo socket, sẽ in ra thông báo lỗi và kết thúc chương trình.

1. Thiết lập thông tin địa chỉ cho server

* **server\_addr.sin\_family = AF\_INET;**: Xác định loại giao thức (IPv4).
* **server\_addr.sin\_port = htons(port);**: Xác định cổng dịch vụ, chuyển đổi từ local byte order sang network byte order.
* **server\_addr.sin\_addr.s\_addr = inet\_addr(ip);**: Xác định địa chỉ IP.

1. Ràng buộc socket với địa chỉ và cổng

* **e = bind(sockfd, (struct sockaddr \*)&server\_addr, sizeof(server\_addr));**: Ràng buộc socket với địa chỉ và cổng đã chỉ định.
* Kiểm tra nếu có lỗi trong quá trình ràng buộc, sẽ in ra thông báo lỗi và kết thúc chương trình.

1. Lắng nghe kết nối đến socket

* **listen(sockfd, 10)**: Lắng nghe kết nối đến socket với hàng đợi có thể chứa tối đa 10 kết nối đang chờ.
* Kiểm tra nếu có lỗi trong quá trình lắng nghe, sẽ in ra thông báo lỗi và kết thúc chương trình.

1. Chấp nhận kết nối từ client và tạo socket mới

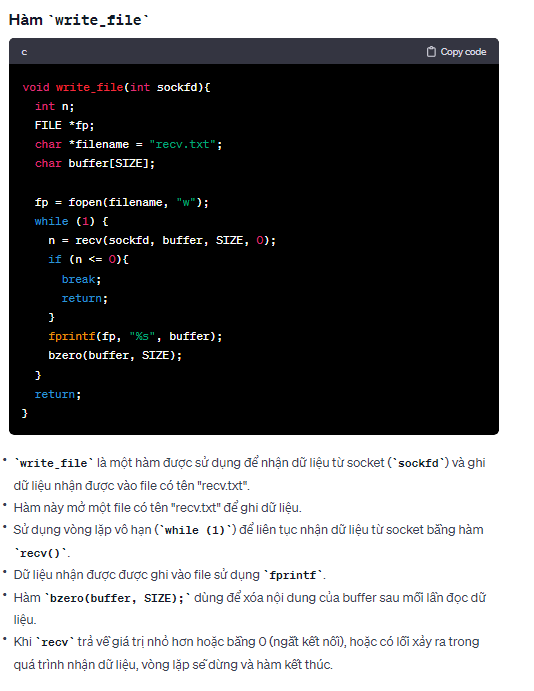
* **new\_sock = accept(sockfd, (struct sockaddr \*)&new\_addr, &addr\_size);**: Chấp nhận kết nối từ client và trả về một socket mới để trao đổi dữ liệu với client.
* Kiểm tra nếu có lỗi trong quá trình chấp nhận kết nối, sẽ in ra thông báo lỗi và kết thúc chương trình.

1. Gọi hàm write\_file để ghi dữ liệu từ client vào file

* **write\_file(new\_sock);**: Gọi hàm **write\_file** và truyền vào socket mới để nhận dữ liệu từ client và ghi vào file.
* In ra thông báo sau khi ghi dữ liệu vào file thành công.

1. Đóng socket đã chấp nhận kết nối và socket server

* **close(new\_sock);**: Đóng socket đã chấp nhận kết nối với client.
* **close(sockfd);**: Đóng socket của server.



1. **File client.c**
2. Khởi tạo biến và cấu hình thông số cho client

* **char \*ip = "127.0.0.1";**: Địa chỉ IP của server mà client sẽ kết nối tới.
* **int port = 8080;**: Số cổng mà server đang lắng nghe.
* **int e;**: Biến để kiểm tra lỗi trả về từ các hàm.

1. Khai báo các biến và struct cho việc thiết lập kết nối

* **int sockfd;**: Số thực hiện nhiệm vụ socket cho client.
* **struct sockaddr\_in server\_addr;**: Struct chứa thông tin về địa chỉ của server.
* **FILE \*fp;**: Con trỏ tới file dữ liệu sẽ được gửi đi từ client đến server.
* **char \*filename = "send.txt";**: Tên file chứa dữ liệu sẽ được gửi.

1. Tạo socket cho client và kiểm soát lỗi

* **sockfd = socket(AF\_INET, SOCK\_STREAM, 0);**: Tạo một socket sử dụng giao thức AF\_INET (IPv4) và SOCK\_STREAM (TCP).
* Kiểm tra nếu có lỗi trong quá trình tạo socket, sẽ in ra thông báo lỗi và kết thúc chương trình.

1. Thiết lập thông tin của server address (server\_addr)

* **server\_addr.sin\_family = AF\_INET;**: Xác định loại giao thức (IPv4).
* **server\_addr.sin\_port = port;**: Xác định cổng dịch vụ.
* **server\_addr.sin\_addr.s\_addr = inet\_addr(ip);**: Xác định địa chỉ IP của server.

1. Kết nối tới server và kiểm tra lỗi

* **e = connect(sockfd, (struct sockaddr\*)&server\_addr, sizeof(server\_addr));**: Kết nối tới server sử dụng hàm **connect()**.
* Kiểm tra nếu có lỗi trong quá trình kết nối, sẽ in ra thông báo lỗi và kết thúc chương trình.

1. Mở file để đọc dữ liệu và kiểm tra lỗi

* **fp = fopen(filename, "r");**: Mở file "send.txt" để đọc dữ liệu sẽ được gửi.
* Kiểm tra nếu không thể mở file, sẽ in ra thông báo lỗi và kết thúc chương trình.

1. Gửi dữ liệu từ file đến server

* Gọi hàm **send\_file(fp, sockfd);** để gửi dữ liệu từ file đến server thông qua kết nối socket đã thiết lập.
* In ra thông báo sau khi gửi dữ liệu thành công.

1. Đóng kết nối socket

* **close(sockfd);**: Đóng kết nối socket sau khi hoàn thành việc gửi dữ liệu.

A screenshot of a computer program

Description automatically generated

**PHẦN 3: KẾT LUẬN**

Trong phần lý thuyết, chúng ta đã tìm hiểu về cơ sở lập trình và lập trình Socket trên hệ điều hành Linux. Đầu tiên, chúng ta đã nắm bắt khái niệm về cơ sở lập trình, với sự tập trung vào tổng quan về cơ sở lập trình. Tiếp theo, chúng ta đã khám phá lập trình Socket, một phương tiện quan trọng để giao tiếp giữa các ứng dụng trên mạng.

Mô hình Client/Server đã được đề cập, và chúng ta đã tìm hiểu cách viết chương trình phía server và client sử dụng Socket trên hệ điều hành Linux. Qua đó, ta có cái nhìn tổng quan về cách ứng dụng có thể giao tiếp và truyền thông tin qua mạng sử dụng lập trình Socket.

Chúng ta cũng đã đề cập đến các công nghệ chính liên quan, bao gồm ngôn ngữ lập trình C và môi trường lập trình Linux. Ngôn ngữ C là một ngôn ngữ mạnh mẽ và hiệu quả cho lập trình hệ thống, trong khi Linux cung cấp môi trường ổn định và linh hoạt cho quá trình phát triển.

Ở phần thực hành, chúng ta có hai chương trình demo: "File server.c" và "File client.c". Các chương trình này thực hiện việc truyền tải tệp tin giữa server và client thông qua mô hình Client/Server sử dụng Socket. Điều này minh họa cách ứng dụng có thể sử dụng lập trình Socket để chia sẻ dữ liệu trên mạng.

Tổng kết, việc nắm vững cơ sở lý thuyết và áp dụng kiến thức thông qua các chương trình demo giúp chúng ta hiểu rõ hơn về cách lập trình Socket trên hệ điều hành Linux. Các kỹ thuật này có thể được ứng dụng rộng rãi trong phát triển ứng dụng mạng và hệ thống, đồng thời mở ra những khả năng mới trong việc tối ưu hóa giao tiếp giữa các thành phần của hệ thống.