ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

****

BÁO CÁO

Bài tập thực hành :

Chương 2: Cơ bản về lập trình C trên linux

Học phần: Thực hành Lập Trình Mạng

Giảng viên hướng dẫn : Trần Hải Anh

Mã lớp : 151907

Sinh viên thực hiện : Phạm Vân Anh

Mã số sinh viên : 20214988

Hà Nội, tháng 9 năm 2024

Bài 1: Xây dựng một struct để lưu thông tin của tên sv và điểm các môn cùng điểm trung bình (mean):

a. Viết chương trình và in kết quả:

```c
#include <stdio.h>
struct student{
    char name[20];
    int eng;
    int math;
    int phys;
    double mean;
};
int main(){
    static struct student data[] = {
        {"Tuan", 82, 72, 58, 0.0},
        {"Nam", 77, 82, 79, 0.0},
        {"Khanh", 52, 62, 39, 0.0},
        {"Phuong", 61, 82, 88, 0.0}
    };

    int num_students = sizeof(data) / sizeof(data[0]);
    for (int i = 0; i< num_students; i++){
    data[i].mean = (data[i].eng + data[i].math + data[i].phys) / 3.0;
    }

    printf("Diem trung binh cua sinh vien:\n");

    for (int i = 0; i < num_students; i++){
    printf("%s: %.2f\n", data[i].name, data[i].mean);
    }
    return 0;
}
```

- Kết quả

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

vaah@vaah-VirtualBox:~/Documents/lab2$ gcc student.c -o student
vaah@vaah-VirtualBox:~/Documents/lab2$ ./student
Diem trung binh cua sinh vien:
Tuan: 70.67
Nam: 79.33
Khanh: 51.00
Phuong: 77.00
vaah@vaah-VirtualBox:~/Documents/lab2$
```

b. Chương trình để tính điểm trung bình của từng SV và hiện lên hạng của SV đó.

rank_average.c

```c
#include <stdio.h>
struct student{
        char name[20];
        int eng;
        int math;
        int phys;
        char grade;
        double mean;
};
char grade(double mean){
    if (mean >= 90) return 'S';
    if (mean >= 80) return 'A';
    if (mean >= 70) return 'B';
    if (mean >= 60) return 'C';
    return 'D';
}

int main(){
    static struct student  data[] = {
        {"Tuan", 82, 72, 58, 0.0},
        {"Nam", 77, 82, 79, 0.0},
        {"Khanh", 52, 62, 39, 0.0},
        {"Phuong", 61, 82, 88, 0.0}
    };

    int num_students = sizeof(data) / sizeof(data[0]);
    for (int i = 0; i< num_students; i++){
        data[i].mean = (data[i].eng + data[i].math + data[i].phys) / 3.0;
        data[i].grade = grade(data[i].mean);
    }

    printf("Diem trung binh va xep hang cua sinh vien \n");
    for (int i = 0; i < num_students; i++){
        printf ("%s: %.2f, Grade: %c\n", data[i].name,
                                         data[i].mean,
                                         data[i].grade);
    }
    return 0;
}
```

- Kết quả:

```
vaah@vaah-VirtualBox:~/Documents/lab2$ gcc rank_average.c -o rank_average
vaah@vaah-VirtualBox:~/Documents/lab2$ ./rank_average
Diem trung binh va xep hang cua sinh vien
Tuan: 70.67, Grade: B
Nam: 79.33, Grade: B
Khanh: 51.00, Grade: D
Phuong: 77.00, Grade: B
vaah@vaah-VirtualBox:~/Documents/lab2$
```

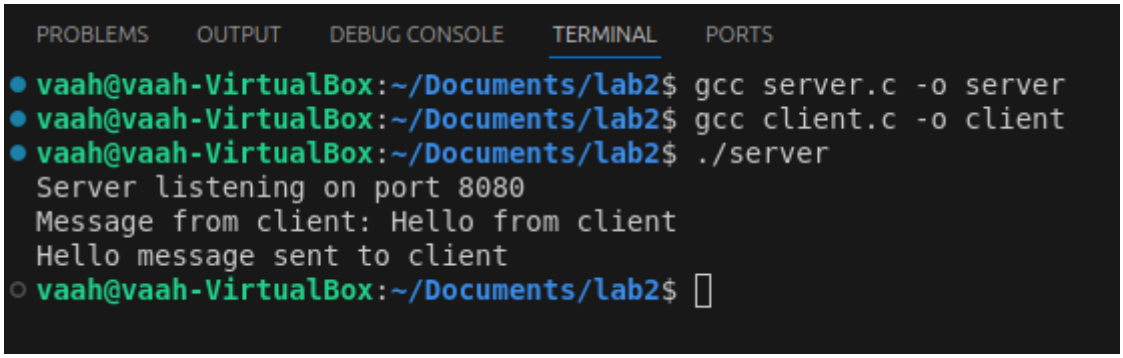
c. Sử dụng typedef để định nghĩa kiểu STUDENT, sử dụng con trỏ p để duyệt các thành phần trong mảng data

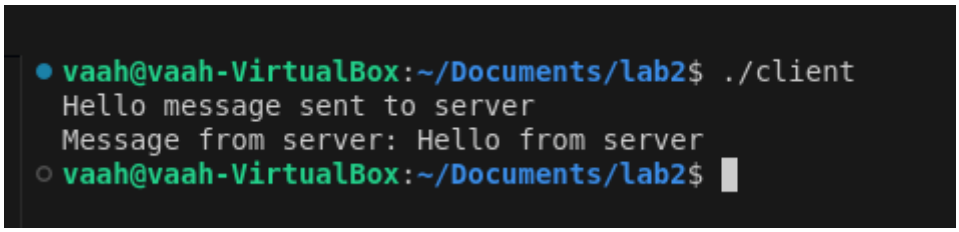
```c
#include <stdio.h>

typedef struct student{
    char name[20];
    int eng;
    int math;
    int phys;
    double mean;
} STUDENT;

int main(){
    static struct student data[]={
    {"Tuan", 82, 72, 58, 0.0},
    {"Nam", 77, 82, 79, 0.0},
    {"Khanh", 52, 62, 39, 0.0},
    {"Phuong", 61, 82, 88, 0.0}
    };

    STUDENT *p;
    int num_students = sizeof(data)/sizeof(data[0]);
    for (p = data; p < data + num_students; p++){
        p->mean = (p->eng + p->math + p->phys)/3.0;
    }

    printf("Diem trung binh cua sinh vien:\n");
    for (p = data; p < data + num_students; p++){
        printf ("%s: %.2f\n", p->name, p->mean);
    }

    return 0;
}
```

- Kết quả:

```
vaah@vaah-VirtualBox:~/Documents/lab2$ gcc typedef_student.c -o typedef_student
vaah@vaah-VirtualBox:~/Documents/lab2$ ./typedef_student
Diem trung binh cua sinh vien:
Tuan: 70.67
Nam: 79.33
Khanh: 51.00
Phuong: 77.00
vaah@vaah-VirtualBox:~/Documents/lab2$
```

Bài 2: Tải 2 tệp server.c và client.c, dịch để ra chương trình chạy





Câu hỏi 2: Trình bày nhiệm vụ của từng chương trình server và client

Server:

1. Khởi tạo Socket:
- Server tạo một socket sử dụng hàm socket(). Socket này sử dụng IPv4 (AF_INET), kiểu dữ liệu truyền là stream (SOCK_STREAM tức TCP), và giao thức mặc định (0).
2. Gán địa chỉ (Bind):
- Địa chỉ IP (INADDR_ANY) và port (PORT, 8080 trong trường hợp này) được gán cho socket. INADDR_ANY cho phép socket nhận kết nối từ bất kỳ địa chỉ IP nào.
3. Lắng nghe (Listen):
- Server đặt socket vào trạng thái lắng nghe, nơi nó có thể chấp nhận kết nối đến listen() được sử dụng để thiết lập hàng đợi kết nối với độ dài tối đa là 3 kết nối đang chờ xử lý.
4. Chấp nhận Kết nối (Accept):
- Khi có kết nối đến, server chấp nhận kết nối đó bằng cách sử dụng hàm accept().
- Kết nối này được xử lý trên một socket mới (new_socket) để cho phép server tiếp tục lắng nghe trên socket chính (server_fd).
5. Đọc và Xử lý Dữ liệu:
- Server đọc dữ liệu gửi từ client qua socket mới và in ra màn hình.
6. Gửi Phản hồi

- Server gửi lại một thông điệp "Hello from server" cho client.

7. Đóng Socket:

- Cuối cùng, socket cho kết nối hiện tại (new_socket) và socket lắng nghe chính (server_fd) được đóng lại.

Client

1. Khởi tạo Socket:
- Tương tự như server, client cũng tạo một socket.
2. Thiết lập địa chỉ Server:
- Địa chỉ và port của server được thiết lập, và địa chỉ IP được chuyển từ dạng văn bản ("127.0.0.1" tức localhost) sang dạng nhị phân bằng inet_pton().
3. Kết nối tới Server:
- Client sử dụng hàm connect() để thiết lập kết nối tới địa chỉ server đã được chỉ định.
4. Gửi Dữ liệu:
- Client gửi thông điệp "Hello from client" tới server.
5. Nhận Phản hồi:
- Client đọc và in ra thông điệp phản hồi từ server.
6. Đóng Socket:
- Sau khi hoàn thành trao đổi, socket client được đóng lại.

➔ Cả hai chương trình cùng hoạt động trên cùng một máy (localhost) và cùng port 8080, cho phép chúng giao tiếp với nhau qua mạng local. Đoạn mã này mô tả một kịch bản cơ bản của clientserver interaction sử dụng TCP, phổ biến trong nhiều ứng dụng mạng.

Bài 3:

Kết quả:

```
vaah@vaah-VirtualBox:~/Documents/lab2$ gcc capitalize_server.c -o capitalize_
server
vaah@vaah-VirtualBox:~/Documents/lab2$ gcc capitalize_client.c -o capitalize_
client
vaah@vaah-VirtualBox:~/Documents/lab2$ ./capitalize_server
Server listening on port8080...
Message from client:  happy birthday
Message sent to client: HAPPY BIRTHDAY
vaah@vaah-VirtualBox:~/Documents/lab2$
```

```
vaah@vaah-VirtualBox:~/Documents/lab2$ ./capitalize_client
Enter a string: happy birthday
Message sent to server.
Message received from server: HAPPY BIRTHDAY
vaah@vaah-VirtualBox:~/Documents/lab2$
```

Giải thích:

- Server:

```c
// Accept a connection from the client
if ((new_socket = accept(server_fd, (struct sockaddr *)&address, (socklen_t*)&addrlen)) < 0) {
    perror("accept");
    exit(EXIT_FAILURE);
}

// Read data sent by the client
int valread = read(new_socket, buffer, BUFFER_SIZE);
printf("Message from client:  %s\n", buffer);


capitalize(buffer);

// Send a response to the client
send(new_socket, buffer, strlen(buffer), 0);
printf("Message sent to client: %s\n", buffer);

// close
close(new_socket);
close(server_fd);
```

- **accept(server_fd, (struct sockaddr *)&address, (socklen_t*) &addrlen)**: Hàm accept() chờ đợi cho đến khi có một kết nối đến từ client. Nếu có kết nối thành công, nó sẽ trả về một file descriptor mới (new_socket) dùng để giao tiếp với client đã kết nối đó. Địa chỉ của client được lưu trong address, và kích thước của cấu trúc địa chỉ được lưu trong addrlen. Nếu accept() thất bại (trả về giá trị nhỏ hơn 0), perror("Accept failed") sẽ được gọi để in thông báo lỗi ra màn hình, và chương trình kết thúc với exit(EXIT_FAILURE).

- **read(new_socket, buffer, BUFFER_SIZE)**: Đọc dữ liệu gửi đến từ client thông qua socket mới (new_socket). Dữ liệu được lưu vào biến buffer, và kích thước tối đa của dữ liệu có thể đọc được là BUFFER_SIZE.
- **printf("Message received from client: %s\n", buffer):** In ra màn hình thông điệp đã nhận được từ client.
- **capitalize(buffer):** Hàm capitalize được gọi để chuyển đổi tất cả các ký tự trong buffer thành chữ hoa. Hàm này thay đổi trực tiếp nội dung của buffer.
- **send(new_socket, buffer, strlen(buffer), 0):** Gửi lại dữ liệu đã được viết hoa trong buffer cho client qua new_socket. strlen(buffer) tính độ dài của chuỗi để chỉ gửi đúng số byte chứa dữ liệu.
- **close(new_socket):** Sau khi đã gửi phản hồi cho client, socket dùng để giao tiếp với client đó được đóng lại, giải phóng tài nguyên liên quan và cho phép server chấp nhận kết nối mới từ các client khác.

- **Client**

```
37      // Enter string:
38      printf("Enter a string: ");
39      fgets(buffer, BUFFER_SIZE, stdin);
40      buffer[strcspn(buffer, "\n")] = 0;
41
42      //send stirng to server
43      send(sock, buffer, strlen(buffer), 0);
44      printf("Message sent to server.\n");
45
46      // enter string
47      int valread = read(sock, buffer, BUFFER_SIZE);
48      buffer[valread] = '\0';
49
50      printf("Message received from server: %s\n", buffer);
51
52      // Đóng socket
53      close(sock);
```

- **printf("Enter a string: "):** Hiển thị một thông báo trên màn hình yêu cầu người dùng nhập một chuỗi ký tự.
- **fgets(message, BUFFER_SIZE, stdin):** Hàm fgets được sử dụng để đọc một dòng văn bản từ chuẩn nhập (bàn phím), lưu vào biến message. BUFFER_SIZE định nghĩa giới hạn số ký tự tối đa có thể đọc, để tránh

tràn bộ đệm. Hàm này đọc đến khi gặp ký tự xuống dòng hoặc đến giới hạn số ký tự.

- **message[strcspn(message, "\n")] = 0:** Loại bỏ ký tự xuống dòng (\n) khỏi cuối chuỗi, nếu có. Hàm strcspn trả về vị trí đầu tiên của ký tự xuống dòng trong chuỗi message, và biểu thức này thay thế ký tự đó bằng ký tự kết thúc chuỗi (\0), từ đó chuẩn hóa chuỗi để chuẩn bị gửi tới server.
- **send(sock, message, strlen(message), 0):** Gửi chuỗi đến server qua socket sock. strlen(message) tính độ dài của chuỗi để chỉ gửi đúng số byte cần thiết, và 0 là các cờ điều khiển (không có cờ nào được sử dụng ở đây).
- **read(sock, buffer, BUFFER_SIZE):** Đọc nhận từ server mà server sau khi xử lý chuỗi nhận được. Dữ liệu được lưu vào biến buffer và không quá BUFFER_SIZE ký tự.
- **printf("Message received from server: %s\n", buffer):** In chuỗi nhận từ server ra màn hình.