Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное учреждение высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

Кафедра информационных технологий и систем

**Сетевая игра «Русская рулетка»**

Курсовая работа

по дисциплине:

«Операционные системы»

Разработал:

Студент группы 8091

Ефимов Артем

«\_\_».\_\_\_\_\_.2020г.

Проверил:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_».\_\_\_\_\_.2020г.

**Великий Новгород**

**2020**

Оглавление

ОГЛАВЛЕНИЕ............................................................................................................................... 2

ВВЕДЕНИЕ.................................................................................................................................... 3

ГЛАВА 1. РАЗРАБОТКА ПРОЕКТА............................................................................................ 3

1.1 Постановка задачи................................................................................................................... 3

1.2 Выбор инструментальных средств........................................................................................ 3

1.3 Выбор модели.......................................................................................................................... 3

1.4 Алгоритм решения задачи.......................................................................................................3

ГЛАВА 2. РЕАЛИЗАЦИЯ ПРОЕКТА....................................................................................... ..4

2.1 Структура клиента и сервера................................................................................................. 4

2.2 Программный интерфейс сервера......................................................................................... 4

2.3 Описание программной реализации клиента...................................................................... 4

2.4 Системные вызовы................................................................................................................. 4

2.5 Пример работы приложения.................................................................................................. 5

2.6 Результат работы..................................................................................................................... 5

ЗАКЛЮЧЕНИЕ............................................................................................................................. 5

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ...................................................................... 5

Листинг В1 — Код из файла server.с........................................................................................... 6

Листинг В2 — Код из файла client.с........................................................................................... 9

ВВЕДЕНИЕ

Цель данной работы – создание простейшей консольной версии игры «Русская рулетка». **Ру́сская руле́тка**, или **гуса́рская рулетка**, также **сопра́но** — экстремальная азартная игра или пари с летальным исходом. По правилам игры в пустой барабан револьвера заряжается один или несколько патронов с тем, чтобы остались свободные каморы, после чего барабан несколько раз проворачивается так, чтобы игроки не знали, где располагаются патроны, а где — пустые каморы. После этого игроки по очереди подносят ствол револьвера к собственной голове и нажимают на спусковой крючок. В обычных азартных играх для имитации русской рулетки используется игрушечный револьвер.

Правила игры: Заряжается револьвер одним патроном, случайно прокручивается, делаются ходы по очереди до смерти одного из игроков или до того момента, пока один не сдастся

Для целей данной курсовой работы взята вариация игры с двумя игроками

Функции, реализованные в игре:

1. Подключение двух игроков
2. Возможность для игроков совершать ходы
3. Консольный интерфейс для ввода хода и вывода результата

ГЛАВА 1. РАЗРАБОТКА ПРОЕКТА

1.1 Постановка задачи

Задачей данной курсовой работы является разработка сетевой игры «Русская рулетка», которая рассчитана на двух игроков. Игра должна работать под управлением операционной системы семейства Unix/Linux. Сетевая часть игры должна быть выполнена на языке C++ с помощью сокетов предоставляемых системной библиотекой <sys/socket.h>.

1.2 Выбор инструментальных средств

Для реализации сетевого взаимодействия используются TCP – сокеты.

1.3 Выбор модели

Для взаимодействия между игроками выбрана клиент-серверная модель. Сервер обеспечивает обработку ходов, отправку результатов клиентам.

1.4 Алгоритм решения задачи

Подключение игрока/игроков происходит следующим образом:

К серверу подключаются первый и второй игрок, введя адрес сервера и номер порта.

Ход игры:

Сервер ожидает ходы от обоих игроков. Создается виртуальный револьвер, в который помещается один патрон. Сервер отправляет коды игрокам о том, кто ходит, а кто нет. После этого, сервер ждет от игрока ответа. При получении ответа сервер либо проверяет наличие патрона в револьвере, либо отправляет сообщения другому клиенту, что противник сдался. После проверки револьвера рассылаются сообщения о статусе этого хода.

ГЛАВА 2. РЕАЛИЗАЦИЯ ПРОЕКТА

2.1 Структура клиента и сервера

Соединение между клиентом и сервером. Для игры также используются некоторые вспомогательные функции.

2.2 Программный интерфейс сервера

Для целей обмена информацией между клиентом и сервером используются специальные коды: 0 — ход, 1- сдался. В ответ клиенту сервер отправляет следующие коды: 0 — успешный ход, 1 — смерть, 2- противник сдался, 3 — противник умер, 4 — право на ход

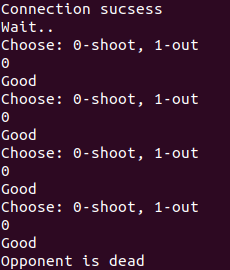
2.3 Описание программной реализации клиента

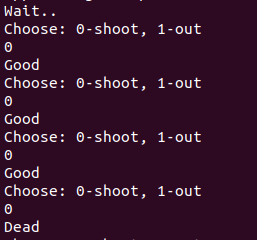
При запуске клиента происходит подключение к серверу. Клиент ожидает от сервера коды. В зависимости от полученных кодов выводит соответствующее сообщение в консоль, дает возможность совершить ход

2.4 Системные вызовы

|  |  |
| --- | --- |
| socket() | Служит для создания сокета в ОС. Имеет три парметра. Первый указывает, к какому семейству протоколов относится создаваемый сокет, а второй и третий параметры определяют конкретный протокол внутри данного семейства. |
| Connect() | Служит для установки логического соединения со стороны клиента. Имеет три параметра: дескриптор активного сокета, через который будет устанавливаться соединение, полный адрес сокета сервера и его длина. |
| ioctl() | Выполняет различные управляющие действия над обычными устройствами и псевдоустройствами. Если в качестве параметров используются дескриптор сокета, флаг FIONREAD и целочисленная переменная, записывает в переменную, указанную третим параметром, количество доступных для чтения байт на сокете. |
| read() и write() | Используются для чтения и записи файлов. Могут быть использованы для обмена данных, если в качестве параметров вместо дескрипторов файлов в них задаются дескрипторы сокетов. |
| bind() | Настраивает адрес для созданного сокета. Первый параметр вызова должен содержать дескриптор сокета, для которого производится настройка адреса. Второй и третий параметры задают этот адрес. |
| listen() | Прослушивает соединения на сокете. Позволяет показать готовность принимать соединения и задать лимит входящих соединений. В качестве первого параметра используется дескриптор сокета. Второй параметр определяет максимальную длину очереди входящих соединений. |
| **htons()** | преобразует узловой порядок расположения байтов положительного короткого целого *hostshort* в сетевой порядок расположения байтов. |
| **ntohs()** | преобразует сетевой порядок расположения байтов положительного короткого целого *netshort* в узловой порядок расположения байтов. |
| accept() | Эта функция используется сервером для принятия связи на сокет. Сокет должен быть уже слушающим в момент вызова функции. Если сервер устанавливает связь с клиентом, то функция accept возвращает новый сокет-дескриптор, через который и происходит общение клиента с сервером. Пока устанавливается связь клиента с сервером, функция accept блокирует другие запросы связи с данным сервером, а после установления связи "прослушивание" запросов возобновляется. |

2.5 Пример работы приложения





2.6 Результат работы

Создана работающая игра сетевая игра «Русская рулетка», в которой для сетевого взаимодействия игроков используются клиент-серверная модель, а обмен данными осуществляется через TCP – сокеты.

ЗАКЛЮЧЕНИЕ

В результате, было создано сетевое приложение соответствующее цели работы. Приложение работает на операционных системах семейства Linux. Для построения сетевой части приложения использован язык C++ и компилятор g++. Сетевое взаимодействие было построено с помощью протокола TCP.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Скотт Граннеман «Linux. Карманный справочник»

2. Брюс Моли «Unix/Linux. Теория и практика программирования»

3. [https://ru.wikipedia.org/wiki/%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%B0%D1%8F\_%D1%80%D1%83%D0%BB%D0%B5%D1%82%D0%BA%D0%B0](https://ru.wikipedia.org/wiki/Русская_рулетка)

4. <https://linux.die.net/>

[Листинг В1 — Код из файла server.](#__RefHeading___Toc10981_812659571)с

#include <stdio.h>

#include <sys/shm.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <string.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <sys/sem.h>

#include <sys/msg.h>

#include <sys/ipc.h>

#include <fcntl.h>

#include <sys/mman.h>

#include <sys/stat.h>

#include <signal.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <errno.h>

#include <time.h>

#include <string>

#include <iostream>

#include <ctime>

void game(int listenfd, int cl1, int cl2) {

    while (1) {

    int num = 8;

    srand(static\_cast<unsigned int>(time(0)));

    rand();

    int rev[8];

    int check;

    for(int i=0; i<num; i++)

    {

        rev[i]=0;

    }

    rev[rand()%num]=1;

    for(int i=0; i<num; i++)

    {

        if (i%2==0) // ход 1 игрока

        {

            check = 4;

            check = htons (check);

            write (cl1, &check, sizeof(int));

            read (cl1, &check, sizeof(int));

            check = ntohs (check);

            if (check == 0) // ход

            {

                if(rev[i]==0) // успешный ход

                {

                check = 0;

                check = htons (check);

                write (cl1, &check, sizeof(int));

                }

                else // смэрть

                {

                check = 1;

                check = htons (check);

                write (cl1, &check, sizeof(int));

                check = 3;

                check = htons (check);

                write (cl2, &check, sizeof(int));

                break;

                }

            }

            else // сдался

            {

            check = 2;

            check = htons (check);

            write (cl2, &check, sizeof(int));

            cl1 = accept(listenfd, (struct sockaddr\*)NULL, NULL); //ждем подключеня 1 клиента

            break;

            }

        }

        else // аналогично для хода второго игрока

        {

            check = 4;

            check = htons (check);

            write (cl2, &check, sizeof(int));

            read (cl2, &check, sizeof(int));

            check = ntohs (check);

            if (check == 0)

            {

                if(rev[i]==0)

                {

                check = 0;

                check = htons (check);

                write (cl2, &check, sizeof(int));

                }

                else

                {

                check = 1;

                check = htons (check);

                write (cl2, &check, sizeof(int));

                check = 3;

                check = htons (check);

                write (cl1, &check, sizeof(int));

                break;

                }

            }

            else

            {

            check = 2;

            check = htons (check);

            write (cl1, &check, sizeof(int));

            cl2 = accept(listenfd, (struct sockaddr\*)NULL, NULL); //ждем подключеня 2 клиента

            break;

            }

        }

    }

    }

}

int main(int argc, char \*argv[]) {

        int listenfd = 0;

        struct sockaddr\_in serv\_addr;

        char sendBuff[1025];

        time\_t ticks;

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

        memset(&serv\_addr, '0', sizeof(serv\_addr));

        memset(sendBuff, '0', sizeof(sendBuff));

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

        serv\_addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

        serv\_addr.sin\_port = htons(atoi(argv[1]));

        bind(listenfd, (struct sockaddr\*)&serv\_addr, sizeof(serv\_addr));

        listen(listenfd, 10);

    int cl1 = accept(listenfd, (struct sockaddr\*)NULL, NULL); //ждем подключеня 1 клиента

        int cl2 = accept(listenfd, (struct sockaddr\*)NULL, NULL); //ждем подключеня 2 клиента

        game(listenfd, cl1, cl2);

}

[Листинг В2 — Код из файла client.](#__RefHeading___Toc10983_812659571)c

#include <stdio.h>

#include <sys/shm.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <string.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <sys/sem.h>

#include <sys/msg.h>

#include <sys/ipc.h>

#include <fcntl.h>

#include <sys/mman.h>

#include <sys/stat.h>

#include <signal.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <errno.h>

#include <string>

#include <iostream>

int main(int argc, char \*argv[]) {

        int sockfd = 0, n = 0;

        char recvBuff[1024];

        struct sockaddr\_in serv\_addr;

        if(argc != 3) {

            printf("\n Usage: %s <ip of server> \n",argv[0]);

            return 1;

        }

        memset(recvBuff, '0',sizeof(recvBuff));

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

        {

            printf("\n Error : Could not create socket \n");

            return 1;

        }

        memset(&serv\_addr, '0', sizeof(serv\_addr));

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

        serv\_addr.sin\_port = htons(atoi(argv[2]));

        if(inet\_pton(AF\_INET, argv[1], &serv\_addr.sin\_addr)<=0)

        {

            printf("\n inet\_pton error occured\n");

            return 1;

        }

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

        {

           printf("\n Error : Connect Failed \n");

           return 1;

        }

        printf("Connection sucsess\n");

        printf("Wait..\n");

    while (1) {

    int check;

    read (sockfd, &check, sizeof(int));

    check = ntohs(check);

    switch (check)

    {

    case 0:

    printf("Good\n");

    break;

    case 1:

    printf("Dead\n");

    break;

    case 2:

    printf("Opponent give up\n");

    printf("Wait..\n");

    break;

    case 3:

    printf("Opponent is dead\n");

    break;

    case 4:

    printf("Choose: 0-shoot, 1-out\n");

    do {

    std::cin >> check;

    } while (check != 0 && check != 1);

    if (check == 0) {

        check = htons(check);

        write (sockfd, &check, sizeof(int));

    }

    else {

        check = htons(check);

        write (sockfd, &check, sizeof(int));

        return 0;

    }

    break;

    }

    }

}