Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовая работа по курсу

«Операционные системы»

ПРОЕКТИРОВАНИЕ КЛИЕНТ-СЕРВЕРНОЙ ИГРЫ

Студент: Юнусов Рустам Кобилович
Группа: М8О–310Б–22
Вариант: 5
Преподаватель: Миронов Евгений Сергеевич
Оценка:_____
Дата:_____
Подпись: _____

Москва, 2024

Постановка задачи

Цель работы

1. Приобретение практических навыков в использовании знаний, полученных в течении курса
2. Проведение исследования в выбранной предметной области

Задание

Необходимо спроектировать и реализовать программный прототип в соответствии с выбранным вариантом. Произвестианализ и сделать вывод на основании данных, полученных приработе программного прототипа.

Вариант

Морской бой. Общение между сервером и клиентом необходимо организовать при помощи memory map. Каждый игрок должен при запуске ввести свой логин. Для каждого игрока должна вестись статистика игр (сколько побед/поражений). Игрок может посмотреть свою статистику

Общие сведения о программе

Архитектура программы - клиент-серверная. К серверу подключаются два процесса - участники игры. Далее им предоставлена возможность осуществить сражение по правилам морского боя.

Общий алгоритм решения

При запуске исполняемого файла клиента, он связывается с сервером с помощью специального сокета. Основная идея решения состоит в общении между клиентами и сервером в формате “запрос-ответ”, при котором сервер выполняет определённые действия в зависимости от типа запроса.

Для реализации поставленной задачи необходимо:
1) Написать программы, реализующие сервер и клиент, а также заголовочный файл, который содержит структуру сообщения
2) Протестировать

Основные файлы программы

common.h

```c
#ifndef COMMON_H
#define COMMON_H

#include <stdalign.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <stdbool.h>

#define PORT 5555
#define MAX_PLAYERS 2
#define BOARD_SIZE 5
#define BUFFER_SIZE (sizeof(Message))

typedef struct {
    char name[50];
    int wins;
    int losses;
} PlayerStats;

typedef struct {
    int board[BOARD_SIZE][BOARD_SIZE];
    int hits;
} GameBoard;

typedef struct {
    char command[50];
    char data[256];
} Message;

void error(const char* msg) {
    perror(msg);
    exit(EXIT_FAILURE);
}

size_t min(size_t a, size_t b){
    return a < b ? a : b;
}

bool starts_with(const char* alpha, const char* theta){
    for(int i=0; i < min(a: strlen(s: alpha), b: strlen(s: theta)); ++i){
        if(alpha[i] != theta[i]){
            return false;
        }
    }
    return true;
}

#endif

```

server.c

```c
#include "common.h"
#include <stdint.h>
#include <time.h>

PlayerStats* load_stats(const char* filename) {
    int fd = open(filename, O_RDWR | O_CREAT, 0644);
    if (fd == -1) error(msg: "Ошибка открытия файла статистики");
    ftruncate(fd, sizeof(PlayerStats) * MAX_PLAYERS);

    PlayerStats* stats = mmap(NULL, sizeof(PlayerStats) * MAX_PLAYERS, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
    if (stats == MAP_FAILED) error(msg: "Ошибка memory mapping");
    close(fd);
    return stats;
}

void init_board(GameBoard* board) {
    memset(board->board, 0, sizeof(board->board));
    board->hits = 0;

    for (int i = 0; i < BOARD_SIZE; i++) {
        int x = rand() % BOARD_SIZE;
        int y = rand() % BOARD_SIZE;
        board->board[x][y] = 1;
    }
}

void handle_client(int client_socket, PlayerStats* stats) {
    GameBoard board;
    init_board(board: &board);
    char buffer[BUFFER_SIZE];
    while (1) {
        bzero(buffer, BUFFER_SIZE);
        read(client_socket, buffer, BUFFER_SIZE);

        Message* msg = (Message*)buffer;

        if (strcmp(s1: msg->command, s2: "move") == 0) {
            int x, y;
            sscanf(msg->data, "%d %d", &x, &y);

            if (x < 0 || x >= BOARD_SIZE || y < 0 || y >= BOARD_SIZE) {
                snprintf(msg->data, sizeof(msg->data), "Некорректный ход!");
            } else if (board.board[x][y] == 1) {
                board.board[x][y] = 0;
                board.hits++;
                snprintf(msg->data, sizeof(msg->data), "Попадание!");

                if (board.hits == BOARD_SIZE) {
                    snprintf(msg->data, sizeof(msg->data), "Вы победили!");
                    stats->wins++;
                }
            } else {
                snprintf(msg->data, sizeof(msg->data), "Мимо!");
            }
        } else if (strcmp(s1: msg->command, s2: "stats") == 0) {
            snprintf(msg->data, sizeof(msg->data), "Победы: %d, Поражения: %d", stats->wins, stats->losses);
        } else if (strcmp(s1: msg->command, s2: "quit") == 0) {
            close(client_socket);
            return;
        }

        write(fd: client_socket, buf: buffer, nbyte: sizeof(Message));
    }
}


int main() {
    srand(time(NULL));

    PlayerStats* stats = load_stats(filename: "stats.dat");
    int server_socket, client_socket;
    struct sockaddr_in server_addr, client_addr;

    server_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (server_socket == -1) error(msg: "Ошибка создания сокета");

    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(PORT);

    if (bind(server_socket, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0)
        error(msg: "Ошибка привязки");

    listen(server_socket, MAX_PLAYERS);
    printf("Сервер запущен. Ожидание подключения игроков...\n");

    socklen_t client_len = sizeof(client_addr);
    while ((client_socket = accept(server_socket, (struct sockaddr*)&client_addr, &client_len)) > 0) {
        printf("Игрок подключился!\n");
        if (!fork()) {
            close(server_socket);
            handle_client(client_socket, stats);
            exit(0);
        }
        close(client_socket);
    }

    close(server_socket);
    munmap(stats, sizeof(PlayerStats) * MAX_PLAYERS);
    return 0;
}

```

client.c

```c
#include "common.h"

void play_game(int server_socket) {
    char buffer[256];
    Message msg;

    while (1) {
        printf("Введите команду (move X Y, stats, quit): ");
        fgets(buffer, 256, stdin);
        sscanf(buffer, "%s %[^\n]", msg.command, msg.data);

        write(fd: server_socket, buf: &msg, nbyte: sizeof(Message));
        read(server_socket, &msg, sizeof(Message));

        printf("Ответ сервера: %s\n", msg.data);
        if (strcmp(s1: msg.command, s2: "quit") == 0) break;
    }
}

int main() {
    int client_socket;
    struct sockaddr_in server_addr;

    client_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (client_socket == -1) error(msg: "Ошибка создания сокета");

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

    if (connect(client_socket, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0)
        error(msg: "Ошибка подключения");

    printf("Успешно подключено к серверу!\n");
    play_game(server_socket: client_socket);
    close(client_socket);

    return 0;
}
```

Пример работы

```
Успешно подключено к серверу!
Введите команду (move X Y, stats, quit): stats
Ответ сервера: Победы: 0, Поражения: 0
Введите команду (move X Y, stats, quit): move 1 1
Ответ сервера: Мимо!
Введите команду (move X Y, stats, quit): move 2 2
Ответ сервера: Мимо!
Введите команду (move X Y, stats, quit): move 3 1
Ответ сервера: Мимо!
Введите команду (move X Y, stats, quit):
```

```
~/codes/OperationSystemsLabs/course work git:(main)±3
./client

Успешно подключено к серверу!
Введите команду (move X Y, stats, quit): stats
Ответ сервера: Победы: 0, Поражения: 0
Введите команду (move X Y, stats, quit): move 2 2
Ответ сервера: Мимо!
Введите команду (move X Y, stats, quit):
```

```
~/codes/OperationSystemsLabs/course work git:(main)±3
./server

Сервер запущен. Ожидание подключения игроков...
Игрок подключился!
Игрок подключился!
Игрок подключился!
```

Вывод

В ходе выполнения курсовой работы мне удалось лучше понять и закрепить знания о работе с памятью через memory-mapped files. Я научился эффективно использовать этот механизм для взаимодействия между процессами и реализации многозадачности. Работа над проектом была полезной, так как она связана с реальными задачами программирования, и я смог не только применить теоретические знания, но и получить практический опыт, который, уверен, пригодится в будущей профессиональной деятельности.