Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

Курсовой проект по курсу

«Операционные системы»

Группа: М80-206Б-22

Студент: Волков А.Д.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 11.01.2024

Москва, 2024

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.

Вариант

Вариант 12.
«Быки и коровы» (угадывать необходимо числа). Общение между сервером и клиентом необходимо организовать при помощи memory map. При создании каждой игры необходимо указывать количество игроков, которые будут участвовать. То есть угадывать могут несколько игроков. Должна быть реализована функция поиска игры, то есть игрок пытается войти в игру не по имени, а просто просит сервер найти ему игру.

Общий алгоритм

Игра включает в себя 3 программы: код клиента, код сервера и код менеджера сессий. Клиент, сервер и менеджер взаимодействуют с помощью memory mapping, а синхронизируются сигналами. Практически все сообщения клиента, сервера и менеджера имеют json структуру. Клиент работает на трех потоках: поток работы с сервером, поток для поимки сигнала о победителе в сессии и поток для поимки сигнала о том, что сервер и менеджер сессий завершили работу. Для начала клиенту предлагается ввести имя, потом команды для подключения к сессии или создания сессии. Потом начнется игра. В течении игры к сессии, могут подключаться новые игроки. Если какой-то игрок в сессии выиграл, то сессия закрывается, а игроки удаляются. Сервер отвечает на игровые сообщения клиентов и делает логи в формате json. Менеджер в свою очередь управляет сессиями, а именно создает, подключает к ним игроков и удаляет их. Также сервер контролирует количество игроков в сессии, чтобы оно не превышало заданное количество при ее создании.

Метод решения

Была использована сторонняя библиотека nlohmann/json (https://github.com/nlohmann/json) для удобства распределения информации в отображаемой памяти. Для начала запускается менеджер, который сразу отображает все необходимую для общения с клиентом информацию в память. Потом менеджер ждет команд от клиентов. Клиенты для начала должны ввести имя, потом ввести команду для создания или подключения к сессии. Эта команда определенным образом парсится и отображается в память. Менеджер сессий получает сигнал от клиента что данные о команде записаны в память и начинает их обрабатывать. Если не сработало никакое исключение, то менеджер дает разрешение на игру, если же срабатывает какое то исключение, то менеджер говорит об этом клиенту, и клиент должен ввести команду заново. Сервер же включается только при создании сессии, и для каждой сессии включается свой сервер. Когда создана сессия, то клиенты могут завершить свою работу, если завершит работу менеджер сессий. Далее сервера коммуницируют с клиентами, и когда в сессии кто-то победил, то сервер отправляет всем клиентам в сессии сигнал о том что кто-то победил, и клиенты завершают свою работу. После этого менеджер удаляет сессию из словаря сессий и дает возможность переиспользовать сессию с таким же названием.

Листинг программы

game_configuration.cpp

```
#pragma once

#include <bits/stdc++.h>
#include <nlohmann/json.hpp>
#include <iostream>
#include <sys/mman.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <semaphore.h>
#include <cstring>

#define GAME_FILENAME "game"
#define SERVER_PID_IN_MEMORY "server_pid"
#define MANAGER_PID_IN_MEMORY "manger_pid"
#define MANAGER_CLIENT_COMMUNICATION "manager"
#define MANAGER_WINNER_THREAD_PID_IN_MEMORY "manager_winner_pid"
#define MANAGER_WINNER_THREAD_CLIENT_COMMUNICATION "manager_winner"
#define SIZE 4096

struct Player {
    std::string name;
    int bulls = 0;
    int cows = 0;
    int supposition = 0;
    bool win = false;
    pid_t pid;
    std::string sess_name;
    Player(std::string _name) {
        name = _name;
        pid = getpid();
    }
};

std::ostream& operator<<(std::ostream& os, const Player& player) {
    os << "bulls: " << player.bulls << "\n";
    os << "cows: " << player.cows << "\n";
    return os;
}

struct Session {
    std::string session_name;
    std::vector<std::string> players_list;
    int max_players_quantity;
};

bool operator==(const Session& session1, const Session& session2) {
    return session1.session_name == session2.session_name;
}

int random_number() {
    srand(time(NULL));
    return rand() % 900 + 100;
}
```

```
std::string toup(std::string str) {
    std::transform(str.begin(), str.end(), str.begin(), toupper);
    return str;
}

void game(nlohmann::json& player_stats, std::string answer) {
    std::string supposition = std::to_string(player_stats["supposition"].get<int>());
    int cows = 0; int bulls = 0;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (answer[i] == supposition[j]) {
                if (i == j) {
                    bulls++;
                } else {
                    cows++;
                }
            }
        }
    }
    if (bulls == 3) {
        player_stats["win"] = true;
    }
    player_stats["bulls"] = bulls;
    player_stats["cows"] = cows;
}
```

client.cpp

```
#include "game_configuration.hpp"

static int flag = 1;
static int winner_flag = 1;
static int exit_flag = 1;

void signal_handler(int sugnum) {
    flag = 0;
}

void winner_signal_handler(int signum) {
    winner_flag = 0;
}

void exit_signal_handler(int signum) {
    exit_flag = 0;
}

void *exit_handler(void *arg) {
    signal(SIGINT, exit_signal_handler);
    while (exit_flag) {sleep(1);}
    std::cout << "\n";
    exit(0);
}

void *winner_handler(void *arg) {
    std::string *session = (std::string *) arg;
    signal(SIGUSR2, winner_signal_handler);
    while (winner_flag) {sleep(1);}
    int game_fd = shm_open((*session).c_str(), O_CREAT | O_RDWR, 0666);
```

```
    ftruncate(game_fd, SIZE);
    char *game_mmap = static_cast<char *>(mmap(NULL, SIZE, PROT_WRITE | PROT_READ,
MAP_SHARED, game_fd, 0));
    std::cout << "\n" << "[INFO] " << std::string(game_mmap) << "\n";
    sleep(1);
    munmap(game_mmap, SIZE);
    close(game_fd);
    shm_unlink((*session).c_str());
    int manth_fd = shm_open(MANAGER_WINNER_THREAD_PID_IN_MEMORY, O_CREAT | O_RDWR,
0666);
    ftruncate(manth_fd, SIZE);
    void *manth_mmap = mmap(NULL, SIZE, PROT_WRITE | PROT_READ, MAP_SHARED, manth_fd,
0);
    pid_t manth_pid = std::atoi((char *) manth_mmap);
    int man_fd = shm_open(MANAGER_WINNER_THREAD_CLIENT_COMMUNICATION, O_CREAT | O_RDWR,
0666);
    ftruncate(man_fd, SIZE);
    void *man_mmap = mmap(NULL, SIZE, PROT_WRITE | PROT_READ, MAP_SHARED, man_fd, 0);
    strcpy((char *) man_mmap, (const_cast<char *>((*session).c_str())));
    kill(manth_pid, SIGUSR2);
    exit(0);
}

void *client(void *arg) {
    sleep(1);
    Player player = *((Player *) arg);
    std::string pid_sess = std::string(SERVER_PID_IN_MEMORY) + player.sess_name;
    int pid_fd = shm_open(pid_sess.c_str(), O_CREAT | O_RDWR, 0666);
    ftruncate(pid_fd, SIZE);
    void *pid_mmap = mmap(NULL, SIZE, PROT_WRITE | PROT_READ, MAP_SHARED, pid_fd, 0);
    pid_t server_pid = std::atoi((char *) pid_mmap);
    int game_fd = shm_open(player.sess_name.c_str(), O_CREAT | O_RDWR, 0666);
    ftruncate(game_fd, SIZE);
    char *game_mmap = static_cast<char *>(mmap(NULL, SIZE, PROT_WRITE | PROT_READ,
MAP_SHARED, game_fd, 0));
    nlohmann::json stats {};
    stats["name"] = player.name;
    stats["session"] = player.sess_name;
    while (1) {
        int supposition;
        std::cout << "[CLIENT] Input your supposition: ";
        std::cin >> supposition;
        player.supposition = supposition;
        stats["bulls"] = player.bulls;
        stats["cows"] = player.cows;
        stats["supposition"] = player.supposition;
        stats["win"] = player.win;
        stats["pid"] = player.pid;
        std::cout << stats.dump() << "\n";
        strcpy(game_mmap, stats.dump().c_str());
        kill(server_pid, SIGUSR1);
        signal(SIGUSR1, signal_handler);
        while (flag) {sleep(1);}
        flag = 1;
        std::string message = std::string(game_mmap);
        stats = nlohmann::json::parse(message);
        player.bulls = stats["bulls"];
```

```
        player.cows = stats["cows"];
        player.win = stats["win"];
        std::cout << "\n[INFO] bulls: " << player.bulls << "\n";
        std::cout << "[INFO] cows: " << player.cows << "\n\n";
    }
    munmap(game_mmap, SIZE);
    close(game_fd);
    shm_unlink(player.sess_name.c_str());
}

int main() {
    int man_pid_fd = shm_open(MANAGER_PID_IN_MEMORY, O_CREAT | O_RDWR, 0666);
    ftruncate(man_pid_fd, SIZE);
    void *man_pid_mmap = mmap(NULL, SIZE, PROT_WRITE | PROT_READ, MAP_SHARED,
man_pid_fd, 0);
    pid_t man_pid = std::atoi((char *) man_pid_mmap);

    int man_fd = shm_open(MANAGER_CLIENT_COMMUNICATION, O_CREAT | O_RDWR, 0666);
    ftruncate(man_fd, SIZE);
    void *man_mmap = mmap(NULL, SIZE, PROT_WRITE | PROT_READ, MAP_SHARED, man_fd, 0);

    std::string name;
    std::cout << "[CLIENT] Input your name: ";
    std::cin >> name;
    Player player(name);
    std::cout << "\n[INFO] create [session name] [max players quantity]\n";
    std::cout << "[INFO] join [session name]\n";
    std::cout << "[INFO] find\n\n";
    std::string sess_name;
    while (true) {
        std::cout << "[COMMAND] ";
        std::string command;
        std::cin >> command;
        if (command == "create") {
            nlohmann::json request {};
            request["action"] = "create";
            request["player"] = name;
            request["pid"] = player.pid;
            std::string session_name;
            std::cin >> session_name;
            sess_name = session_name;
            request["name"] = session_name;
            int maxc; std::cin >> maxc;
            request["max"] = maxc;
            strcpy((char *) man_mmap, request.dump().c_str());
            kill(man_pid, SIGUSR1);
            signal(SIGUSR1, signal_handler);
            while (flag) {sleep(1);}
            flag = 1;
            nlohmann::json reply = nlohmann::json::parse(std::string((char
*)man_mmap));
            std::cout << "[INFO] " << reply["desc"].get<std::string>() << "\n\n";
            if (!reply["ok"]) {
                continue;
            } else {
                break;
            }
```

```
        } else if (command == "join") {
            nlohmann::json request {};
            request["action"] = "join";
            request["player"] = name;
            request["pid"] = player.pid;
            std::string session_name;
            std::cin >> session_name;
            sess_name = session_name;
            request["name"] = session_name;
            strcpy((char *) man_mmap, request.dump().c_str());
            kill(man_pid, SIGUSR1);
            signal(SIGUSR1, signal_handler);
            while (flag) {sleep(1);}
            flag = 1;
            nlohmann::json reply = nlohmann::json::parse(std::string((char
*)man_mmap));
            std::cout << "[INFO] " << reply["desc"].get<std::string>() << "\n\n";
            if (!reply["ok"]) {
                continue;
            } else {
                break;
            }
        } else if (command == "find") {
            nlohmann::json request {};
            request["action"] = "find";
            request["player"] = name;
            request["pid"] = player.pid;
            strcpy((char *) man_mmap, request.dump().c_str());
            kill(man_pid, SIGUSR1);
            signal(SIGUSR1, signal_handler);
            while (flag) {sleep(1);}
            flag = 1;
            nlohmann::json reply = nlohmann::json::parse(std::string((char
*)man_mmap));
            std::cout << "[INFO] " << reply["desc"].get<std::string>() << "\n\n";
            if (!reply["ok"]) {
                continue;
            } else {
                sess_name = reply["session"];
                break;
            }
        } else {
            std::cout << "[INFO] Incorrect command!\n";
        }
    }
    player.sess_name = sess_name;

    pthread_t client_thread, signal_thread, exit_thread;
    pthread_create(&client_thread, NULL, client, &player);
    pthread_create(&signal_thread, NULL, winner_handler, &sess_name);
    pthread_create(&exit_thread, NULL, exit_handler, NULL);
    pthread_join(client_thread, NULL);
    pthread_join(signal_thread, NULL);
    pthread_join(exit_thread, NULL);
}
```

server.cpp

```
#include "game_configuration.hpp"

static int flag = 1;

void signal_handler(int signum) {
    flag = 0;
}

bool member(std::vector<pid_t> arr, pid_t pid) {
    for (int i = 0; i < arr.size(); i++) {
        if (pid == arr[i]) {
            return true;
        }
    }
    return false;
}

int main(int argc, char *argv[]) {
    int answer = random_number();
    std::vector<pid_t> clients_pid;
    std::cout << "[" << toup(std::string(argv[0])) << "] Answer in session " <<
std::string(argv[0]) << " is " << answer << "\n\n";
    std::string pid_sess = std::string(SERVER_PID_IN_MEMORY) + std::string(argv[0]);
    int pid_fd = shm_open(pid_sess.c_str(), O_CREAT | O_RDWR, 0666);
    ftruncate(pid_fd, SIZE);
    void *pid_mmap = mmap(NULL, SIZE, PROT_WRITE | PROT_READ, MAP_SHARED, pid_fd, 0);
    strcpy((char *)pid_mmap, std::to_string(getpid()).c_str());
    munmap(pid_mmap, SIZE);
    close(pid_fd);
    int game_fd = shm_open(argv[0], O_CREAT | O_RDWR, 0666);
    ftruncate(game_fd, SIZE);
    char *game_mmap = static_cast<char *>(mmap(NULL, SIZE, PROT_WRITE | PROT_READ,
MAP_SHARED, game_fd, 0));
    while (1) {
        signal(SIGUSR1, signal_handler);
        while (flag) {sleep(1);}
        std::string message = std::string(game_mmap);
        std::cout << "[RECEIVED] " << message << "\n";
        nlohmann::json stats = nlohmann::json::parse(message);
        if (!member(clients_pid, stats["pid"])) {
            clients_pid.push_back(stats["pid"]);
        }
        strcpy(game_mmap, "");
        game(stats, std::to_string(answer));
        std::cout << "[SENT] " << stats.dump() << "\n\n";
        if (stats["win"] == true) {
            std::string reply = "Game over! Player " + stats["name"].get<std::string>()
+ " won!";
            strcpy(game_mmap, reply.c_str());
            for (auto pid : clients_pid) {
                kill(pid, SIGUSR2);
                flag = 1;
            }
        } else {
            strcpy(game_mmap, stats.dump().c_str());
            kill(stats["pid"], SIGUSR1);
```

```
            flag = 1;
        }
    }
    munmap(game_mmap, SIZE);
    close(game_fd);
    shm_unlink(argv[0]);
}
```

session_manager.cpp

```
#include "game_configuration.hpp"

static int flag = 1;
static int exit_flag = 1;
static int winner_flag = 1;

std::map<std::string, pid_t> players_map;
std::map<std::string, Session> sessions;

void signal_handler(int signum) {
    flag = 0;
}

void exit_handler(int signum) {
    exit_flag = 0;
    for (auto i : players_map) {
        kill(i.second, SIGINT);
    }
    std::cout << "\n";
    exit(0);
}

void winner_handler(int signum) {
    winner_flag = 0;
}

pid_t create_process() {
    pid_t pid = fork();
    if (-1 == pid) {
        perror("fork");
        exit(-1);
    }
    return pid;
}

bool member(const Session& session, std::vector<Session> arr) {
    for (int i = 0; i < arr.size(); i++) {
        if (session == arr[i]) {
            return true;
        }
    }
    return false;
}

bool not_full_sess(std::map<std::string, Session> sessions) {

    for (auto i : sessions) {
        if (i.second.players_list.size() < i.second.max_players_quantity) {
            return false;
```

```
        }
    }
    return true;
}

void *winner(void *arg) {
    int manth_fd = shm_open(MANAGER_WINNER_THREAD_PID_IN_MEMORY, O_CREAT | O_RDWR,
0666);
    ftruncate(manth_fd, SIZE);
    void *manth_pid_mmap = mmap(NULL, SIZE, PROT_READ | PROT_WRITE, MAP_SHARED,
manth_fd, 0);
    strcpy((char *)manth_pid_mmap, std::to_string(getpid()).c_str());
    munmap(manth_pid_mmap, SIZE);
    close(manth_fd);
    int man_fd = shm_open(MANAGER_WINNER_THREAD_CLIENT_COMMUNICATION, O_CREAT | O_RDWR,
0666);
    ftruncate(man_fd, SIZE);
    char *session_mmap = static_cast<char *>(mmap(NULL, SIZE, PROT_WRITE | PROT_READ,
MAP_SHARED, man_fd, 0));
    while (1) {
        signal(SIGUSR2, winner_handler);
        while (winner_flag) {sleep(1);}
        winner_flag = 1;
        auto sess = sessions.find(std::string(session_mmap));
        for (int i = 0; i < sess->second.players_list.size(); i++) {
            players_map.erase(sess->second.players_list[i]);
        }
        sessions.erase(std::string(session_mmap));
        std::cout << "[" << toup(std::string(session_mmap)) << "] Session " <<
std::string(session_mmap) << " was closed!\n\n";
    }
}

void *manager(void *arg) {
    int pid_fd = shm_open(MANAGER_PID_IN_MEMORY, O_CREAT | O_RDWR, 0666);
    ftruncate(pid_fd, SIZE);
    void *pid_mmap = mmap(NULL, SIZE, PROT_WRITE | PROT_READ, MAP_SHARED, pid_fd, 0);
    strcpy((char *)pid_mmap, std::to_string(getpid()).c_str());
    munmap(pid_mmap, SIZE);
    close(pid_fd);
    int fd = shm_open(MANAGER_CLIENT_COMMUNICATION, O_CREAT | O_RDWR, 0666);
    ftruncate(fd, SIZE);
    char *session_mmap = static_cast<char *>(mmap(NULL, SIZE, PROT_WRITE | PROT_READ,
MAP_SHARED, fd, 0));
    signal(SIGINT, exit_handler);
    while (exit_flag) {
        signal(SIGUSR1, signal_handler);
        while (flag) {sleep(1);}
        nlohmann::json session_action =
nlohmann::json::parse(std::string(session_mmap));
        if (players_map.find(session_action["player"]) == players_map.end()) {
            players_map[session_action["player"]] = session_action["pid"].get<pid_t>();
        }
        if (session_action["action"] == "create") {
            nlohmann::json reply {};
            if (sessions.find(session_action["name"].get<std::string>()) !=
sessions.end()) {
```

```
                reply["ok"] = false;
                reply["desc"] = "Session with the same name already exists!";
            } else {
                Session sess;
                sess.session_name = session_action["name"];
                sess.max_players_quantity = session_action["max"];

sess.players_list.push_back(session_action["player"].get<std::string>());
                sessions[sess.session_name] = sess;
                pid_t pid = create_process();
                reply["ok"] = true;
                reply["desc"] = "Session " + session_action["name"].get<std::string>()
+ " was created successfully!";
                if (pid == 0) {
                    char *argv[] = {const_cast<char*>(sess.session_name.c_str()), (char
*) NULL};
                    execv("server", argv);
                }
            }
            strcpy(session_mmap, reply.dump().c_str());
            kill(session_action["pid"], SIGUSR1);
            flag = 1;
        } else if (session_action["action"] == "join") {
            std::string session_name = session_action["name"];
            nlohmann::json reply {};
            if (sessions.find(session_name) == sessions.end()) {
                reply["ok"] = false;
                reply["desc"] = "Session not found!";
            } else {
                Session dest = sessions.find(session_name)->second;
                if (dest.players_list.size() == dest.max_players_quantity) {
                    reply["ok"] = false;
                    reply["desc"] = "Not more free places in session!";
                } else {
                    dest.players_list.push_back(session_action["player"]);
                    sessions.find(session_name)->second.players_list =
dest.players_list;
                    reply["ok"] = true;
                    reply["desc"] = "You successfully joined to session " +
session_action["name"].get<std::string>() + "!";
                }
            }
            strcpy(session_mmap, reply.dump().c_str());
            kill(session_action["pid"], SIGUSR1);
            flag = 1;
        } else if (session_action["action"] == "find") {
            nlohmann::json reply {};
            if (sessions.size() == 0 || not_full_sess(sessions)) {
                reply["ok"] = false;
                reply["desc"] = "Free sessions not found!";
            } else {
                for (auto& i : sessions) {
                    if (i.second.players_list.size() != i.second.max_players_quantity)
{
                        i.second.players_list.push_back(session_action["player"]);
                        reply["ok"] = true;
```

```
                        reply["desc"] = "You successfully joined to session " + i.first
+ "!";
                        reply["session"] = i.first;
                        break;
                    }
                }
            }
            strcpy(session_mmap, reply.dump().c_str());
            kill(session_action["pid"], SIGUSR1);
            flag = 1;
        }
    }
    munmap(session_mmap, SIZE);
    close(fd);
}

int main() {
    pthread_t manager_thread, win_thread;
    pthread_create(&manager_thread, NULL, manager, NULL);
    pthread_create(&win_thread, NULL, winner, NULL);
    pthread_join(manager_thread, NULL);
    pthread_join(win_thread, NULL);
}
```

Strace

```
$ strace -f ./client
execve("./client", ["./client"], 0x7fff21a54678 /* 83 vars */) = 0
brk(NULL)                               = 0x55eb90f9e000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe729900b0) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f0cbf181000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=57879, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 57879, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f0cbf172000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libstdc++.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=2260296, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 2275520, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f0cbee00000
mprotect(0x7f0cbee9a000, 1576960, PROT_NONE) = 0
mmap(0x7f0cbee9a000, 1118208, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x9a000) = 0x7f0cbee9a000
mmap(0x7f0cbefab000, 454656, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1ab000) = 0x7f0cbefab000
mmap(0x7f0cbf01b000, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x21a000) = 0x7f0cbf01b000
mmap(0x7f0cbf029000, 10432, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7f0cbf029000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=125488, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 127720, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f0cbf152000
mmap(0x7f0cbf155000, 94208, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x3000) = 0x7f0cbf155000
```

```
mmap(0x7f0cbf16c000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1a000)
= 0x7f0cbf16c000
mmap(0x7f0cbf170000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1d000) = 0x7f0cbf170000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0\0"..., 832) =
832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64)
= 784
pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48,
848) = 48
pread64(3,
"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\302\211\332Pq\2439\235\350\223\322\257\201\326\243\f"..
., 68, 896) = 68
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 0
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64)
= 784
mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f0cbea00000
mprotect(0x7f0cbea28000, 2023424, PROT_NONE) = 0
mmap(0x7f0cbea28000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x28000) = 0x7f0cbea28000
mmap(0x7f0cbebbd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1bd000) = 0x7f0cbebbd000
mmap(0x7f0cbec16000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x215000) = 0x7f0cbec16000
mmap(0x7f0cbec1c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7f0cbec1c000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=940560, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 942344, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f0cbf06b000
mmap(0x7f0cbf079000, 507904, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0xe000) = 0x7f0cbf079000
mmap(0x7f0cbf0f5000, 372736, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x8a000) = 0x7f0cbf0f5000
mmap(0x7f0cbf150000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0xe4000) = 0x7f0cbf150000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f0cbf069000
arch_prctl(ARCH_SET_FS, 0x7f0cbf06a3c0) = 0
set_tid_address(0x7f0cbf06a690)         = 29202
set_robust_list(0x7f0cbf06a6a0, 24)     = 0
rseq(0x7f0cbf06ad60, 0x20, 0, 0x53053053) = 0
mprotect(0x7f0cbec16000, 16384, PROT_READ) = 0
mprotect(0x7f0cbf150000, 4096, PROT_READ) = 0
mprotect(0x7f0cbf170000, 4096, PROT_READ) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f0cbf067000
mprotect(0x7f0cbf01b000, 45056, PROT_READ) = 0
mprotect(0x55eb909e3000, 4096, PROT_READ) = 0
mprotect(0x7f0cbf1bb000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7f0cbf172000, 57879)           = 0
getrandom("\x1b\xb1\x4f\xb7\x5b\x26\x3e\xe1", 8, GRND_NONBLOCK) = 8
```

```
brk(NULL)                               = 0x55eb90f9e000
brk(0x55eb90fbf000)                     = 0x55eb90fbf000
futex(0x7f0cbf02977c, FUTEX_WAKE_PRIVATE, 2147483647) = 0
openat(AT_FDCWD, "/dev/shm/manger_pid", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0666) = 3
ftruncate(3, 4096)                      = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f0cbf1ba000
openat(AT_FDCWD, "/dev/shm/manager", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0666) = 4
ftruncate(4, 4096)                      = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x7f0cbf180000
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...},
AT_EMPTY_PATH) = 0
write(1, "[CLIENT] Input your name: ", 26[CLIENT] Input your name: ) = 26
newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...},
AT_EMPTY_PATH) = 0
read(0, player1
"player1\n", 1024)              = 8
getpid()                                = 29202
write(1, "\n[INFO] create [session name] [m"..., 53
[INFO] create [session name] [max players quantity]
) = 53
write(1, "[INFO] join [session name]\n", 27[INFO] join [session name]
) = 27
write(1, "[INFO] find\n\n", 13[INFO] find

)         = 13
write(1, "[COMMAND] ", 10[COMMAND] )             = 10
read(0, create session1 1
"create session1 1\n", 1024)    = 18
kill(29170, SIGUSR1)                    = 0
rt_sigaction(SIGUSR1, {sa_handler=0x55eb909a05fd, sa_mask=[USR1],
sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x7f0cbea42520}, {sa_handler=SIG_DFL,
sa_mask=[], sa_flags=0}, 8) = 0
clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0}, {tv_sec=0, tv_nsec=3880555})
= ? ERESTART_RESTARTBLOCK (Interrupted by signal)
--- SIGUSR1 {si_signo=SIGUSR1, si_code=SI_USER, si_pid=29170, si_uid=1000} ---
rt_sigreturn({mask=[]})                 = -1 EINTR (Interrupted system call)
write(1, "[INFO] Session session1 was crea"..., 51[INFO] Session session1 was created
successfully!

) = 51
rt_sigaction(SIGRT_1, {sa_handler=0x7f0cbea91870, sa_mask=[],
sa_flags=SA_RESTORER|SA_ONSTACK|SA_RESTART|SA_SIGINFO, sa_restorer=0x7f0cbea42520},
NULL, 8) = 0
rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x7f0cbe1ff000
mprotect(0x7f0cbe200000, 8388608, PROT_READ|PROT_WRITE) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CL
ONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f0cbe9ff910,
parent_tid=0x7f0cbe9ff910, exit_signal=0, stack=0x7f0cbe1ff000, stack_size=0x7fff00,
tls=0x7f0cbe9ff640}strace: Process 29281 attached
 => {parent_tid=[29281]}, 88) = 29281
[pid 29202] rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
[pid 29281] rseq(0x7f0cbe9fffe0, 0x20, 0, 0x53053053 <unfinished ...>
[pid 29202] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0)
= 0x7f0cbd9fe000
```

```
[pid 29281] <... rseq resumed>)         = 0
[pid 29202] mprotect(0x7f0cbd9ff000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>
[pid 29281] set_robust_list(0x7f0cbe9ff920, 24 <unfinished ...>
[pid 29202] <... mprotect resumed>)     = 0
[pid 29202] rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0
[pid 29281] <... set_robust_list resumed>) = 0
[pid 29202]
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CL
ONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f0cbe1fe910,
parent_tid=0x7f0cbe1fe910, exit_signal=0, stack=0x7f0cbd9fe000, stack_size=0x7fff00,
tls=0x7f0cbe1fe640}strace: Process 29282 attached
 <unfinished ...>
[pid 29281] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid 29202] <... clone3 resumed> => {parent_tid=[29282]}, 88) = 29282
[pid 29282] rseq(0x7f0cbe1fefe0, 0x20, 0, 0x53053053 <unfinished ...>
[pid 29202] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid 29281] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid 29202] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid 29282] <... rseq resumed>)         = 0
[pid 29202] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0
<unfinished ...>
[pid 29282] set_robust_list(0x7f0cbe1fe920, 24 <unfinished ...>
[pid 29202] <... mmap resumed>)         = 0x7f0cbd1fd000
[pid 29282] <... set_robust_list resumed>) = 0
[pid 29281] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},  <unfinished ...>
[pid 29202] mprotect(0x7f0cbd1fe000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>
[pid 29282] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid 29202] <... mprotect resumed>)     = 0
[pid 29282] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid 29202] rt_sigprocmask(SIG_BLOCK, ~[],  <unfinished ...>
[pid 29282] rt_sigaction(SIGUSR2, {sa_handler=0x55eb909a0615, sa_mask=[USR2],
sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x7f0cbea42520},  <unfinished ...>
[pid 29202] <... rt_sigprocmask resumed>[], 8) = 0
[pid 29282] <... rt_sigaction resumed>{sa_handler=SIG_DFL, sa_mask=[], sa_flags=0}, 8)
= 0
[pid 29202]
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CL
ONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f0cbd9fd910,
parent_tid=0x7f0cbd9fd910, exit_signal=0, stack=0x7f0cbd1fd000, stack_size=0x7fff00,
tls=0x7f0cbd9fd640} <unfinished ...>
[pid 29282] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0}, strace: Process
29283 attached
 <unfinished ...>
[pid 29202] <... clone3 resumed> => {parent_tid=[29283]}, 88) = 29283
[pid 29283] rseq(0x7f0cbd9fdfe0, 0x20, 0, 0x53053053 <unfinished ...>
[pid 29202] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid 29283] <... rseq resumed>)         = 0
[pid 29202] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid 29283] set_robust_list(0x7f0cbd9fd920, 24 <unfinished ...>
[pid 29202] futex(0x7f0cbe9ff910, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 29281, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid 29283] <... set_robust_list resumed>) = 0
[pid 29283] rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
[pid 29283] rt_sigaction(SIGINT, {sa_handler=0x55eb909a062d, sa_mask=[INT],
sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x7f0cbea42520}, {sa_handler=SIG_DFL,
sa_mask=[], sa_flags=0}, 8) = 0
[pid 29283] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},  <unfinished ...>
```

```
[pid 29281] <... clock_nanosleep resumed>0x7f0cbe9fecf0) = 0
[pid 29282] <... clock_nanosleep resumed>0x7f0cbe1fdd80) = 0
[pid 29282] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},  <unfinished ...>
[pid 29281] mmap(NULL, 134217728, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_NORESERVE,
-1, 0 <unfinished ...>
[pid 29283] <... clock_nanosleep resumed>0x7f0cbd9fce00) = 0
[pid 29281] <... mmap resumed>)         = 0x7f0cb51fd000
[pid 29283] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},  <unfinished ...>
[pid 29281] munmap(0x7f0cb51fd000, 48246784) = 0
[pid 29281] munmap(0x7f0cbc000000, 18862080) = 0
[pid 29281] mprotect(0x7f0cb8000000, 135168, PROT_READ|PROT_WRITE) = 0
[pid 29281] openat(AT_FDCWD, "/dev/shm/server_pidsession1",
O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0666) = 5
[pid 29281] ftruncate(5, 4096)          = 0
[pid 29281] mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 5, 0) = 0x7f0cbf17f000
[pid 29281] openat(AT_FDCWD, "/dev/shm/session1", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC,
0666) = 6
[pid 29281] ftruncate(6, 4096)          = 0
[pid 29281] mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 6, 0) = 0x7f0cbf17e000
[pid 29281] write(1, "[CLIENT] Input your supposition:"..., 33[CLIENT] Input your
supposition: ) = 33
[pid 29281] read(0, 123
"123\n", 1024)      = 4
[pid 29281] kill(29280, SIGUSR1)        = 0
[pid 29281] rt_sigaction(SIGUSR1, {sa_handler=0x55eb909a05fd, sa_mask=[USR1],
sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x7f0cbea42520},
{sa_handler=0x55eb909a05fd, sa_mask=[USR1], sa_flags=SA_RESTORER|SA_RESTART,
sa_restorer=0x7f0cbea42520}, 8) = 0
[pid 29202] <... futex resumed>)        = ? ERESTARTSYS (To be restarted if SA_RESTART
is set)
[pid 29281] --- SIGUSR1 {si_signo=SIGUSR1, si_code=SI_USER, si_pid=29280, si_uid=1000}
---
[pid 29202] futex(0x7f0cbe9ff910, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 29281, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid 29281] rt_sigreturn({mask=[]})     = 0
[pid 29281] write(1, "\n", 1
)           = 1
[pid 29281] write(1, "[INFO] bulls: 0\n", 16[INFO] bulls: 0
) = 16
[pid 29281] write(1, "[INFO] cows: 1\n\n", 16[INFO] cows: 1

) = 16
[pid 29281] write(1, "[CLIENT] Input your supposition:"..., 33[CLIENT] Input your
supposition: ) = 33
[pid 29281] read(0,  <unfinished ...>
[pid 29282] <... clock_nanosleep resumed>0x7f0cbe1fdd80) = 0
[pid 29282] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},  <unfinished ...>
[pid 29283] <... clock_nanosleep resumed>0x7f0cbd9fce00) = 0
[pid 29283] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},  <unfinished ...>
[pid 29282] <... clock_nanosleep resumed>0x7f0cbe1fdd80) = 0
[pid 29282] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},  <unfinished ...>
[pid 29283] <... clock_nanosleep resumed>0x7f0cbd9fce00) = 0
[pid 29283] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0}, 0x7f0cbd9fce00) =
0
[pid 29282] <... clock_nanosleep resumed>0x7f0cbe1fdd80) = 0
[pid 29283] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},  <unfinished ...>
[pid 29282] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0}, 321
```

```
 <unfinished ...>
[pid 29281] <... read resumed>"321\n", 1024) = 4
[pid 29281] kill(29280, SIGUSR1)       = 0
[pid 29281] rt_sigaction(SIGUSR1, {sa_handler=0x55eb909a05fd, sa_mask=[USR1],
sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x7f0cbea42520},
{sa_handler=0x55eb909a05fd, sa_mask=[USR1], sa_flags=SA_RESTORER|SA_RESTART,
sa_restorer=0x7f0cbea42520}, 8) = 0
[pid 29202] <... futex resumed>)       = ? ERESTARTSYS (To be restarted if SA_RESTART
is set)
[pid 29202] --- SIGUSR1 {si_signo=SIGUSR1, si_code=SI_USER, si_pid=29280, si_uid=1000}
---
[pid 29281] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},  <unfinished ...>
[pid 29202] rt_sigreturn({mask=[]})     = 202
[pid 29202] futex(0x7f0cbe9ff910, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 29281, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid 29283] <... clock_nanosleep resumed>0x7f0cbd9fce00) = 0
[pid 29282] <... clock_nanosleep resumed>0x7f0cbe1fdd80) = 0
[pid 29283] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},  <unfinished ...>
[pid 29282] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},  <unfinished ...>
[pid 29281] <... clock_nanosleep resumed>0x7f0cbe9fecf0) = 0
[pid 29281] write(1, "\n", 1
)           = 1
[pid 29281] write(1, "[INFO] bulls: 1\n", 16[INFO] bulls: 1
) = 16
[pid 29281] write(1, "[INFO] cows: 0\n\n", 16[INFO] cows: 0

) = 16
[pid 29281] write(1, "[CLIENT] Input your supposition:"..., 33[CLIENT] Input your
supposition: ) = 33
[pid 29281] read(0,  <unfinished ...>
[pid 29283] <... clock_nanosleep resumed>0x7f0cbd9fce00) = 0
[pid 29282] <... clock_nanosleep resumed>0x7f0cbe1fdd80) = 0
[pid 29283] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},  <unfinished ...>
[pid 29282] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},  <unfinished ...>
[pid 29283] <... clock_nanosleep resumed>0x7f0cbd9fce00) = 0
[pid 29282] <... clock_nanosleep resumed>0x7f0cbe1fdd80) = 0
[pid 29283] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},  <unfinished ...>
[pid 29282] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},  <unfinished ...>
[pid 29283] <... clock_nanosleep resumed>0x7f0cbd9fce00) = 0
[pid 29282] <... clock_nanosleep resumed>0x7f0cbe1fdd80) = 0
[pid 29283] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},  <unfinished ...>
[pid 29282] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},  <unfinished ...>
[pid 29283] <... clock_nanosleep resumed>0x7f0cbd9fce00) = 0
[pid 29282] <... clock_nanosleep resumed>0x7f0cbe1fdd80) = 0
[pid 29283] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},  <unfinished ...>
[pid 29282] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},  <unfinished ...>
[pid 29283] <... clock_nanosleep resumed>0x7f0cbd9fce00) = 0
[pid 29282] <... clock_nanosleep resumed>0x7f0cbe1fdd80) = 0
[pid 29283] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},  <unfinished ...>
[pid 29282] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0}, 551 <unfinished
...>
[pid 29283] <... clock_nanosleep resumed>0x7f0cbd9fce00) = 0
[pid 29282] <... clock_nanosleep resumed>0x7f0cbe1fdd80) = 0
[pid 29283] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},  <unfinished ...>
[pid 29282] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},
 <unfinished ...>
[pid 29281] <... read resumed>"551\n", 1024) = 4
```

```
[pid 29281] kill(29280, SIGUSR1)        = 0
[pid 29281] rt_sigaction(SIGUSR1, {sa_handler=0x55eb909a05fd, sa_mask=[USR1],
sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x7f0cbea42520},  <unfinished ...>
[pid 29202] <... futex resumed>)        = ? ERESTARTSYS (To be restarted if SA_RESTART
is set)
[pid 29281] <... rt_sigaction resumed>{sa_handler=0x55eb909a05fd, sa_mask=[USR1],
sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x7f0cbea42520}, 8) = 0
[pid 29202] --- SIGUSR2 {si_signo=SIGUSR2, si_code=SI_USER, si_pid=29280, si_uid=1000}
---
[pid 29281] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},  <unfinished ...>
[pid 29202] rt_sigreturn({mask=[]})     = 202
[pid 29202] futex(0x7f0cbe9ff910, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 29281, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid 29283] <... clock_nanosleep resumed>0x7f0cbd9fce00) = 0
[pid 29282] <... clock_nanosleep resumed>0x7f0cbe1fdd80) = 0
[pid 29283] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},  <unfinished ...>
[pid 29282] openat(AT_FDCWD, "/dev/shm/session1", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC,
0666) = 7
[pid 29282] ftruncate(7, 4096)          = 0
[pid 29282] mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 7, 0) = 0x7f0cbf17d000
[pid 29282] write(1, "\n", 1
)           = 1
[pid 29282] mmap(NULL, 134217728, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_NORESERVE,
-1, 0) = 0x7f0cb0000000
[pid 29282] munmap(0x7f0cb4000000, 67108864) = 0
[pid 29282] mprotect(0x7f0cb0000000, 135168, PROT_READ|PROT_WRITE) = 0
[pid 29282] write(1, "[INFO] Game over! Player player1"..., 38[INFO] Game over! Player
player1 won!
) = 38
[pid 29282] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},  <unfinished ...>
[pid 29281] <... clock_nanosleep resumed>0x7f0cbe9fecf0) = 0
[pid 29281] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},  <unfinished ...>
[pid 29283] <... clock_nanosleep resumed>0x7f0cbd9fce00) = 0
[pid 29283] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},  <unfinished ...>
[pid 29282] <... clock_nanosleep resumed>0x7f0cbe1fdd80) = 0
[pid 29282] munmap(0x7f0cbf17d000, 4096) = 0
[pid 29282] close(7)                    = 0
[pid 29282] unlink("/dev/shm/session1") = 0
[pid 29282] openat(AT_FDCWD, "/dev/shm/manager_winner_pid",
O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0666) = 7
[pid 29282] ftruncate(7, 4096)          = 0
[pid 29282] mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 7, 0) = 0x7f0cbf17d000
[pid 29282] openat(AT_FDCWD, "/dev/shm/manager_winner",
O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0666) = 8
[pid 29282] ftruncate(8, 4096)          = 0
[pid 29282] mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 8, 0) = 0x7f0cbf17c000
[pid 29282] kill(29170, SIGUSR2)        = 0
[pid 29282] lseek(0, -1, SEEK_CUR)      = -1 ESPIPE (Illegal seek)
[pid 29282] exit_group(0)               = ?
[pid 29281] <... clock_nanosleep resumed> <unfinished ...>) = ?
[pid 29282] +++ exited with 0 +++
[pid 29283] <... clock_nanosleep resumed> <unfinished ...>) = ?
[pid 29202] <... futex resumed>)        = ?
[pid 29281] +++ exited with 0 +++
[pid 29283] +++ exited with 0 +++
+++ exited with 0 +++
```

Тесты

Менеджер сессий и сервер

```
$ ./manager
[SESSION1] Answer in session session1 is 261

[RECEIVED]
{"bulls":0,"cows":0,"name":"player1","pid":185125,"session":"session1","supposition":12
3,"win":false}
[SENT]
{"bulls":0,"cows":2,"name":"player1","pid":185125,"session":"session1","supposition":12
3,"win":false}

[RECEIVED]
{"bulls":0,"cows":0,"name":"player2","pid":185157,"session":"session1","supposition":12
3,"win":false}
[SENT]
{"bulls":0,"cows":2,"name":"player2","pid":185157,"session":"session1","supposition":12
3,"win":false}

[RECEIVED]
{"bulls":0,"cows":0,"name":"player3","pid":185174,"session":"session1","supposition":32
1,"win":false}
[SENT]
{"bulls":1,"cows":1,"name":"player3","pid":185174,"session":"session1","supposition":32
1,"win":false}

[RECEIVED]
{"bulls":0,"cows":2,"name":"player1","pid":185125,"session":"session1","supposition":23
7,"win":false}
[SENT]
{"bulls":1,"cows":0,"name":"player1","pid":185125,"session":"session1","supposition":23
7,"win":false}

[RECEIVED]
{"bulls":0,"cows":2,"name":"player2","pid":185157,"session":"session1","supposition":98
9,"win":false}
[SENT]
{"bulls":0,"cows":0,"name":"player2","pid":185157,"session":"session1","supposition":98
9,"win":false}

[RECEIVED]
{"bulls":1,"cows":1,"name":"player3","pid":185174,"session":"session1","supposition":26
1,"win":false}
[SENT]
{"bulls":3,"cows":0,"name":"player3","pid":185174,"session":"session1","supposition":26
1,"win":true}

[SESSION1] Session session1 was closed!

[SESSION2] Answer in session session2 is 324

[RECEIVED]
{"bulls":0,"cows":0,"name":"player4","pid":185460,"session":"session2","supposition":12
3,"win":false}
```

[SENT]
{"bulls":1,"cows":1,"name":"player4","pid":185460,"session":"session2","supposition":123,"win":false}

[RECEIVED]
{"bulls":1,"cows":1,"name":"player4","pid":185460,"session":"session2","supposition":321,"win":false}
[SENT]
{"bulls":2,"cows":0,"name":"player4","pid":185460,"session":"session2","supposition":321,"win":false}

[RECEIVED]
{"bulls":2,"cows":0,"name":"player4","pid":185460,"session":"session2","supposition":341,"win":false}
[SENT]
{"bulls":1,"cows":1,"name":"player4","pid":185460,"session":"session2","supposition":341,"win":false}

[RECEIVED]
{"bulls":1,"cows":1,"name":"player4","pid":185460,"session":"session2","supposition":325,"win":false}
[SENT]
{"bulls":2,"cows":0,"name":"player4","pid":185460,"session":"session2","supposition":325,"win":false}

[RECEIVED]
{"bulls":2,"cows":0,"name":"player4","pid":185460,"session":"session2","supposition":326,"win":false}
[SENT]
{"bulls":2,"cows":0,"name":"player4","pid":185460,"session":"session2","supposition":326,"win":false}

[RECEIVED]
{"bulls":2,"cows":0,"name":"player4","pid":185460,"session":"session2","supposition":327,"win":false}
[SENT]
{"bulls":2,"cows":0,"name":"player4","pid":185460,"session":"session2","supposition":327,"win":false}

[RECEIVED]
{"bulls":2,"cows":0,"name":"player4","pid":185460,"session":"session2","supposition":328,"win":false}
[SENT]
{"bulls":2,"cows":0,"name":"player4","pid":185460,"session":"session2","supposition":328,"win":false}

[RECEIVED]
{"bulls":2,"cows":0,"name":"player4","pid":185460,"session":"session2","supposition":329,"win":false}
[SENT]
{"bulls":2,"cows":0,"name":"player4","pid":185460,"session":"session2","supposition":329,"win":false}

[RECEIVED]
{"bulls":2,"cows":0,"name":"player4","pid":185460,"session":"session2","supposition":320,"win":false}

```
[SENT]
{"bulls":2,"cows":0,"name":"player4","pid":185460,"session":"session2","supposition":32
0,"win":false}

[RECEIVED]
{"bulls":2,"cows":0,"name":"player4","pid":185460,"session":"session2","supposition":32
4,"win":false}
[SENT]
{"bulls":3,"cows":0,"name":"player4","pid":185460,"session":"session2","supposition":32
4,"win":true}

[SESSION2] Session session2 was closed!
```

Игрок 1

```
$ ./client
[CLIENT] Input your name: player1

[INFO] create [session name] [max players quantity]
[INFO] join [session name]
[INFO] find

[COMMAND] create session1 3
[INFO] Session session1 was created successfully!

[CLIENT] Input your supposition: 123

[INFO] bulls: 0
[INFO] cows: 2

[CLIENT] Input your supposition: 237

[INFO] bulls: 1
[INFO] cows: 0

[CLIENT] Input your supposition:
[INFO] Game over! Player player3 won!
```

Игрок 2

```
$ ./client
[CLIENT] Input your name: player2

[INFO] create [session name] [max players quantity]
[INFO] join [session name]
[INFO] find

[COMMAND] join session1
[INFO] You successfully joined to session session1!

[CLIENT] Input your supposition: 123

[INFO] bulls: 0
[INFO] cows: 2

[CLIENT] Input your supposition: 989
```

```
[INFO] bulls: 0
[INFO] cows: 0

[CLIENT] Input your supposition:
[INFO] Game over! Player player3 won!
```

Игрок 3

```
$ ./client
[CLIENT] Input your name: player3

[INFO] create [session name] [max players quantity]
[INFO] join [session name]
[INFO] find

[COMMAND] find
[INFO] You successfully joined to session session1!

[CLIENT] Input your supposition: 321

[INFO] bulls: 1
[INFO] cows: 1

[CLIENT] Input your supposition: 261

[INFO] Game over! Player player3 won!
```

Игрок 4

```
$ ./client
[CLIENT] Input your name: player4

[INFO] create [session name] [max players quantity]
[INFO] join [session name]
[INFO] find

[COMMAND] find
[INFO] Free sessions not found!

[COMMAND] join session1
[INFO] Not more free places in session!

[COMMAND] create session2 1
[INFO] Session session2 was created successfully!

[CLIENT] Input your supposition: 123

[INFO] bulls: 1
[INFO] cows: 1

[CLIENT] Input your supposition: 321

[INFO] bulls: 2
```

```
[INFO] cows: 0

[CLIENT] Input your supposition: 341

[INFO] bulls: 1
[INFO] cows: 1

[CLIENT] Input your supposition: 325

[INFO] bulls: 2
[INFO] cows: 0

[CLIENT] Input your supposition: 326

[INFO] bulls: 2
[INFO] cows: 0

[CLIENT] Input your supposition: 327

[INFO] bulls: 2
[INFO] cows: 0

[CLIENT] Input your supposition: 328

[INFO] bulls: 2
[INFO] cows: 0

[CLIENT] Input your supposition: 329

[INFO] bulls: 2
[INFO] cows: 0

[CLIENT] Input your supposition: 320

[INFO] bulls: 2
[INFO] cows: 0

[CLIENT] Input your supposition: 324

[INFO] Game over! Player player4 won!
```

Вывод

В ходе выполнения курсового проекта я научился работать с сигналами, а также вспомнил основы работы с memory mapping. Также освежил знания по потокам и изучил клиент-серверную

архитектуру. Главной проблемой во время выполнения данного курсового проекта было проектирование программы. На начальных этапах иногда приходилось полностью писать заново всю программу и менять полностью схему ее работы. Но это дало мне полное понимание того как работает моя программа и по итогу я смог добавить несколько фишек, которые делают использование этой программы удобнее.