**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**BÁO CÁO THỰC HÀNH HỆ ĐIỀU HÀNH**

**| CHỦ ĐỀ |**

**SOCKET**

**| GIẢNG VIÊN HƯỚNG DẪN |**

**Cô Vũ Thị Mỹ Hằng**

**Thầy Lê Quốc Hòa**

**| SINH VIÊN THỰC HIỆN |**

**21120302 – Huỳnh Trí Nhân**

**Thành phố Hồ Chí Minh – 2024**

**MỤC LỤC**

[**MỤC LỤC** 2](#_Toc155448762)

[1 ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH 3](#_Toc155448763)

[1.1 Đề tài 3](#_Toc155448764)

[1.2 Đánh giá mức độ hoàn thành 3](#_Toc155448765)

[2 HƯỚNG GIẢI QUYẾT PHẦN LẬP TRÌNH 4](#_Toc155448766)

[2.1 Thuật toán đề xuất 4](#_Toc155448767)

[2.1.1 Thiết kế concurrent server cho đa tiến trình khác sử dụng fork() 4](#_Toc155448768)

[2.1.2 Xử lý các phép tính toán 6](#_Toc155448769)

[2.2 Kết quả chạy chương trình 8](#_Toc155448770)

[**PHỤ LỤC** 11](#_Toc155448771)

[**TÀI LIỆU THAM KHẢO** 11](#_Toc155448772)

# ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH

## Đề tài

Viết chương trình Socket 1 server – n client thực hiện yêu cầu sau:

* Client gửi yêu cầu để Server tính toán số nguyên (chỉ cần các phép cơ bản +, -, \*, /) và trả kết quả về cho client.
* Nếu client nhập sai cú pháp thì server gửi thông báo không hợp lệ về cho client.

Ví dụ:

- Client gửi 1 + 2. Server sẽ tính toán và trả về kết quả cho client là 3

- Client gửi (1 + 2)\*3. Server sẽ tính toán và trả về kết quả cho client là 9

## Đánh giá mức độ hoàn thành

|  |  |
| --- | --- |
| Yêu cầu | Tự đánh giá |
| Khỏi tạo server | 100% |
| Khởi tạo client | 100% |
| Muti-Threading: 1 server và nhiều client | 100% |
| Giao tiếp client-server | 100% |
| Xử lý phép toán với số nguyên | 100% |
| Thực hiện chương trình trên GCC của hệ điều hành Linux | 100% |

# HƯỚNG GIẢI QUYẾT PHẦN LẬP TRÌNH

## Thuật toán đề xuất

### Thiết kế concurrent server cho đa tiến trình khác sử dụng fork()

* Lệnh gọi Fork() tạo nhiều tiến trình con cho các máy khách đồng thời và chạy từng khối lệnh gọi trong khối điều khiển tiến trình (PCB) riêng của nó.

Ảnh có chứa văn bản, đồ điện tử, ảnh chụp màn hình, màn hình

Mô tả được tạo tự độngÝ tưởng của em là thiết kết đoạn mã có thể lắng nghe liên tục các kết nối từ nhiều client khác nhau. Một client có thể giao tiếp liên tục đến với server. Do đó em đã thiết kết đoạn code code trên với những chức năng sau:

Hình Concurrent Server với fork()

1. Lắng Nghe Kết Nối Liên Tục: Server sử dụng một vòng lặp vô tận (while (1)) để liên tục lắng nghe các kết nối đến. Điều này đảm bảo rằng máy chủ luôn sẵn sàng chấp nhận kết nối mới từ các clients.
2. Chấp Nhận Kết Nối:

* Hàm accept() chờ đợi và chấp nhận một kết nối mới từ client. Nếu có lỗi, chương trình sẽ in ra thông báo lỗi và thoát.
* Khi một kết nối được chấp nhận, máy chủ in ra thông tin của kết nối đó, bao gồm địa chỉ IP và cổng của client.

1. Sử dụng fork() để tạo ra tiến trình con: Sau khi chấp nhận một kết nối, máy chủ gọi fork() để tạo một tiến trình con. Hàm fork() trả về 0 trong tiến trình con và trả về ID của tiến trình con trong tiến trình cha.
2. Xử Lý Kết Nối Trong Tiến Trình Con:

* Nếu fork() trả về 0, tức là đang ở trong tiến trình con, máy chủ đóng socket chính (server\_sockfd) vì nó không cần thiết cho việc xử lý kết nối hiện tại.
* Tiến trình con sau đó gọi hàm handle\_client() để xử lý kết nối đó, sau đó thoát.

1. Tiến Trình Cha Đóng Kết Nối:Trong tiến trình cha, sau khi fork(), socket của client được đóng (close(client\_sockfd)). Điều này không ảnh hưởng đến tiến trình con vì mỗi tiến trình có bản sao riêng của các tài nguyên hệ thống.

### Xử lý các phép tính toán

* Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

  Mô tả được tạo tự độngThuật toán em quyết định sử dụng trong bài này là thuật toán Shutting Yard. Biến biểu thức trung tố mà client nhập vào trờ thành một biểu thức hậu tố. Sau đó thực hiện tính toán trên biểu thức đó.

Hình Thuật toán Shutting Yard

* Ảnh có chứa văn bản, ảnh chụp màn hình, số, Phông chữ

  Mô tả được tạo tự độngSau khi có được biểu thức hậu tố, em tiến hành tính toán trên biểu thức đó. Tiên hành push từng số vào stack cho đến khi gặp dâu thì pop 2 giá trị của stack để tính toán rồi lại push vào stack. Cứ tiếp tục cho đến khi stack còn lại con số cuối cùng, đó chính là kết quả.

Hình Minh họa thuật toán

* Từ đó em định nghĩa hàm xử lý yêu cầu của client với các bước như sau:
  + Tạo một recv với vòng (while()) để luôn nhận được thông điện từ client
  + Sau đó trả qua hàm checkformat xem có đúng định dạng không. Nếu đúng đưa qua bước kết tiếp. Nếu không đúng thì trả về thông báo cho client.
  + Chuyển đoạn infix thành postfix bằng thuật toán đã nêu trên.
  + Tính toán trên postfix và trả về kết quả cho client
  + Cuối mỗi lần giao tiếp thì clear những buffer dữ

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Mô tả được tạo tự động

Hình Xử lý thông điệp của client

## Kết quả chạy chương trình

Ảnh có chứa ảnh chụp màn hình, Phần mềm đa phương tiện, phần mềm, Phần mềm đồ họa

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

Hình Biên dịch và khởi tạo Server

Hình Nhiều client được khỏi tạo và kết nối đến server

* Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

  Mô tả được tạo tự độngEm đã gửi phép tính lần lượt từ các client khác nhau để xem xét xem server có xử lý đồng thời được không

Hình Thực hiện nhiều yêu cầu từ client cùng lúc

* Kết quả nhận được :
  + Server kết nối được nhiều client cùng lúc cho dù giao tiếp với tiến trình khác nhưng vẫn xử lý được tác vụ từ một tiến trình khác gửi đến tức là vẫn giữ kết nối với 3 client trong suốt quả trình hoạt động.

**PHỤ LỤC**

Thư mục 21120302

* File **21120302\_client.c và 21120302\_server.c** : chứa mã nguồn chương trình em đã cài đặt.
* File **21120302.pdf** chứa báo cáo bài thực hành.
* Kết quả chạy chương trình chạy trên **hệ điều hành Linux của Ubuntu bản 20.04.6.**
* Sử dụng editor VSCode có sẵn trên Ubuntu.

**TÀI LIỆU THAM KHẢO**

* Giáo trình Hệ Điều Hành HCMUS
* Tài liệu thực hành lớp Hệ Điều Hành 21\_4
* [Design a concurrent server for handling multiple clients using fork() - GeeksforGeeks](https://www.geeksforgeeks.org/design-a-concurrent-server-for-handling-multiple-clients-using-fork/)
* [Shunting Yard (aquarchitect.github.io)](https://aquarchitect.github.io/swift-algorithm-club/Shunting%20Yard/)