Báo cáo tuần 7

Họ và tên: Lưu Tuấn Hùng

MSSV: 20225131

Nội dung báo cáo

Server: Nhiệm vụ

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#include <sys/select.h>

#include <errno.h>

#include <signal.h>

#include <stdbool.h>

#define PORT 8080

#define MAX\_CLIENTS 10

#define BUFFER\_SIZE 1024

char buffer[BUFFER\_SIZE];

void notify(int client){

    buffer[0] = 0x05;

    send(client, buffer, strlen(buffer),0);

}

int opt = 1;

int game\_board[3][3] = {0};

int check\_win() {

    // Kiểm tra hàng ngang

    for (int i = 0; i < 3; i++) {

        if (game\_board[i][0] != 0 &&

            game\_board[i][0] == game\_board[i][1] &&

            game\_board[i][1] == game\_board[i][2]) {

            return game\_board[i][0]; // 1 hoặc 2 thắng

        }

    }

    // Kiểm tra hàng dọc

    for (int i = 0; i < 3; i++) {

        if (game\_board[0][i] != 0 &&

            game\_board[0][i] == game\_board[1][i] &&

            game\_board[1][i] == game\_board[2][i]) {

            return game\_board[0][i]; // 1 hoặc 2 thắng

        }

    }

    // Kiểm tra haihai đường chéo chính

    if (game\_board[0][0] != 0 &&

        game\_board[0][0] == game\_board[1][1] &&

        game\_board[1][1] == game\_board[2][2]) {

        return game\_board[0][0]; // 1 hoặc 2 thắng

    }

    if (game\_board[0][2] != 0 &&

        game\_board[0][2] == game\_board[1][1] &&

        game\_board[1][1] == game\_board[2][0]) {

        return game\_board[0][2]; // 1 hoặc 2 thắng

    }

    // Kiểm tra hòa (không còn ô trống)

    int is\_full = 1;

    for (int i = 0; i < 3; i++) {

        for (int j = 0; j < 3; j++) {

            if (game\_board[i][j] == 0) {

                is\_full = 0;

                break;

            }

        }

    }

    if (is\_full) {

        return 3; // Hòa

    }

    return 0; // Chưa kết thúc

}

void res\_send(int cl\_1, int cl\_2, int res) {

    char buffer[BUFFER\_SIZE];

    buffer[0] = 0x04;

    const char \*message;

    if (res == 1) {

        message = "Nguoi choi X thang\n";

    } else if (res == 2) {

        message = "Nguoi choi O thang\n";

    } else {

        message = "Tran dau hoa\n";

    }

    // Ghi message vào buffer từ vị trí thứ 2 (sau header)

    snprintf(buffer + 1, BUFFER\_SIZE - 1, "%s", message);

    // Gửi cho cả 2 client

    send(cl\_1, buffer, strlen(buffer), 0);

    send(cl\_2, buffer, strlen(buffer), 0);

}

void update(int cl\_1, int cl\_2){

    buffer[0] = 0x03;

    for(int i = 0 ; i < 3; i++ ){

        for(int j = 0 ; j < 3; j++){

            if(game\_board[i][j] == 0){

                buffer[i \* 3 + j + 1] = ' ';

            }

            else if(game\_board[i][j] == 1){

                buffer[i \* 3 + j + 1] = 'X';

            }

            else if(game\_board[i][j] == 2){

                buffer[i \* 3 + j + 1] = 'O';

            }

        }

    }

    send(cl\_1, buffer, strlen(buffer),0);

    send(cl\_2, buffer,  strlen(buffer),0);

}

bool check\_board(int row, int col , int mark){

    if (row < 4 && row >0 && col < 4 && col > 0){

        if (game\_board[row - 1][col - 1] == 0){

            game\_board[row - 1][col - 1] = mark;

            return true;

        }

        return false;

    }

    return false;

}

int main(){

    int tcp\_socket, client\_1, client\_2;

    struct sockaddr\_in address;

    // Create socket

    if ((tcp\_socket = socket(AF\_INET, SOCK\_STREAM, 0)) == 0) {

        perror("socket failed");

        exit(EXIT\_FAILURE);

    }

    if (setsockopt(tcp\_socket, SOL\_SOCKET, SO\_REUSEADDR, &opt, sizeof(opt))) {

        perror("setsockopt");

        exit(EXIT\_FAILURE);

    }

    address.sin\_family = AF\_INET;

    address.sin\_addr.s\_addr = INADDR\_ANY;

    address.sin\_port = htons(PORT);

    if (bind(tcp\_socket, (struct sockaddr \*)&address, sizeof(address)) < 0) {

        perror("bind failed");

        exit(EXIT\_FAILURE);

    }

    if (listen(tcp\_socket, 2) < 0) {

        perror("listen");

        exit(EXIT\_FAILURE);

    }

    socklen\_t addrlen = sizeof(address);

    client\_1 = accept(tcp\_socket, (struct sockaddr \*)&address, &addrlen);

    client\_2 =  accept(tcp\_socket, (struct sockaddr \*)&address, &addrlen);

    int valread;

    while(1){

        while(1){

        printf("goi 1\n");

        notify(client\_1);

        valread = read(client\_1,buffer, BUFFER\_SIZE);

        if(buffer[0] == 0x02){

            int row = buffer[1] - '0';

            int col =buffer[2] - '0';

            if (check\_board(row, col, 1) == false){

                continue;

            }

            else{

                break;

            }

        }

    }

    update(client\_1, client\_2);

    printf("up cho sau khi 1 danh\n");

    if(check\_win() != 0){

        break;

    }

    while(1){

        printf("goi 2\n");

        notify(client\_2);

        valread = read(client\_2,buffer, BUFFER\_SIZE);

        if(buffer[0] == 0x02){

            int row = buffer[1] - '0';

            int col =buffer[2] - '0';

            if (check\_board(row, col, 2) == false){

                continue;

            }

            else{

                break;

            }

        }

    }

    update(client\_1, client\_2);

    printf("up cho sau khi 2 danh\n");

    if(check\_win() != 0){

        break;

    }

    }

    int result = check\_win();

    res\_send(client\_1, client\_2, result);

    close(client\_1);

    close(client\_2);

    close(tcp\_socket);

}

Client: Nhiệm vụ

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#include <sys/select.h>

#include <errno.h>

#define PORT 8080

#define BUFFER\_SIZE 100

int main(){

    int sockfd;

    char line[BUFFER\_SIZE];

    struct sockaddr\_in server\_addr;

    char buffer[BUFFER\_SIZE];

    fd\_set read\_fds;

    // Create a socket

    sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

    if (sockfd < 0) {

        perror("socket");

        exit(EXIT\_FAILURE);

    }

    // Set up the server address structure

    server\_addr.sin\_family = AF\_INET;

    server\_addr.sin\_port = htons(PORT);

    server\_addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1"); // Change this if needed

    // Connect to the server

    if (connect(sockfd, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) < 0) {

        perror("connect");

        close(sockfd);

        exit(EXIT\_FAILURE);

    }

    int valread;

    while(1){

        while(1){

        valread = read(sockfd, buffer, BUFFER\_SIZE);

        if (valread != 0){

            // printf("Da doc\n");

            break;

        }

        }

        if (buffer[0] == 0x05){

            // printf("Danh \n");

            printf("Nhap vi tri \n");

            buffer[0] = 0x02;

            fgets(line, sizeof(line), stdin);

            sscanf(line, "%c %c", &buffer[1], &buffer[2]);

            send(sockfd, buffer, 3, 0);

        }

        else if (buffer[0] == 0x03){

            // printf("ban \n");

            for(int i = 1; i <= 9; i++){

                if(i % 3 == 1){

                    printf("\n %c|", buffer[i]);

                }

                else{

                    printf(" %c|", buffer[i]);

                }

            }

            printf("\n");

        }

        else if(buffer[0] == 0x04){

            // printf("thua \n");

            printf("%s", &buffer[1]);

            break;

        }

    }

    close(sockfd);

}

Demo:

0x02: Thông điệp ghi dấu của người chơi

0x03: Thông điệp update thông tin bàn cờ

0x04: Thông điệp báo kết quả

0x05: Thông điệp báo đã đến lượt của người chơi

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

Nội dung do AI tạo ra có thể không chính xác.Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Nội dung do AI tạo ra có thể không chính xác.