**Московский авиационный институт (национальный исследовательский университет)**

Институт информационных технологий и прикладной математики

«Кафедра вычислительной математики и программирования»

**Курсовой проект по предмету "Операционные системы"**

Студент: Федоров М. А.

Преподаватель: Миронов Е.С.

Группа: М8О-207Б-22

Дата: 15.09.2022

Оценка:

Подпись:

Оглавление

[Цель работы 3](#__RefHeading__738_2004454639)

[Постановка задачи 3](#__RefHeading__740_2004454639)

[Общие сведения о программе 4](#__RefHeading__742_2004454639)

[Общий алгоритм решения 5](#__RefHeading__744_2004454639)

[Реализация 5](#__RefHeading__746_2004454639)

[Пример работы 8](#__RefHeading__748_2004454639)

[Вывод 9](#__RefHeading__750_2004454639)

# **Цель работы**

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

# **Постановка задачи**

Необходимо спроектировать и реализовать программный прототип в соответствии с выбранным вариантом. Произвести анализ и сделать вывод на основании данных, полученных при работе программного прототипа.

Проектирование консольной клиент-серверной игры на основе любой из выбранных технологий:

* Pipes
* Sockets
* Сервера очередей
* И другие

Создать собственную игру более, чем для одного пользователя. Игра может быть устроена по принципу: клиент-клиент, сервер-клиент.

*Вариант 1*

Консоль-серверная игра. Необходимо написать консоль-серверную игру. Необходимо написать 2 программы: сервер и клиент. Сначала запускается сервер, а далее клиенты соединяются с сервером. Сервер координирует клиентов между собой. При запуске клиента игрок может выбрать одно из следующих действий (возможно больше, если предусмотрено вариантом):

* Создать игру, введя ее имя
* Присоединиться к одной из существующих игр по имени игры

Морской бой.

Общение между сервером и клиентом необходимо организовать при

помощи pipe'ов. 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**

Программа представлена двумя файлами **–** client.cpp и server.cpp.

В программе используются следующие системные вызовы:

**pipe() –** создаёт однонаправленный канал данных, который можно использовать для взаимодействия между процессами(конвейер)

**fork() –** создание дочернего процесса, в переменной id будет лежать «специальный код» процесса(-1 -ошибка, 0- дочерний процесс, >0- родительский)

**read()** **–** чтение из канала pipe()

**write()** **–** запись в канал pipe()

**dup2() –** перенаправление дескриптора

**execv() –** создание процесса с другой программой

**close() –** закрытие файлового дескриптора, который после этого не ссылается ни на один и файл и может быть использован повторно.

# **Общий алгоритм решения**

# **Реализация**

**client.cpp**

#include <bits/stdc++.h>

#include <unistd.h>

#include <sys/wait.h>

int main(int argc, char \*argv[])

{

    const int max\_count\_ships=1;

    int state = std::stoi(argv[1]);

    if (state == 1)

    {

        std::string login;

        std::cin >> login;

        write(STDOUT\_FILENO, login.c\_str(), login.length());

        return 0;

    }

    else if (state==2)

    {

        std::string request,temp;

        for (size\_t i = 0; i < max\_count\_ships; ++i)

        {

            std::getline(std::cin, temp);

            request+=(temp+"\n");

        }

        write(STDOUT\_FILENO, request.c\_str(), request.length());

        return 0;

    }

    else if(state==3){

        std::string status;

        std::cin >> status;

        write(STDOUT\_FILENO, status.c\_str(), status.length());

        return 0;

    }

    else if (state==4)

    {

        int x,y;

        std::cin>>x>>y;

        --x;

        --y;

        std::string coords=std::to\_string(x)+" "+std::to\_string(y);

        write(STDOUT\_FILENO, coords.c\_str(), coords.length());

    }

    else if (state==5)

    {

        std::string request;

        std::cin>>request;

        write(STDOUT\_FILENO, request.c\_str(), request.length());

    }

    return 0;

}

**-----------------------------------------------------------**

**server.cpp**

#include <bits/stdc++.h>

#include <unistd.h>

#include <sys/wait.h>

using matrix=std::vector<std::vector<int>>;

const int MAX\_COUNT\_SHIPS=1;

void arrangement\_ships(matrix &map\_player, const matrix input\_ships)

{

    for (size\_t i = 0; i <input\_ships.size()-1; ++i)

    {

        int size\_ship = input\_ships[i][0];

        switch (size\_ship)

        {

        case 1:

            if (input\_ships[i][1] == input\_ships[i][3] && input\_ships[i][2] == input\_ships[i][4])

            {

                int placeX = input\_ships[i][1] - 1;

                int placeY = input\_ships[i][2] - 1;

                map\_player[placeX][placeY] = 1;

            }

            break;

        case 2:

        case 3:

        case 4:

            if (input\_ships[i][1] == input\_ships[i][3])

            {

                int min\_place = std::min(input\_ships[i][2], input\_ships[i][4]) - 1;

                int max\_place = std::max(input\_ships[i][2], input\_ships[i][4]) - 1;

                int placeX = input\_ships[i][1] - 1;

                for (size\_t j = min\_place; j <= max\_place; ++j)

                {

                    map\_player[placeX][j] = 1;

                }

            }

            if (input\_ships[i][2] == input\_ships[i][4])

            {

                int min\_place = std::min(input\_ships[i][1], input\_ships[i][3]) - 1;

                int max\_place = std::max(input\_ships[i][1], input\_ships[i][3]) - 1;

                int placeY = input\_ships[i][2] - 1;

                for (size\_t j = min\_place; j <= max\_place; ++j)

                {

                    map\_player[j][placeY] = 1;

                }

            }

            break;

        default:

            break;

        }

    }

}

void print\_matrix(matrix matrix)

{

    for (int row\_index = 0; row\_index < 10; ++row\_index)

    {

        for (int column\_index = 0; column\_index < 10; ++column\_index)

        {

            if (matrix[row\_index][column\_index] == 3)

            {

                std::cout << "\u2715"

                          << " ";

            }

            if (matrix[row\_index][column\_index] == 2)

            {

                std::cout << "\u25A1\u0338"

                          << " ";

            }

            if (matrix[row\_index][column\_index] == 1)

            {

                std::cout << "\u25A1"

                          << " ";

            }

            if (matrix[row\_index][column\_index] == 0)

            {

                std::cout << "\u00B7"

                          << " ";

            }

        }

        std::cout << "\n";

    }

}

void clear()

{

    std::cout << "\x1B[2J\x1B[H";

}

std::string startMenu(int (&pipe\_first)[2]){

    std::cout<<"1 - посмотреть статистику"<<std::endl;

    pid\_t process\_id = fork();

        if (process\_id == 0)

        {

            close(pipe\_first[0]);

            dup2(pipe\_first[1], STDOUT\_FILENO);

            close(pipe\_first[1]);

            execl("./client", "./client", "5", NULL);

            perror("exec");

            exit(1);

        }

        else

        {

            wait(NULL);

            char result\_child\_first[200];

            int bytes\_read\_first = read(pipe\_first[0], result\_child\_first, sizeof(result\_child\_first));

            if (bytes\_read\_first > 0)

            {

                std::string request(result\_child\_first, bytes\_read\_first);

                return request;

            }

        }

    return "";

}

void startGame(int (&pipe\_first)[2], matrix &map\_player,matrix& input\_ships,auto iterator\_players)

{

    std::string request=startMenu(pipe\_first);

    if (request=="1")

    {

        std::cout<<iterator\_players->first + ": Выиграл "<<iterator\_players->second.first<<".";

        std::cout<<"Проиграл "<<iterator\_players->second.second<<"\n";

    }

    std::cout << "Введите для каждых " << MAX\_COUNT\_SHIPS << " кораблей размер (от 1 до 4) и две точки (в одну строку, разделяя пробелами). При вводе неверных данных корабль не будет размещен"

              << "\n";

        pid\_t process\_id = fork();

        if (process\_id == 0)

        {

            close(pipe\_first[0]);

            dup2(pipe\_first[1], STDOUT\_FILENO);

            close(pipe\_first[1]);

            execl("./client", "./client", "2", NULL);

            perror("exec");

            exit(1);

        }

        else

        {

            wait(NULL);

            char result\_child\_first[200];

            int bytes\_read\_first = read(pipe\_first[0], result\_child\_first, sizeof(result\_child\_first));

            if (bytes\_read\_first > 0)

            {

                std::stringstream ss(result\_child\_first);

                std::string line;

                while (std::getline(ss, line, '\n'))

                {

                    std::vector<int> row;

                    std::istringstream iss(line);

                    int number;

                    while (iss >> number)

                    {

                        row.push\_back(number);

                    }

                    input\_ships.push\_back(row);

                }

                arrangement\_ships(map\_player, input\_ships);

                print\_matrix(map\_player);

            }

        }

}

bool killShip(matrix &map\_player,matrix& input\_ships){

    for (size\_t i = 0; i < input\_ships.size()-1; ++i)

    {

        bool isKill=true;

        int size\_ship = input\_ships[i][0];

        int placeX, placeY;

        switch (size\_ship)

        {

        case 1:

            placeX = input\_ships[i][1] - 1;

            placeY = input\_ships[i][2] - 1;

            if (map\_player[placeX][placeY] == 2)

            {

                input\_ships.erase(input\_ships.begin() + i);

                return true;

            }

            break;

        case 2:

        case 3:

        case 4:

            if (input\_ships[i][1] == input\_ships[i][3])

            {

                int min\_place = std::min(input\_ships[i][2], input\_ships[i][4]) - 1;

                int max\_place = std::max(input\_ships[i][2], input\_ships[i][4]) - 1;

                int placeX = input\_ships[i][1] - 1;

                size\_t j;

                for (j = min\_place; j <= max\_place; ++j)

                {

                    if(map\_player[placeX][j] !=2){

                        isKill=false;

                        break;

                    }

                }

                if(isKill){

                    input\_ships.erase(input\_ships.begin() + i);

                    return true;

                }

            }

            if (input\_ships[i][2] == input\_ships[i][4])

            {

                int min\_place = std::min(input\_ships[i][1], input\_ships[i][3]) - 1;

                int max\_place = std::max(input\_ships[i][1], input\_ships[i][3]) - 1;

                int placeY = input\_ships[i][2] - 1;

                size\_t j;

                 for (j = min\_place; j <= max\_place; ++j)

                {

                    if(map\_player[j][placeY] !=2){

                        isKill=false;

                        break;

                    }

                }

                if(isKill){

                    input\_ships.erase(input\_ships.begin() + i);

                    return true;

                }

            }

            break;

        default:

            return false;

            break;

        }

    }

    return false;

}

void giveInfo(int x, int y, matrix &map\_player, matrix &map\_enemy,matrix& input\_ships,int& count\_kill\_ships)

{

    if (map\_player[x][y] == 1)

    {

        map\_player[x][y] = 2;

        map\_enemy[x][y] = 2;

        if (killShip(map\_player,input\_ships))

        {

            ++count\_kill\_ships;

            std::cout<<"убил"<<std::endl;

        }

        else{

        std::cout << "ранил" << std::endl;

        }

    }

    else if (map\_player[x][y] == 0)

    {

        map\_player[x][y] = 3;

        map\_enemy[x][y] = 3;

        std::cout << "мимо" << std::endl;

    }

    else

    {

        std::cout << "ход в молоко" << std::endl;

    }

}

void movePlayer(int (&pipe\_first)[2],matrix &map\_current,matrix &map\_player, matrix &map\_enemy, std::string player,matrix &input\_ships,int& count\_kill\_ships)

{

    std::cout << player + " ходит"

              << "\n";

    pid\_t process\_id\_first = fork();

    if (process\_id\_first == 0)

    {

        close(pipe\_first[0]);

        dup2(pipe\_first[1], STDOUT\_FILENO);

        close(pipe\_first[1]);

        execl("./client", "./client", "4", NULL);

        perror("exec");

        exit(1);

    }

    else

    {

        wait(NULL);

        char result\_child\_first[200];

        std::vector<int> coords;

        int bytes\_read\_first = read(pipe\_first[0], result\_child\_first, sizeof(result\_child\_first));

        if (bytes\_read\_first > 0)

        {

            std::stringstream ss(result\_child\_first);

            std::string line;

            std::getline(ss, line, '\n');

            std::istringstream iss(line);

            int number;

            while (iss >> number)

            {

                coords.push\_back(number);

            }

        }

        giveInfo(coords[0], coords[1], map\_player, map\_enemy,input\_ships,count\_kill\_ships);

        std::cout << "Ваше поле:" << std::endl;

        print\_matrix(map\_current);

        std::cout << std::endl;

        std::cout << "Поле противника:" << std::endl;

        print\_matrix(map\_enemy);

    }

}

int main()

{

    int pipe\_first[2];

    int pipe\_second[2];

    int count\_kill\_ships\_first=0;

    int count\_kill\_ships\_second=0;

    matrix input\_ships\_first;

    matrix input\_ships\_second;

    matrix map\_player1(10, std::vector<int>(10, 0));

    matrix map\_player2(10, std::vector<int>(10, 0));

    matrix map\_enemy1(10, std::vector<int>(10, 0));

    matrix map\_enemy2(10, std::vector<int>(10, 0));

    std::unordered\_map<std::string, std::pair<int, int>> players;

    if (pipe(pipe\_first) < 0 || pipe(pipe\_second) < 0)

    {

        perror("create pipe");

        exit(1);

    }

    {

        pid\_t process\_id\_first = fork();

        if (process\_id\_first == 0)

        {

            std::cout << "Введите логины: " << std::endl;

            close(pipe\_first[0]);

            dup2(pipe\_first[1], STDOUT\_FILENO);

            close(pipe\_first[1]);

            execl("./client", "./client", "1", NULL);

            perror("exec");

            exit(1);

        }

        else

        {

            pid\_t process\_id\_second = fork();

            if (process\_id\_second == 0)

            {

                close(pipe\_second[0]);

                dup2(pipe\_second[1], STDOUT\_FILENO);

                close(pipe\_second[1]);

                execl("./client", "./client", "1", NULL);

                perror("exec");

                exit(1);

            }

            else

            {

                wait(NULL);

                wait(NULL);

                char result\_child\_first[200];

                char result\_child\_second[200];

                int bytes\_read\_first = read(pipe\_first[0], result\_child\_first, sizeof(result\_child\_first));

                int bytes\_read\_second = read(pipe\_second[0], result\_child\_second, sizeof(result\_child\_second));

                if (bytes\_read\_first > 0)

                {

                    std::string player(result\_child\_first, bytes\_read\_first);

                    players[player] = std::make\_pair(0, 0);

                    std::cout << player + " is loggined" << std::endl;

                }

                if (bytes\_read\_second > 0)

                {

                    std::string player2(result\_child\_second, bytes\_read\_second);

                    players[player2] = std::make\_pair(0, 0);

                    std::cout << player2 + " is loggined" << std::endl;

                }

            }

        }

    }

    auto it1 = players.begin();

    auto it2 = players.begin();

    ++it2;

    while (true)

    {

        startGame(pipe\_first, map\_player1,input\_ships\_first,it1);

        clear();

        startGame(pipe\_second, map\_player2,input\_ships\_second,it2);

        clear();

        while (true)

        {

            movePlayer(pipe\_first,map\_player1,map\_player2, map\_enemy1, it1->first,input\_ships\_first,count\_kill\_ships\_first);

            if (count\_kill\_ships\_first==MAX\_COUNT\_SHIPS)

            {

                std::cout<<it1->first+" победил"<<std::endl;

                std::cout<<it2->first+" проиграл"<<std::endl;

                it1->second.first++;

                it2->second.second++;

                break;

            }

            // clear(pipe\_first);

            movePlayer(pipe\_second,map\_player2, map\_player1, map\_enemy2, it2->first,input\_ships\_second,count\_kill\_ships\_second);

            if (count\_kill\_ships\_second==MAX\_COUNT\_SHIPS)

            {

                std::cout<<it2->first+" победил"<<std::endl;

                std::cout<<it1->first+" проиграл"<<std::endl;

                it2->second.first++;

                it1->second.second++;

                break;

            }

            // clear(pipe\_second);

        }

        std::cout<<"\n";

    }

    return 0;

}

**-----------------------------------------------------------**

# **Пример работы**

**Test 1**

|  |  |
| --- | --- |
| Input | Output |
| A  B  1  1 1 1 1 1  1  1 1 1 1 1  1 1  1 |  |

# **Вывод**