Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовой проект по курсу «Операционные системы»

Студент: Ядров Артем Леонидович
Группа: М8О-208Б-20
Вариант: 7
Преподаватель: Миронов Евгений Сергеевич

Оценка: ___________
Дата: ___________
Подпись: ___________

Москва, 2021

Содержание

Репозиторий

https://github.com/Yadroff/OS/tree/master/KP

Постановка задачи

Цель работы
Целью работы является:

- Приобретение практических навыков в использовании знаний, полученных в течении курса
- Проведение исследования в выбранной предметной области

Задание
Необходимо спроектировать и реализовать программный прототип в соответствии с выбранным вариантом. Произвести анализ и сделать вывод на основании данных, полученных при работе программного прототипа.
Спроектировать консольную клиент-серверную игру на основе технологии Pipes.
Вариант №7: «Быки и коровы» (угадывать необходимо слова). Общение между сервером и клиентом необходимо организовать при помощи pipe'ов. При создании каждой игры необходимо указывать количество игроков, которые будут участвовать. То есть угадывать могут несколько игроков. Если кто-то из игроков вышел из игры, то игра должна быть продолжена

Общие сведения о программе

Связь между клиентом и сервером поддерживается путем именнованных pipe'ов. Все клиенты записывают команды в pipe сервера. Сервер же, в свою очередь, создает pipe для клиента и записывает туда. Также организована система аккаунтов (для игры необходимо зарегистрироваться или войти в существующий аккаунт. Все логины и пароли хранятся в шифрованной базе данных), результаты игр записываются в отдельный файл.

Общий метод и алгоритм решения

Используемые методы системные вызовы:

- write
- read
- mkfifo

- perror

Также используется библиотека «fstream» для чтения/записи в файл.

Исходный код

functions.h

```
#ifndef KP_FUNCTIONS_H
#define KP_FUNCTIONS_H

#include <valarray>
#include <string>
#include <iostream>
#include <vector>
#include <unistd.h>

const char *SERVER_INPUT_FILE_NAME = "server_in";
const char *DATA_BASE_FILE_NAME = "data_base";
const char *RESULTS_FILE_NAME = "results";
const char *SECRET_KEY = "$";
const std::string ADMIN_NAME("Yadroff");
const std::string ADMIN_PASSWORD("qwerty");
const int INF = (int) pow(10, 9) + 7;
#define ACCESS_PERMS S_IRWXO | S_IRWXG | S_IRWXU
std::pair<int, int> transform(std::pair<std::string, std::string> &pair) {
    /* Преобразование пары строк в пару чисел происходит следующим образом:
     * первая строка проходится в прямом порядке от начала к концу и хэшируется
     * вторая строка проходится в обратном порядке*/
    std::string first_string = pair.first, second_string = pair.second;
    int first = 0, second = 0;
    for (auto i: first_string) {
        first = (first * 10 + i) % INF;
    }
    for (auto it = second_string.rbegin(); it != second_string.rend(); ++it) {
        second = (second * 10 + *it) % INF;
    }
    return std::make_pair(first, second);
}

// сообщение отправляется в виде строки: login#cmd#data
void send_to_server(int fd, std::string &login, std::string &cmd, std::string &data) {
    std::string str = login + "#" + cmd + "#" + data;
    size_t size = str.size();
    write(fd, &size, sizeof(size));
    write(fd, str.c_str(), sizeof(char) * size);
}

void send_to_client(int fd, std::string &message) {
    size_t size = message.size();
    write(fd, &size, sizeof(size));
    write(fd, message.c_str(), sizeof(char) * size);
}

std::string receive(int fd) {
    size_t size;
    read(fd, &size, sizeof(size_t));
    std::string recv;
    char c;
    for (size_t i = 0; i < size; ++i) {
        read(fd, &c, sizeof(char));
        if (c == '#') {
            recv += ": ";
        } else {
            recv.push_back(c);
        }
    }
    return recv;
}



void receive_from_client(int fd, std::string *login, std::string *cmd, std::string *data) {
    size_t size;
    read(fd, &size, sizeof(size));
    int k = 0;
    char c;
    for (size_t i = 0; i < size; ++i){
        read(fd, &c, sizeof(char));
        if (c == '#'){
            k++;
            continue;
        }
        switch (k) {
            case 0: {
                *login += c;
                break;
            }
            case 1: {
                *cmd += c;
                break;
            }
            case 2:{
                *data += c;
                break;
            }
        }
    }
}
```

```
void parse_game_word(std::string& message, std::string &game, std::string &word){
    int i = 0;
    for (auto item: message){
        if (item == '%'){
            ++i;
            continue;
        }
        (i == 0) ? game.push_back(item) : word.push_back(item);
    }
}
#endif //KP_FUNCTIONS_H
```

client.cpp

```
#include "functions.h"
#include <iostream>
#include <fcntl.h>
#include <thread>

inline void intro() {
    std::cout << "\t\t\tHello! Welcome to the game \"Bulls and cows\"." << std::endl;
    std::cout << "Have you already account in this game?\n Answer: \"yes\" or \"no\"" <<
std::endl;
    std::cout << "> ";
    std::cout.flush();
}

void func(int fd,const std::string& login){
    while (true){
        std::string respond = receive(fd);
        std::cout << "\n" << respond << std::endl;
        if (respond == "SERVER CLOSED"){
            exit(EXIT_SUCCESS);
        }
        std::cout << login << ">";
        std::cout.flush();
    }
}

std::pair<std::string, std::string> login() {
    std::string login, password;
    std::cout << "Please enter your login:\n> ";
    std::cout.flush();
    std::cin >> login;
    std::cout << "Please enter your password:\n> ";
    std::cout.flush();
    std::cin >> password;
    return std::make_pair(login, SECRET_KEY + password);
}



int connect_to_server() {
    int fd = open(SERVER_INPUT_FILE_NAME, O_RDWR);
    if (fd == -1) {
        std::cout << "Server doesn't exists" << std::endl;
        exit(EXIT_FAILURE);
    }
    return fd;
}

void menu() {
    std::cout << "\t\t Main menu" << std::endl;
    std::cout << "\t List of commands" << std::endl;
    std::cout << "1) rules" << std::endl;
    std::cout << "2) create $table_name $game_word" << std::endl;
    std::cout << "3) connect $table_name" << std::endl;
    std::cout << "4) quit" << std::endl;
}

void loading() {
    std::cout << "\t\tLoading" << std::endl;
    int width = 50;
    int progress = 0;
    while (progress <= 100) {
        int now = progress / 2;
        std::cout << "[";
        for (int i = 0; i < width; ++i) {
            if (i < now) { std::cout << "="; }
            else if (i == now) { std::cout << ">"; }
            else { std::cout << " "; }
        }
        std::cout << "] " << progress << "%" << std::endl;
        progress += 4;
        usleep(50000);
    }
}

void rules() {
    std::cout << "\t\tRules" << std::endl;
    std::cout << "1) One word is made up" << std::endl;
    std::cout << "2) Players try to guess the word" << std::endl;
    std::cout << "3) The server gives response like \"N bulls, M cows\"." << std::endl;
    std::cout << "4) Server's response means N chars are in the answer and are in their place" <<
std::endl;
```

```
    std::cout << "5) M chars are in the answer, but aren't in their place" << std::endl;
}


int main() {
    int client_out_fd = connect_to_server();
    intro();
    std::string answer;
    std::cin >> answer;
    bool ok = (answer == "yes");
    auto pair = login();
    std::string password = std::to_string(transform(pair).second);
    std::string login = pair.first;
    pair.second = ""; // "security"
    std::string cmd;
    ok ? cmd = "log" : cmd = "reg";
    send_to_server(client_out_fd, login, cmd, password);
    std::cout << "Connecting to server" << std::endl;
    loading();
    int fd = open(pair.first.c_str(), O_RDWR);
    if (fd == -1) {
        perror("Open file");
        exit(EXIT_FAILURE);
    }
    menu();
    std::string command, data, game, game_word;
    int game_fd;
    std::thread thread(func, fd, login);
    while (true) {
        std::cout << "> ";
        std::cout.flush();
        std::cin >> command;
        if (command == "rules") {
```

```
        rules();
    } else if (command == "create") {
        std::cin >> game >> game_word;
        data = game + "%" + game_word;
        send_to_server(client_out_fd, login, command, data);
    } else if (command == "connect") {
        std::cin >> game;
        data = game;
        loading();
        game_fd = open(("game_" + game).c_str(), O_RDWR);
        if (game_fd == -1) {
            perror("Open file");
            exit(EXIT_FAILURE);
        }
        send_to_server(game_fd, login, command, data);
        while (true) {
            std::cin >> command;
            if (command == "maybe"){
                std::cin >> data;
                send_to_server(game_fd, login, command, data);
            } else if (command == "leave"){
                send_to_server(game_fd, login, command, data);
                break;
            }
            std::cout << "> ";
            std::cout.flush();
        }
        menu();
    } else if (command == "quit") {
        data = "";
        send_to_server(client_out_fd, login, command, data);
        thread.detach();
```

```
            std::cout << "Thank you for game. Come to us later" << std::endl;
            exit(EXIT_SUCCESS);
        } else if (command == "shut_down" and login == ADMIN_NAME){
            data = "";
            send_to_server(client_out_fd, login, command, data);
            thread.detach();
            std::cout << "Server will be turned off" << std::endl;
            exit(EXIT_SUCCESS);
        }
    }
}
```

control_node.cpp

```
#include "functions.h"
#include <iostream>
#include <fstream>
#include <fcntl.h>
#include <unistd.h>
#include <sys/stat.h>
#include <map>
#include <thread>


void check_data_base() {
    std::ifstream file(DATA_BASE_FILE_NAME);
    file.open(DATA_BASE_FILE_NAME);
    if (file.is_open()) {
        std::cout << "OK: DATA BASE" << std::endl;
        file.close();
    } else {
        std::ofstream f(DATA_BASE_FILE_NAME);
        std::string empty_string;
        auto pair = std::make_pair(empty_string, SECRET_KEY + ADMIN_PASSWORD);
        std::string pass = std::to_string(transform(pair).second);
        pair.first = ADMIN_NAME;
        pair.second = pass;
        auto log_pas = transform(pair);
        f << log_pas.first << " " << log_pas.second << std::endl;
        f.close();
        std::cout << "OK: DATA BASE" << std::endl;
    }
}

void add_results(const std::string &game_name, const std::string &game_word, const std::string
&winner) {
    std::fstream file;
    file.open(RESULTS_FILE_NAME, std::ios::in | std::ios::out);
    if (file.is_open()) {
        file << game_name << "\t\t" << game_word << "\t\t" << winner << std::endl;
        file.close();
    } else {
        std::ofstream f(RESULTS_FILE_NAME);
        f << "Game name\t\tGame_word\t\tWinner" << std::endl;
        f << game_name << " " << game_word << " " << winner << std::endl;
        f.close();
    }
}

bool check_in_data_base(std::pair<std::string, std::string> &login_password) {
    std::pair<int, int> pair = transform(login_password);
    std::ifstream file;
    file.open(DATA_BASE_FILE_NAME);
    int log, pas;
    while (file >> log >> pas) {
        if (log == pair.first) {
            if (pas == pair.second) {
                file.close();
                return true;
            }
            file.close();
            return false;
        }
    }
    file.close();
    return false;
}

bool add_in_data_base(std::pair<std::string, std::string> &login_password) {
    std::pair<int, int> pair = transform(login_password);
    std::ifstream file;
    file.open(DATA_BASE_FILE_NAME, std::ios::in);
    int log, pas;
    while (file >> log >> pas) {
        if (log == pair.first) {
            file.close();
            return false;
        }
    }
    file.close();
    std::ofstream f(DATA_BASE_FILE_NAME, std::ios::app);
    f << pair.first << " " << pair.second << std::endl;
    f.close();
    return true;
}

int create_main_pipe() {
    if (mkfifo(SERVER_INPUT_FILE_NAME, ACCESS_PERMS) == -1) {
        perror("Mkfifo");
        exit(EXIT_FAILURE);
    }
    int fd = open(SERVER_INPUT_FILE_NAME, O_RDWR);
    if (fd == -1) {
        perror("Open file");
        exit(EXIT_FAILURE);
    }
    return fd;
}

int create_client_pipe(std::string &login) {
    if (mkfifo(login.c_str(), ACCESS_PERMS) == -1) {
        perror("Mkfifo");
        exit(EXIT_FAILURE);
    }
    int fd = open(login.c_str(), O_RDWR);
    if (fd == -1) {
        perror("open");
        exit(EXIT_FAILURE);
    }
    return fd;
}

int create_game_pipe(const std::string &game_name) {
    if (mkfifo(("game_" + game_name).c_str(), ACCESS_PERMS) == -1) {
        perror("mkfifo");
        exit(EXIT_FAILURE);
    }
    int fd = open(("game_" + game_name).c_str(), O_RDWR);
    if (fd == -1) {
        perror("open");
        exit(EXIT_FAILURE);
    }
    return fd;
}

int check(const std::string &game_word, const std::string &try_word, int &cows, int &bulls) {
    if (try_word.size() != game_word.size()) {
        return -1;
    }
    if (try_word == game_word) {
        return 1;
    }
    size_t size = game_word.size();
    for (size_t i = 0; i < size; ++i) {
        for (size_t j = 0; j < size; ++j) {
            if (game_word[i] == game_word[j]) {
                if (i == j) {
                    ++bulls;
                } else {
                    ++cows;
                }
            }
        }
    }
    return 0;
}

void game_func(const std::string &game_name, const std::string &game_word) {
    std::map<std::string, int> players_fd;
    int fd = create_game_pipe(game_name);
    int cows, bulls;
    std::string game_respond;
    int game_status;


    std::cout << "START GAME: " << game_name << std::endl;
    std::string login, cmd, data;
    while (true) {
        login = cmd = data = "";
        receive_from_client(fd, &login, &cmd, &data);
        if (cmd == "connect") {
            players_fd[login] = open(login.c_str(), O_RDWR);
            std::cout << "CLIENT " << login << " JOINED THE GAME: " << game_name << 
std::endl;
            game_respond =
                    "Welcome to the game " + game_name + "\n" + "Make your guesses with the 
command \"maybe $word\"";
            send_to_client(players_fd[login], game_respond);
        } else if (cmd == "maybe") {
            game_status = check(game_word, data, cows, bulls);
            if (game_status == -1) {
                game_respond = "Wrong string size";
                send_to_client(players_fd[login], game_respond);
            } else if (game_status == 1) {
                game_respond = "You won the game";
                send_to_client(players_fd[login], game_respond);
                add_results(game_name, game_word, login);
                game_respond = "Game is over. Winner is " + login + " . Game word is " + 
game_word + ".";
                for (const auto &it: players_fd) {
                    send_to_client(it.second, game_respond);
                    do {
                        std::string annoy = "Please leave the table (Use command \"leave\")";
                        send_to_client(it.second, annoy);
                        receive_from_client(fd, &login, &cmd, &data);
                    } while (cmd != "leave");
                    std::cout << "CLIENT " << it.first << "LEFT FROM THE TABLE" << std::endl;
                    players_fd.erase(it.first);
                }
                close(fd);
                std::cout << "FINISH GAME: " << game_name << std::endl;
                int mainFD = open("main_input", O_RDWR);
                game_respond = "finish";
                login = game_name;
                send_to_server(mainFD, login, game_respond, data);
                return;
            } else {
                game_respond = "Bulls: " + std::to_string(bulls) + " Cows: " + std::to_string(cows);
                send_to_client(players_fd[login], game_respond);
            }
        } else if (cmd == "leave") {
            players_fd.erase(login);
            std::cout << "CLIENT " << login << " LEFT FROM THE TABLE" << std::endl;
        }
    }
}


int main() {
    int fd_rec = create_main_pipe();
    std::map<std::string, int> logins_fds;
    std::vector<std::thread> games_threads;
    std::vector<std::string> games_names;
    std::string game_name, game_word;
    std::string login, cmd, data;
    check_data_base();
    while (true) {
        login = cmd = data = "";
```

```
receive_from_client(fd_rec, &login, &cmd, &data);
if (cmd == "log") {
    std::string password = data;
    std::pair<std::string, std::string> pair = std::make_pair(login, password);
    if (check_in_data_base(pair)) {
        std::cout << "NEW CLIENT: " << login << std::endl;
        logins_fds[login] = create_client_pipe(login);
    }
} else if (cmd == "reg") {
    std::string password = data;
    std::pair<std::string, std::string> pair = std::make_pair(login, password);
    if (add_in_data_base(pair)) {
        std::cout << "NEW USER: " << login << std::endl;
        logins_fds[login] = create_client_pipe(login);
    }
} else if (cmd == "create") {
    game_name = game_word = "";
    std::cout << data << std::endl;
    parse_game_word(data, game_name, game_word);
    std::cout << game_name << std::endl;
    games_names.emplace_back("game_" + game_name);
    games_threads.emplace_back(std::thread(game_func, game_name, game_word));
} else if (cmd == "quit") {
    close(logins_fds[login]);
    std::remove(login.c_str());
    logins_fds.erase(login);
    std::cout << "CLIENT " << login << " LEFT THE GAME" << std::endl;
} else if (cmd == "shut_down" and login == ADMIN_NAME) {
    std::string message = "SERVER CLOSED";
    for (auto &it: logins_fds) {
        send_to_client(it.second, message);
        close(it.second);
```

```
                remove(it.first.c_str());
            }
            for (size_t i = 0; i < games_threads.size(); ++i) {
                std::remove(games_names[i].c_str());
                games_threads[i].detach();
            }
            std::remove(SERVER_INPUT_FILE_NAME);
            std::cout << "SERVER OFF" << std::endl;
            exit(EXIT_SUCCESS);
        }
    }
}
}
```

Сборка программы

Сборка осуществляется при помощи утилиты cmake.

```
[yadroff@fedora src]$ cat CMakeLists.txt
cmake_minimum_required(VERSION 3.20)
project(KP)

set(CMAKE_THREAD_PREFER_PTHREAD TRUE)
set(THREADS_PREFER_PTHREAD_FLAG TRUE)
find_package(Threads REQUIRED)

add_executable(client client.cpp)
add_executable(server server.cpp)
target_link_libraries(client Threads::Threads)
target_link_libraries(server Threads::Threads)
[yadroff@fedora src]$ mkdir build
```

```
[yadroff@fedora src]$ cd build
[yadroff@fedora build]$ cmake ..
-- The C compiler identification is GNU 11.2.1
-- The CXX compiler identification is GNU 11.2.1
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Looking for pthread.h
-- Looking for pthread.h - found
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD - Success
-- Found Threads: TRUE
-- Configuring done
-- Generating done
-- Build files have been written to: /home/yadroff/CLionProjects/OS/KP/src/build
[yadroff@fedora build]$ make
[ 25%] Building CXX object CMakeFiles/client.dir/client.cpp.o
[ 50%] Linking CXX executable client
[ 50%] Built target client
[ 75%] Building CXX object CMakeFiles/server.dir/server.cpp.o
[100%] Linking CXX executable server
[100%] Built target server
```

Демонстрация работы программы

```
[yadroff@fedora build]$ ./server
OK: DATA BASE
NEW USER: a
NEW USER: b
NEW CLIENT: Yadroff
abcd%a
abcd
START GAME: abcd
CLIENT a JOINED THE GAME: abcd
CLIENT b JOINED THE GAME: abcd
cad%jk
cad
START GAME: cad
CLIENT Yadroff JOINED THE GAME: cad
CLIENT a LEFT THE GAME
CLIENT b LEFT THE GAME
CLIENT Yadroff LEFT THE GAME
NEW CLIENT: Yadroff
SERVER OFF
[yadroff@fedora build]$ ./client
                        Hello! Welcome to the game "Bulls and cows".
Have you already account in this game?
 Answer: "yes" or "no"
> no
Please enter your login:
> a
Please enter your password:
> a
Connecting to server
                Loading
[>                                        ] 0%
[==>                                      ] 4%
[====>                                    ] 8%
[======>                                  ] 12%
[========>                                ] 16%
[==========>                              ] 20%
[============>                            ] 24%
[==============>                          ] 28%
[================>                        ] 32%
[==================>                      ] 36%
[====================>                    ] 40%
```

```
[======================>                           ] 44%
[========================>                         ] 48%
[==========================>                       ] 52%
[============================>                     ] 56%
[==============================>                   ] 60%
[================================>                 ] 64%
[==================================>               ] 68%
[====================================>             ] 72%
[======================================>           ] 76%
[========================================>         ] 80%
[==========================================>       ] 84%
[============================================>     ] 88%
[==============================================>   ] 92%
[================================================> ] 96%
[==================================================] 100%
                Main menu
        List of commands
1) rules
2) create $table_name $game_word
3) connect $table_name
4) quit
> rules
                Rules
1) One word is made up
2) Players try to guess the word
3) The server gives response like "N bulls, M cows".
4) Server's response means N chars are in the answer and are in their place
5) M chars are in the answer, but aren't in their place
> create abcd a
> connect abcd
                Loading
[>                                                 ] 0%
[==>                                               ] 4%
[====>                                             ] 8%
[======>                                           ] 12%
[========>                                         ] 16%
[==========>                                       ] 20%
[============>                                     ] 24%
[==============>                                   ] 28%
[================>                                 ] 32%
[==================>                               ] 36%
[====================>                             ] 40%
```

```
[======================>                           ] 44%
[========================>                         ] 48%
[==========================>                       ] 52%
[============================>                     ] 56%
[==============================>                   ] 60%
[================================>                 ] 64%
[==================================>               ] 68%
[====================================>             ] 72%
[======================================>           ] 76%
[========================================>         ] 80%
[==========================================>       ] 84%
[============================================>     ] 88%
[==============================================>   ] 92%
[================================================> ] 96%
[==================================================] 100%

Welcome to the game abcd
Make your guesses with the command "maybe $word"
a>maybe bcd
>
Wrong string size
a>maybe b
>
Bulls: 1 Cows: 0
a>maybe c
>
Bulls: 2 Cows: 0
a>maybe a
>
You won the game
a>
Game is over. Winner is a . Game word is a.
a>
Please leave the table (Use command "leave")
a>leave
                Main menu
        List of commands
1) rules
2) create $table_name $game_word
3) connect $table_name

Please leave the table (Use command "leave")
```

```
a>4) quit
>
Please leave the table (Use command "leave")
a>
Please leave the table (Use command "leave")
a>leave
> rules
                Rules
1) One word is made up
2) Players try to guess the word
3) The server gives response like "N bulls, M cows".
4) Server's response means N chars are in the answer and are in their place
5) M chars are in the answer, but aren't in their place
> quit
Thank you for game. Come to us later
[yadroff@fedora build]$ ./client
                        Hello! Welcome to the game "Bulls and cows".
Have you already account in this game?
 Answer: "yes" or "no"
> no
Please enter your login:
> b
Please enter your password:
> abcd
Connecting to server
                Loading
[>                                        ] 0%
[==>                                        ] 4%
[====>                                        ] 8%
[======>                                        ] 12%
[========>                                        ] 16%
[==========>                                        ] 20%
[============>                                        ] 24%
[==============>                                        ] 28%
[================>                                        ] 32%
[==================>                                        ] 36%
[====================>                                        ] 40%
[======================>                                        ] 44%
[========================>                                        ] 48%
[==========================>                                        ] 52%
[============================>                                        ] 56%
[==============================>                                        ] 60%
```

```
[================================>                 ] 64%
[==================================>               ] 68%
[====================================>             ] 72%
[======================================>           ] 76%
[========================================>         ] 80%
[==========================================>       ] 84%
[============================================>     ] 88%
[==============================================>   ] 92%
[================================================> ] 96%
[==================================================] 100%
                Main menu
        List of commands
1) rules
2) create $table_name $game_word
3) connect $table_name
4) quit
> connect abcd
               Loading
[>                                                 ] 0%
[==>                                               ] 4%
[====>                                             ] 8%
[======>                                           ] 12%
[========>                                         ] 16%
[==========>                                       ] 20%
[============>                                     ] 24%
[==============>                                   ] 28%
[================>                                 ] 32%
[==================>                               ] 36%
[====================>                             ] 40%
[======================>                           ] 44%
[========================>                         ] 48%
[==========================>                       ] 52%
[============================>                     ] 56%
[==============================>                   ] 60%
[================================>                 ] 64%
[==================================>               ] 68%
[====================================>             ] 72%
[======================================>           ] 76%
[========================================>         ] 80%
[==========================================>       ] 84%
[============================================>     ] 88%
[==============================================>   ] 92%
```

```
[================================================> ] 96%
[==================================================] 100%

Welcome to the game abcd
Make your guesses with the command "maybe $word"
b>maybe b
> leave
                Main menu
        List of commands
1) rules
2) create $table_name $game_word
3) connect $table_name
4) quit
> quit
Thank you for game. Come to us later
[yadroff@fedora build]$ ./client
                        Hello! Welcome to the game "Bulls and cows".
Have you already account in this game?
 Answer: "yes" or "no"
> yes
Please enter your login:
> Yadroff
Please enter your password:
> qwerty
Connecting to server
               Loading
[>                                                 ] 0%
[==>                                               ] 4%
[====>                                             ] 8%
[======>                                           ] 12%
[========>                                         ] 16%
[==========>                                       ] 20%
[============>                                     ] 24%
[==============>                                   ] 28%
[================>                                 ] 32%
[==================>                               ] 36%
[====================>                             ] 40%
[======================>                           ] 44%
[========================>                         ] 48%
[==========================>                       ] 52%
[============================>                     ] 56%
[==============================>                   ] 60%
```

```
[================================>                 ] 64%
[==================================>               ] 68%
[====================================>             ] 72%
[======================================>           ] 76%
[========================================>         ] 80%
[==========================================>       ] 84%
[============================================>     ] 88%
[==============================================>   ] 92%
[================================================> ] 96%
[==================================================] 100%
                 Main menu
         List of commands
1) rules
2) create $table_name $game_word
3) connect $table_name
4) quit
> shut_down
Server will be turned off

SERVER CLOSED
```

Выводы

Именнованные пайпы хорошо справляются со своей задачей коммуникации между процессами. По моему скромному мнению это один из лучших способов коммуникации в виду его простоты. Был получен опыт разработки консольной клиент-серверной игры. Благодаря этому я понимаю, как происходит процесс общения между клиентом и сервером.