Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

Курсовая работа

«Клиент-серверная система для передачи мгновенных сообщений на memory map»

по курсу

«Операционные системы»

Группа: М80-206Б-22

Студент: Ларченко А.О.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 09.02.2024

Москва, 2024

Постановка задачи

Клиент-серверная система для передачи мгновенных сообщений. Базовый функционал должен быть следующим:

- Клиент может присоединиться к серверу, введя логин
- Клиент может отправить сообщение другому клиенту по его логину
- Клиент в реальном времени принимает сообщения от других клиентов

Вариант 24.

Необходимо предусмотреть возможность создания «групповых чатов». Связь между сервером и клиентом должна быть реализована при помощи memory map

Общий метод и алгоритм решения

Использованные системные вызовы:

- ***getpid***() - получение ID текущего процесса
- ***kill***(int pid, signal)- отправление сигнала signal процессу с ID pid
- ***signal***(int signum, sighandler_t handler) - устанавливает новый обработчик сигнала с номером _signum_ в соответствии с параметром _sighandler_, который может быть функцией пользователя
- ***shm_open***(const char *name) **-** создает и открывает новый (или уже существующий) объект разделяемой памяти POSIX. Объект разделяемой памяти POSIX - это обработчик, используемый несвязанными процессами для исполнения) на одну область разделяемой памяти.
- ***shm_unlink***(const char *name) **-** снимает объекты разделяемой памяти
- ***ftruncate***(int fd, off_t length) - устанавливают длину обычного файла с файловым дескриптором _fd_ в _length_ байт.
- ***mmap***(void *start, size_t length, int prot , int flags, int fd, off_t offset) - отражает файл fd в память отражает _length_ байтов, начиная со смещения _offset_ файла (или другого объекта), определенного файловым описателем _fd_, в память, начиная с адреса _start_. Настоящее местоположение отраженных данных возвращается самой функцией mmap, и никогда не бывает равным 0.
- ***munmap***(void *start, size_t length) - удаляет все отражения из заданной области памяти, после чего все ссылки на данную область будут вызывать ошибку "неправильное обращение к памяти" (invalid memory reference). Отражение удаляется автоматически при завершении процесса. С другой стороны, закрытие файла не приведет к снятию отражения.

Идея программы очень простая, также как и её реализация. Параллельно у нас запущены 1 сервер и произвольное количество клиентов(различных терминалов). Каждый пользователь вводит сообщение из предоставленного ему набора, после чего сообщение записывается в отображаемую память и серверу посылается сигнал. Сервер обрабатывает сообщение, если все хорошо,

пересылает его дальше по назначению, если же ошибка возвращает пользователю сообщение об ошибке.

Код программы

m_map.h

```
#pragma once

#include <iostream>

#include <unistd.h>

#include <sys/mman.h>

#include <sys/stat.h>

#include <fcntl.h>

#include "msg.h"



class Memory_map{

    public:

        int my_pid;

        int server_pid;

        Memory_map(string fn_wr, string fn_re, string fn_p);


        bool write_msg(Message &msg);

        bool read_msg(Message &msg);


        bool write_pid(Msg_pid &msg);

        bool read_pid(Msg_pid &msg);


        void close_sh_file(string fn);


        ~Memory_map();

        string fn_writing;

        string fn_reading;

        string fn_pid;

};
```

```
 void writing(char *to,const char *from, int size);
 string writing_to_str(char *from, int size);
```

m_map.c

```
#include "m_map.h"

void writing(char *to,const char *from, int size){ // *to and *from have equal capacity
    for(int i=0; i< size;++i){
        if(from[i]==EOF or from[i]=='\0' or from[i]=='\n'){
            to[i]='\0';
            break;
        }
        to[i]=from[i];
    }
    to[size-1]='\0';

}

 string writing_to_str(char *from, int size){
    string tmp;
    for(int i=0; i<size;++i){
        if(from[i]==EOF or from[i]=='\0' or from[i]=='\n'){
            tmp+='\0';
            break;
        }
        tmp+=from[i];
    }
    return tmp;
 }

Memory_map::Memory_map(string fn_wr, string fn_re, string fn_p){
    fn_writing=fn_wr;
    fn_reading=fn_re;
```

```
    fn_pid=fn_p;


}


Memory_map::~Memory_map(){


}


bool Memory_map::read_pid(Msg_pid &msg){
    int shm_fd=shm_open( fn_pid.c_str(), O_CREAT | O_RDWR, S_IRUSR | S_IWUSR);


    if(shm_fd==-1){
        if(shm_unlink( fn_pid.c_str())==-1){
            perror("munmap trouble: ");
            exit(-1);
        }
        close(shm_fd);
        return -1;
    }
    if(ftruncate(shm_fd, sizeof(Msg_pid))==-1){
        if(shm_unlink( fn_pid.c_str())==-1){
            perror("munmap trouble: ");
            exit(-1);
        }
        close(shm_fd);
        return -1;
    }
    Msg_pid *msg_ptr=(Msg_pid*)mmap(NULL, sizeof(Msg_pid), PROT_READ | PROT_WRITE,
MAP_SHARED, shm_fd, 0);
    if(msg_ptr==MAP_FAILED){
        if(shm_unlink( fn_pid.c_str())==-1){
            perror("munmap trouble: ");
            exit(-1);
        }
```

```
        close(shm_fd);
        return -1;
    }
    msg.Pid=msg_ptr->Pid;
    munmap(msg_ptr, sizeof(Msg_pid));
    close(shm_fd);
    return true;
}

bool Memory_map::write_pid(Msg_pid &msg){
    int shm_fd=shm_open( fn_pid.c_str(), O_CREAT | O_RDWR, S_IRUSR | S_IWUSR);

    if(shm_fd==-1){
        if(shm_unlink( fn_pid.c_str())==-1){
            perror("munmap trouble: ");
            exit(-1);
        }
        close(shm_fd);
        return -1;
    }
    if(ftruncate(shm_fd, sizeof(Msg_pid))==-1){
        if(shm_unlink( fn_pid.c_str())==-1){
            perror("munmap trouble: ");
            exit(-1);
        }
        close(shm_fd);
        return -1;
    }
    Msg_pid *msg_ptr=(Msg_pid*)mmap(NULL, sizeof(Msg_pid), PROT_READ | PROT_WRITE,
MAP_SHARED, shm_fd, 0);
    if(msg_ptr==MAP_FAILED){
        if(shm_unlink( fn_pid.c_str())==-1){
            perror("munmap trouble: ");
            exit(-1);
```

```
        }
        close(shm_fd);
        return -1;
    }
    msg_ptr->Pid=msg.Pid;
    munmap(msg_ptr, sizeof(Msg_pid));
    close(shm_fd);
    return true;
}

bool Memory_map::write_msg(Message &msg){
    int shm_fd=shm_open( fn_writing.c_str(), O_CREAT | O_RDWR, S_IRUSR | S_IWUSR);

    if(shm_fd==-1){
        if(shm_unlink( fn_writing.c_str())==-1){
            perror("munmap trouble: ");
            exit(-1);
        }
        close(shm_fd);
        return -1;
    }
    if(ftruncate(shm_fd, sizeof(Message))==-1){
        if(shm_unlink( fn_writing.c_str())==-1){
            perror("munmap trouble: ");
            exit(-1);
        }
        close(shm_fd);
        return -1;
    }
    Message *msg_ptr=(Message*)mmap(NULL, sizeof(Message), PROT_READ | PROT_WRITE,
MAP_SHARED, shm_fd, 0);
    if(msg_ptr==MAP_FAILED){
        if(shm_unlink( fn_writing.c_str())==-1){
            perror("munmap trouble: ");
```

```
            exit(-1);
        }
        close(shm_fd);
        return -1;
    }
    msg_ptr->type=msg.type;
    msg_ptr->pid=msg.pid;
    writing( msg_ptr->to, msg.to, NAMECAPACITY);
    writing(msg_ptr->usr_from, msg.usr_from, NAMECAPACITY);
    writing(msg_ptr->data, msg.data, DATACAPACITY);
    munmap(msg_ptr, sizeof(Message));
    close(shm_fd);
    return true;
}



bool Memory_map ::read_msg(Message &msg){
    int shm_fd=shm_open( fn_reading.c_str(), O_CREAT | O_RDWR, S_IRUSR | S_IWUSR);

    if(shm_fd==-1){
        if(shm_unlink( fn_reading.c_str())==-1){
            perror("munmap trouble: ");
            exit(-1);
        }
        close(shm_fd);
        return -1;
    }
    if(ftruncate(shm_fd, sizeof(Message))==-1){
        if(shm_unlink( fn_reading.c_str())==-1){
            perror("munmap trouble: ");
            exit(-1);
        }
        close(shm_fd);
```

```
        return -1;
    }
    Message *msg_ptr=(Message*)mmap(NULL, sizeof(Message), PROT_READ | PROT_WRITE,
MAP_SHARED, shm_fd, 0);
    if(msg_ptr==MAP_FAILED){
        if(shm_unlink( fn_reading.c_str())==-1){
            perror("munmap trouble: ");
            exit(-1);
        }
        close(shm_fd);
        return -1;
    }
    msg.type=msg_ptr->type;
    msg.pid=msg_ptr->pid;
    writing(msg.to, msg_ptr->to, NAMECAPACITY);
    writing(msg.usr_from, msg_ptr->usr_from, NAMECAPACITY);
    writing(msg.data, msg_ptr->data, DATACAPACITY);
    munmap(msg_ptr, sizeof(Message));
    close(shm_fd);
    return true;
}


void Memory_map::close_sh_file(string fn){
    if(shm_unlink(fn.c_str())==-1){
        perror("munmap trouble: ");
        exit(-1);
    }
}
```

client.cpp

```
#include <iostream>
#include "m_map.h"
```

```
using namespace std;

string _MSG_SEND = SH_OBJ_MSG_GET;
string _MSG_GET = SH_OBJ_MSG_SEND;
string _SERVER_PID = SH_OBJ_SERVER_PID;

pthread_mutex_t mutex;

static int wf=0;
static bool error_exit=false;

void wait_sig(int sig){
    wf=1;
}



bool check_name(string name){
    if(name.size()>NAMECAPACITY){
        return false;
    }
    for(int i=0;i<name.size();++i){
        if (name[i]=='\n' or name[i]==' '){
            return false;
        }
    }
    return true;
}

void* receiving(void* args){
    Memory_map *binder=(Memory_map*) args;
    // sleep(1);
    while(true){
        Message msg;
```

```
        // cout<<"Start waiting sig...\n";
        signal(SIGUSR1, wait_sig);
        // signal(SIGUSR1, wait_msg);
        while(wf!=1){
            sleep(0.1);
        }
        // cout<<"Got sig\n";
        // sleep(0.2);
        // pthread_mutex_lock(&mutex);
        binder->read_msg(msg);
        if(msg.type==message_type::_error){
            cout<<"Error:\n";
            cout<<msg.usr_from<<":"<<msg.data;
            error_exit=true;
            break;
        } else if(msg.type==message_type::_msg_to_chat){
            cout<<"New message:\n"<<"Chat:"<<msg.to<<":"<<msg.usr_from<<' '<<msg.data;
        } else{
            cout<<"New message:\n";
            cout<<msg.usr_from<<":"<<msg.data;
        }
        // fflush(NULL);
        // pthread_mutex_unlock(&mutex);
        wf=0;

    }

}

int main(int argc, char* argv[]){
    if(argc!=2){
        cout<<"Uncorrect input. Input your name!\n";
        exit(-1);
```

```
}
string username = argv[1];
if(!check_name(username)){
    cout<<"Uncorrect username\n";
    exit(-1);
}
Memory_map binder(_MSG_SEND,_MSG_GET, _SERVER_PID );
Msg_pid server_pid;
binder.read_pid(server_pid);
if(server_pid.Pid==0){
    cout<<"Server is unavailable now...\n";
    exit(-1);
}
int my_pid = getpid();
binder.my_pid=my_pid;
binder.server_pid=server_pid.Pid;

cout<<"Conecting...\n";
Message init;
writing(init.usr_from, username.c_str(), username.size()+1);
init.type=message_type::_create;
init.pid=my_pid;

// signal(SIGUSR1, wait_sig);
pthread_mutex_init(&mutex, NULL);
pthread_t receiver;
if(pthread_create(&receiver, NULL, receiving, &binder)!=0){
    perror("Create thread error ");
}

binder.write_msg(init);
kill(server_pid.Pid, SIGUSR1);
```

```
    // usleep(2000);
    sleep(2);

    pthread_mutex_lock(&mutex);
    cout<<"Server_pid: "<<server_pid.Pid<<'\n';
    cout<<"Welcome in our chat, "<<username<<"!\n";
    cout<<"Here you can communicate with other users directly or using chats\n\n";
    cout<<"--For reading messages write:\n"<<"To:'other_username' 'you message...'\n";
    cout<<"To:chat:'chat_name' 'you message...'\n"<<"-- Or if you want to join or creata
a chat write:\n";
    cout<<"Join:'chat_name'\n";
    cout<<"--Write 'q' to close terminal\n";
    pthread_mutex_unlock(&mutex);
    while(true){
        Message msg;
        msg.pid=my_pid;
        writing(msg.usr_from, username.c_str(), username.size()+1);

        if(error_exit){
            msg.type=message_type::_error;
            pthread_cancel(receiver);
            pthread_detach(receiver);
            binder.write_msg(msg);
            kill(server_pid.Pid, SIGUSR1);
            cout<<"Break;";
            break;
        }
        string request;
        cout<<" > ";
        getline(cin, request);
        bool currect=true;
        if(request=="q" or cin.eof()){
            msg.type=message_type::_exited;
            pthread_cancel(receiver);
```

```
        pthread_detach(receiver);
        binder.write_msg(msg);
        kill(server_pid.Pid, SIGUSR1);
        cout<<"Break;\n";
        break;
    } else if(request.substr(0, 8)=="To:chat:"){
        msg.type=message_type::_msg_to_chat;
        int pos=request.find(" ");
        string chat_rec=request.substr(8, pos-8);
        if(pos!= string::npos){
            string data_str=request.substr(pos+1, request.size()-(pos+1));
            if(data_str.size()==0){
                cout<<"Uncorrect input\n";
                currect=false;
            } else{
                writing(msg.to, chat_rec.c_str(), chat_rec.size()+1);
                writing(msg.data, data_str.c_str(), data_str.size()+1);
            }
        } else{
            cout<<"Uncorrect input\n";
            currect=false;
        }
    } else if(request.substr(0, 3)=="To:"){
        msg.type=message_type::_msg_to_usr;
        int pos=request.find(" ");
        string usr_rec=request.substr(3, pos-3);
        if(pos!= string::npos){
            string data_str=request.substr(pos+1, request.size()-(pos+1));
            if(data_str.size()==0){
                cout<<"Uncorrect input\n";
                currect=false;
            } else{
                writing(msg.to, usr_rec.c_str(), usr_rec.size()+1);
```

```
                    writing(msg.data, data_str.c_str(), data_str.size()+1);
                }
            } else{
                cout<<"Uncorrect input\n";
                currect=false;
            }
        } else if(request.substr(0, 5)=="Join:"){
            msg.type=message_type::_join_chat;
            string chat_name=request.substr(5, request.size()-5);
            if(chat_name.size()==0){
                cout<<"Uncorrect input\n";
                currect=false;
            } else{
                writing(msg.data, chat_name.c_str(), chat_name.size()+1);
            }
        }else if(request==""){
            cout<<'\n';
            currect=false;
        } else{
            cout<<"Uncorrect input\n";
            currect=false;
        }
        if(currect){
            binder.write_msg(msg);
            kill(server_pid.Pid, SIGUSR1);
            sleep(1);
        }
    }

}
```

server.cpp

```
#include <iostream>
#include <map>
#include <string>
#include <vector>

#include "m_map.h"
// #include "timer.h"

using namespace std;

string _MSG_SEND = SH_OBJ_MSG_SEND;
string _MSG_GET = SH_OBJ_MSG_GET;
string _SERVER_PID =SH_OBJ_SERVER_PID;



static int check=0;

void wait_read(int sig){
    check=1;
}

void exit_signal(int sig){
    cout<<"Forced exit\n";
    if(shm_unlink(_SERVER_PID.c_str())==-1){
        perror("munmap trouble: ");
        exit(-1);
    }
    if(shm_unlink(_MSG_SEND.c_str())==-1){
        perror("munmap trouble: ");
        exit(-1);
    }
    if(shm_unlink(_MSG_GET.c_str())==-1){
        perror("munmap trouble: ");
```

```
        exit(-1);
    }
    exit(1);
}

struct status{
    int pid;
    bool active;
};

bool is_active_server(map<string, status> &user_dict){
    int status=false;
    for(const auto& [user, user_stat]: user_dict){
        if(user_stat.active){
            status=true;
            break;
        }
    }
    return status;
}

bool is_in_chat(vector<string> &array, string str){
    for(int i=0; i<array.size();++i){
        if(array[i]==str){
            return true;
        }
    }
    return false;
}

void notify_all_in_chat(vector<string> &chat_members, string chat_name, map<string,
status> &user_dict){
    for(int i=0; i<chat_members.size();++i){
        int cur_pid=user_dict[chat_members[i]].pid;
```

```
        sleep(0.2);
        kill(cur_pid, SIGUSR1);
        cout<<"Notify: "<<cur_pid<<' '<<chat_members[i]<<'\n';
        sleep(0.2);
    }
}

int main(){
    map<string, vector<string>> chat_dict;
    map<string, status> user_dict;
    bool q_server=false;

    int my_pid=getpid();
    // cout<<"Server pid is "<<my_pid<<"\n";

    Memory_map binder(_MSG_SEND, _MSG_GET, _SERVER_PID);
    Msg_pid server_pid;
    // server_pid.Pid=my_pid;
    binder.read_pid(server_pid);
    if(server_pid.Pid!=0){
        cout<<"Server has already run. Exit\n";
        exit(-1);
    } else{
        cout<<"Server pid is "<<my_pid<<"\n";
        server_pid.Pid=my_pid;
    }

    binder.write_pid(server_pid);
    signal(SIGINT, exit_signal);
    while(true){
        Message msg;
        signal(SIGUSR1, wait_read);
        while(check!=1){
```

```
    sleep(0.1);
}
binder.read_msg(msg);
cout<<"New_msg: type: "<<msg.type<<" from: <"<<msg.usr_from<<">"<<msg.pid<<"\n";
check=0;

Message error_msg;
error_msg.type=message_type::_error;
writing(error_msg.to, msg.usr_from, NAMECAPACITY);

Message answ_msg;
answ_msg.type=message_type::_server_answer;
writing(answ_msg.to, msg.usr_from, NAMECAPACITY);
string answ;
const char* sender="Server\0";

string username=writing_to_str(msg.usr_from, NAMECAPACITY);
cout<<"To: "<<msg.to<<" Data: "<<msg.data<<"\n\n";

switch(msg.type){
    case message_type::_create :{
        sleep(0.1);
        // sender="Server\0";
        if(user_dict.count(username)>0){
            if(!user_dict[username].active){
                user_dict[username].pid=msg.pid;
                user_dict[username].active=true;
                // writing(answ_msg.to, msg.usr_from, NAMECAPACITY);
                const char *ok_answ="OK: Successful logined!\0";
                writing(answ_msg.data, ok_answ, 25);
            } else{
                const char *bad_answ="Error: User have already logined!\0";
                writing(error_msg.data, bad_answ, 35);
```

```
                    writing(error_msg.usr_from, sender, NAMECAPACITY);
                    binder.write_msg(error_msg);
                    kill(msg.pid, SIGUSR1);
                    cout<<"Send error msg\n";
                    break;
                }
            } else{
                user_dict[username].pid=msg.pid;
                user_dict[username].active=true;
                const char *ok_answ="OK: Successful created!\0";
                writing(answ_msg.data, ok_answ, 25);
            }
            writing(answ_msg.usr_from, sender, NAMECAPACITY);

            binder.write_msg(answ_msg);
            kill(user_dict[username].pid, SIGUSR1);
            cout<<"Sending msg:"<<"from sender: "<<sender<<" to "<<msg.pid<<'
'<<username<<'\n';
            cout<<"Data: "<<answ_msg.data<<'\n';

            break;
        }
        case message_type::_exited :{
            user_dict[username].active=false; //проверка на количество активных
узлов, если 0 - exit
            if(!is_active_server(user_dict)){
                q_server=true;
            }
            break;
        }
        case message_type::_join_chat :{
            string chat_name=writing_to_str(msg.data, NAMECAPACITY);
            // const char* data_msg;
            string data_msg_str;
```

```
if(chat_dict.count(chat_name)>0){
    if(is_in_chat(chat_dict[chat_name], username)){
        data_msg_str="Error: you have already been in this chat\0";
        writing(answ_msg.data, data_msg_str.c_str(), DATACAPACITY);
        writing(answ_msg.usr_from, sender, NAMECAPACITY);
        writing(answ_msg.to, msg.usr_from, NAMECAPACITY);
        answ_msg.pid=msg.pid;
        binder.write_msg(answ_msg);
        kill(answ_msg.pid, SIGUSR1);
        break;
    }
    //if not
    sleep(0.1);
    data_msg_str="New user has joined to chat.\0";
    writing(answ_msg.data, data_msg_str.c_str(), DATACAPACITY);
    answ_msg.type=message_type::_msg_to_chat;
    writing(answ_msg.usr_from, msg.usr_from, NAMECAPACITY);
    writing(answ_msg.to, msg.data, NAMECAPACITY);
    binder.write_msg(answ_msg);
    notify_all_in_chat(chat_dict[chat_name], chat_name, user_dict);
    // cout<<"=============_has joined to chat================\n";
    // cout<<"Data: "<<data_msg_str<<'\n';

} else{
    answ_msg.type=message_type::_msg_to_chat;
    data_msg_str="Chat has been created successfully.";
    writing(answ_msg.data, data_msg_str.c_str(), DATACAPACITY);
    writing(answ_msg.usr_from, msg.usr_from, NAMECAPACITY);
    writing(answ_msg.to, msg.data, NAMECAPACITY);
    answ_msg.pid=msg.pid;
    binder.write_msg(answ_msg);
    kill(answ_msg.pid, SIGUSR1);
    // cout<<"=============_has created the chat================\n";
```

```
                // cout<<"Data: "<<data_msg_str<<'\n';



            }
            cout<<"Sending msg:"<<"from sender: "<<answ_msg.usr_from<<" to "<<'
'<<msg.data<<'\n';
            cout<<"Data: "<<answ_msg.data<<'\n';
            chat_dict[chat_name].push_back(username);



            break;
        }
        case message_type::_msg_to_usr :{
            string usr_to=writing_to_str(msg.to, NAMECAPACITY);


            if(user_dict.count(usr_to)>0){
                if(user_dict[usr_to].active){
                    writing(answ_msg.usr_from, msg.usr_from, NAMECAPACITY);
                    writing(answ_msg.data, msg.data, DATACAPACITY);
                    writing(answ_msg.to, msg.to, NAMECAPACITY);
                    answ_msg.pid=user_dict[usr_to].pid;
                } else{
                    const char* data_msg="Error: User is inactive now";
                    writing(answ_msg.to, msg.usr_from, NAMECAPACITY);
                    writing(answ_msg.usr_from, sender, NAMECAPACITY);
                    writing(answ_msg.data, data_msg, 28);
                    answ_msg.pid=msg.pid;
                }
            } else{
                const char* data_msg="Error: No such user";
                writing(answ_msg.to, msg.usr_from, NAMECAPACITY);
                writing(answ_msg.usr_from, sender, NAMECAPACITY);
                writing(answ_msg.data, data_msg, 20);
                answ_msg.pid=msg.pid;
```

```
        }
        binder.write_msg(answ_msg);
        kill(answ_msg.pid, SIGUSR1);
        break;
    }
    case message_type::_msg_to_chat :{
        string chat_name=writing_to_str(msg.to, NAMECAPACITY);
        string data_msg_str;
        if(chat_dict.count(chat_name)>0){
            if(is_in_chat(chat_dict[chat_name], username)){
                sleep(0.1);
                writing(answ_msg.data, msg.data, DATACAPACITY);
                answ_msg.type=message_type::_msg_to_chat;
                writing(answ_msg.usr_from, msg.usr_from, NAMECAPACITY);
                writing(answ_msg.to, msg.to, NAMECAPACITY);
                binder.write_msg(answ_msg);
                notify_all_in_chat(chat_dict[chat_name], chat_name, user_dict);
            } else{
                data_msg_str="Error: You aren't in this chat\0";
                writing(answ_msg.data, data_msg_str.c_str(), DATACAPACITY);
                writing(answ_msg.usr_from, sender, NAMECAPACITY);
                writing(answ_msg.to, msg.usr_from, NAMECAPACITY);
                answ_msg.pid=msg.pid;
                binder.write_msg(answ_msg);
                kill(answ_msg.pid, SIGUSR1);
            }
        } else{
            data_msg_str="Error: No such chat exist\0";
            writing(answ_msg.data, data_msg_str.c_str(), DATACAPACITY);
            writing(answ_msg.usr_from, sender, NAMECAPACITY);
            writing(answ_msg.to, msg.usr_from, NAMECAPACITY);
            answ_msg.pid=msg.pid;
```

```
                binder.write_msg(answ_msg);
                kill(answ_msg.pid, SIGUSR1);
            }
            break;
        }
        case message_type::_error:{
            cout<<"<"<<msg.usr_from<<">"<<" was exit with error\n";
            break;
        }
        default:
            break;

    }
    if(q_server){
        cout<<"Close server. There are no active users\n";
        break;
    }
    cout<<"Current number of users: "<<user_dict.size()<<"\n\n";
}


binder.close_sh_file(binder.fn_pid);
binder.close_sh_file(binder.fn_reading);
binder.close_sh_file(binder.fn_writing);
}
```

Протокол работы программы

Тестирование:

<u>Терминал 1(Сервер):</u>

arsenii@PC-Larcha14:~/Documents/VS_code_prog/OSI/KP$ ./server
Server pid is 45486
New_msg: type: 0 from: <Arsenii>45489

```
To:  Data:

Sending msg:from sender: Server to 45489 Arsenii
Data: OK: Successful created!
Current number of users: 1

New_msg: type: 4 from: <Arsenii>45489
 Data:

Close server. There are no active users
arsenii@PC-Larcha14:~/Documents/VS_code_prog/OSI/KP$ make
g++  m_map.cpp  server.cpp -o server
g++  m_map.cpp client.cpp  -o client
client.cpp: In function ‘void* receiving(void*)’:
client.cpp:64:1: warning: no return statement in function returning non-void [-Wreturn-type]
   64 | }
      | ^
arsenii@PC-Larcha14:~/Documents/VS_code_prog/OSI/KP$ ./server
Server pid is 51339
New_msg: type: 0 from: <Arsenii>51340
To:  Data:

Sending msg:from sender: Server to 51340 Arsenii
Data: OK: Successful created!
Current number of users: 1

New_msg: type: 0 from: <Gg>51344
To:  Data:

Sending msg:from sender: Server to 51344 Gg
Data: OK: Successful created!
Current number of users: 2

New_msg: type: 2 from: <Gg>51344
To: Arsenii Data: HI!

Current number of users: 2

New_msg: type: 2 from: <Gg>51344
To: Arsenii Data: How are you&

Current number of users: 2

New_msg: type: 2 from: <Arsenii>51340
To: Gg Data: Hi! Nice I've ended my kp!!!!
```

Current number of users: 2

New_msg: type: 2 from: <Arsenii>51340
To: OO Data: Hi

Current number of users: 2

New_msg: type: 1 from: <Arsenii>51340
To: OO Data: new_chat

Sending msg:from sender: Arsenii to new_chat
Data: Chat has been created successfully.
Current number of users: 2

New_msg: type: 3 from: <Gg>51344
To: new_chat Data: ggggggggggggggg

Current number of users: 2

New_msg: type: 1 from: <Gg>51344
To: new_chat Data: new_chat

Notify: 51340 Arsenii
Sending msg:from sender: Gg to new_chat
Data: New user has joined to chat.
Current number of users: 2

New_msg: type: 2 from: <Gg>51344
To: new_chat Data: Hi

Current number of users: 2

New_msg: type: 3 from: <Gg>51344
To: new_chat Data: Hi

Notify: 51340 Arsenii
Notify: 51344 Gg
Current number of users: 2

New_msg: type: 4 from: <Arsenii>51340
To: OO Data: new_chat

Current number of users: 2

New_msg: type: 4 from: <Gg>51344
To: new_chat Data: Hi

Close server. There are no active users

<u>Терминал 2(Клиент):</u>

arsenii@PC-Larcha14:~/Documents/VS_code_prog/OSI/KP$ ./client Arsenii
Conecting...
Server_pid: 42308
Welcome in our chat, Arsenii!
Here you can communicate with other users directly or using chats

--For reading messages write:
To:'other_username' 'you message...'
To:chat:'chat_name' 'you message...'
-- Or if you want to join or creata a chat write:
Join:'chat_name'
--Write 'q' to close terminal
> New message:

Server:OK: Successful created!
> q
Break;
arsenii@PC-Larcha14:~/Documents/VS_code_prog/OSI/KP$ ./client Arsenii
Server is unavailable now...
arsenii@PC-Larcha14:~/Documents/VS_code_prog/OSI/KP$ ./client Arsenii
Conecting...
Server_pid: 51339
Welcome in our chat, Arsenii!
Here you can communicate with other users directly or using chats

--For reading messages write:
To:'other_username' 'you message...'
To:chat:'chat_name' 'you message...'
-- Or if you want to join or creata a chat write:
Join:'chat_name'
--Write 'q' to close terminal
> New message:

Server:OK: Successful created!
> New message:

Gg:HI!
> New message:

Gg:How are you&
> To:Gg Hi! Nice I've ended my kp!!!!

> To:OO Hi
> New message:

Server:Error: No such user
> To:Gg
Uncorrect input
>

> Join:new_chat
> New message:

Chat:new_chat:Arsenii Chat has been created successfully.
> New message:

Chat:new_chat:Gg New user has joined to chat.
>

> New message:

Chat:new_chat:Gg Hi
> q
Break;

Терминал 3(Клиент):
arsenii@PC-Larcha14:~/Documents/VS_code_prog/OSI/KP$ ./client Gg
Conecting...
Server_pid: 51339
Welcome in our chat, Gg!
Here you can communicate with other users directly or using chats

--For reading messages write:
To:'other_username' 'you message...'
To:chat:'chat_name' 'you message...'
-- Or if you want to join or creata a chat write:
Join:'chat_name'
--Write 'q' to close terminal
> New message:

Server:OK: Successful created!
> To:Arsenii HI!
> To:Arsenii How are you&
> New message:

Arsenii:Hi! Nice I've ended my kp!!!!
> To:chat:new_chat

Uncorrect input
>

> To:chat:new_chat gggggggggggggg
> New message:

Server:Error: You aren't in this chat
> Join:new_chat
> To:new_chat Hi
> New message:

Server:Error: No such user
> To:chat:new_chat Hi
> New message:

Chat:new_chat:Gg Hi
> q
Break;

==========

Strace:

server: arsenii@PC-Larcha14:~/Documents/VS_code_prog/OSI/KP$ strace -e trace=\!clock_nanosleep -oserver_log.log ./server

execve("./server", ["./server"], 0x7fff5a10a308 /* 56 vars */) = 0

brk(NULL) = 0x5628ba8b3000

arch_prctl(0x3001 /* ARCH_??? */, 0x7ffdee8f3440) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f7cc5e03000

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=81715, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 81715, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f7cc5def000

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"..., 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) = 0x7f7cc5a00000

mprotect(0x7f7cc5a9a000, 1576960, PROT_NONE) = 0

mmap(0x7f7cc5a9a000, 1118208, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x9a000) = 0x7f7cc5a9a000

mmap(0x7f7cc5bab000, 454656, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1ab000) = 0x7f7cc5bab000

mmap(0x7f7cc5c1b000, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x21a000) = 0x7f7cc5c1b000

mmap(0x7f7cc5c29000, 10432, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f7cc5c29000

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"..., 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) = 0x7f7cc5dcf000

mmap(0x7f7cc5dd2000, 94208, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x7f7cc5dd2000

mmap(0x7f7cc5de9000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1a000) = 0x7f7cc5de9000

mmap(0x7f7cc5ded000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1d000) = 0x7f7cc5ded000

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 =\340\2563\265?\356\25x\261\27\313A#\350"..., 68, 896) = 68

newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2216304, ...}, 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, 2260560, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f7cc5600000

mmap(0x7f7cc5628000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f7cc5628000

mmap(0x7f7cc57bd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7f7cc57bd000

mmap(0x7f7cc5815000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x214000) = 0x7f7cc5815000

mmap(0x7f7cc581b000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f7cc581b000

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\3\0>\0\1\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) = 0x7f7cc5ce8000

mmap(0x7f7cc5cf6000, 507904, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe000) = 0x7f7cc5cf6000

mmap(0x7f7cc5d72000, 372736, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x8a000) = 0x7f7cc5d72000

mmap(0x7f7cc5dcd000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe4000) = 0x7f7cc5dcd000

close(3) = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f7cc5ce6000

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f7cc5ce3000

arch_prctl(ARCH_SET_FS, 0x7f7cc5ce3740) = 0

set_tid_address(0x7f7cc5ce3a10) = 52571

set_robust_list(0x7f7cc5ce3a20, 24) = 0

rseq(0x7f7cc5ce40e0, 0x20, 0, 0x53053053) = 0

mprotect(0x7f7cc5815000, 16384, PROT_READ) = 0

mprotect(0x7f7cc5dcd000, 4096, PROT_READ) = 0

mprotect(0x7f7cc5ded000, 4096, PROT_READ) = 0

mprotect(0x7f7cc5c1b000, 45056, PROT_READ) = 0

mprotect(0x5628ba3ee000, 4096, PROT_READ) = 0

```
mprotect(0x7f7cc5e3d000, 8192, PROT_READ) = 0

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) =
0

munmap(0x7f7cc5def000, 81715)          = 0

getrandom("\xe1\x0e\x4b\x46\x2a\xe2\x5a\xd6", 8, GRND_NONBLOCK) = 8

brk(NULL)                              = 0x5628ba8b3000

brk(0x5628ba8d4000)                    = 0x5628ba8d4000

futex(0x7f7cc5c2977c, FUTEX_WAKE_PRIVATE, 2147483647) = 0

getpid()                               = 52571

openat(AT_FDCWD, "/dev/shm/myserver_pid",
O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0600) = 3

ftruncate(3, 4)                        = 0

mmap(NULL, 4, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f7cc5e3c000

munmap(0x7f7cc5e3c000, 4)              = 0

close(3)                               = 0

newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}, AT_EMPTY_PATH)
= 0

write(1, "Server pid is 52571\n", 20)   = 20

openat(AT_FDCWD, "/dev/shm/myserver_pid",
O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0600) = 3

ftruncate(3, 4)                        = 0

mmap(NULL, 4, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f7cc5e3c000

munmap(0x7f7cc5e3c000, 4)              = 0

close(3)                               = 0

rt_sigaction(SIGINT, {sa_handler=0x5628ba3e3333, sa_mask=[INT],
sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x7f7cc5642520}, {sa_handler=SIG_DFL,
sa_mask=[], sa_flags=0}, 8) = 0

rt_sigaction(SIGUSR1, {sa_handler=0x5628ba3e331b, sa_mask=[USR1],
sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x7f7cc5642520}, {sa_handler=SIG_DFL,
sa_mask=[], sa_flags=0}, 8) = 0

--- SIGUSR1 {si_signo=SIGUSR1, si_code=SI_USER, si_pid=52579, si_uid=1000} ---

rt_sigreturn({mask=[]})                = -1 EINTR (Interrupted system call)
```

openat(AT_FDCWD, "/dev/shm/reader", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0600) = 3

ftruncate(3, 368) = 0

mmap(NULL, 368, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f7cc5e3c000

munmap(0x7f7cc5e3c000, 368) = 0

close(3) = 0

write(1, "New_msg: type: 0 from: <Arsenii>"..., 38) = 38

write(1, "To: \7 Data: \n\n", 14) = 14

openat(AT_FDCWD, "/dev/shm/writer", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0600) = 3

ftruncate(3, 368) = 0

mmap(NULL, 368, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f7cc5e3c000

munmap(0x7f7cc5e3c000, 368) = 0

close(3) = 0

kill(52579, SIGUSR1) = 0

write(1, "Sending msg:from sender: Server "..., 50) = 50

write(1, "Data: OK: Successful created!\n", 30) = 30

write(1, "Current number of users: 1\n\n", 28) = 28

rt_sigaction(SIGUSR1, {sa_handler=0x5628ba3e331b, sa_mask=[USR1], sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x7f7cc5642520}, {sa_handler=0x5628ba3e331b, sa_mask=[USR1], sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x7f7cc5642520}, 8) = 0

--- SIGUSR1 {si_signo=SIGUSR1, si_code=SI_USER, si_pid=52587, si_uid=1000} ---

rt_sigreturn({mask=[]}) = -1 EINTR (Interrupted system call)

openat(AT_FDCWD, "/dev/shm/reader", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0600) = 3

ftruncate(3, 368) = 0

mmap(NULL, 368, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f7cc5e3c000

munmap(0x7f7cc5e3c000, 368) = 0

close(3) = 0

write(1, "New_msg: type: 0 from: <Gg>52587"..., 33) = 33

write(1, "To: \7 Data: \n\n", 14) = 14

openat(AT_FDCWD, "/dev/shm/writer", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0600) = 3

ftruncate(3, 368) = 0

mmap(NULL, 368, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f7cc5e3c000

munmap(0x7f7cc5e3c000, 368) = 0

close(3) = 0

kill(52587, SIGUSR1) = 0

write(1, "Sending msg:from sender: Server "..., 45) = 45

write(1, "Data: OK: Successful created!\n", 30) = 30

write(1, "Current number of users: 2\n\n", 28) = 28

rt_sigaction(SIGUSR1, {sa_handler=0x5628ba3e331b, sa_mask=[USR1], sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x7f7cc5642520}, {sa_handler=0x5628ba3e331b, sa_mask=[USR1], sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x7f7cc5642520}, 8) = 0

--- SIGUSR1 {si_signo=SIGUSR1, si_code=SI_USER, si_pid=52579, si_uid=1000} ---

rt_sigreturn({mask=[]}) = -1 EINTR (Interrupted system call)

openat(AT_FDCWD, "/dev/shm/reader", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0600) = 3

ftruncate(3, 368) = 0

mmap(NULL, 368, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f7cc5e3c000

munmap(0x7f7cc5e3c000, 368) = 0

close(3) = 0

write(1, "New_msg: type: 2 from: <Arsenii>"..., 38) = 38

write(1, "To: Gg Data: Hi!\n\n", 18) = 18

openat(AT_FDCWD, "/dev/shm/writer", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0600) = 3

ftruncate(3, 368) = 0

mmap(NULL, 368, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f7cc5e3c000

munmap(0x7f7cc5e3c000, 368) = 0

close(3) = 0

kill(52587, SIGUSR1) = 0

write(1, "Current number of users: 2\n\n", 28) = 28

```
rt_sigaction(SIGUSR1, {sa_handler=0x5628ba3e331b, sa_mask=[USR1],
sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x7f7cc5642520},
{sa_handler=0x5628ba3e331b, sa_mask=[USR1], sa_flags=SA_RESTORER|SA_RESTART,
sa_restorer=0x7f7cc5642520}, 8) = 0

--- SIGUSR1 {si_signo=SIGUSR1, si_code=SI_USER, si_pid=52579, si_uid=1000} ---

rt_sigreturn({mask=[]})                 = -1 EINTR (Interrupted system call)

openat(AT_FDCWD, "/dev/shm/reader", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC,
0600) = 3

ftruncate(3, 368)                       = 0

mmap(NULL, 368, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f7cc5e3c000

munmap(0x7f7cc5e3c000, 368)             = 0

close(3)                                = 0

write(1, "New_msg: type: 1 from: <Arsenii>"..., 38) = 38

write(1, "To: Gg Data: new_chat\n\n", 23) = 23

openat(AT_FDCWD, "/dev/shm/writer", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC,
0600) = 3

ftruncate(3, 368)                       = 0

mmap(NULL, 368, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f7cc5e3c000

munmap(0x7f7cc5e3c000, 368)             = 0

close(3)                                = 0

kill(52579, SIGUSR1)                    = 0

write(1, "Sending msg:from sender: Arsenii"..., 46) = 46

write(1, "Data: Chat has been created succ"..., 42) = 42

write(1, "Current number of users: 2\n\n", 28) = 28

rt_sigaction(SIGUSR1, {sa_handler=0x5628ba3e331b, sa_mask=[USR1],
sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x7f7cc5642520},
{sa_handler=0x5628ba3e331b, sa_mask=[USR1], sa_flags=SA_RESTORER|SA_RESTART,
sa_restorer=0x7f7cc5642520}, 8) = 0

--- SIGUSR1 {si_signo=SIGUSR1, si_code=SI_USER, si_pid=52579, si_uid=1000} ---

rt_sigreturn({mask=[]})                 = -1 EINTR (Interrupted system call)

openat(AT_FDCWD, "/dev/shm/reader", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC,
0600) = 3

ftruncate(3, 368)                       = 0
```

mmap(NULL, 368, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f7cc5e3c000

munmap(0x7f7cc5e3c000, 368) = 0

close(3) = 0

write(1, "New_msg: type: 4 from: <Arsenii>"..., 38) = 38

write(1, "To: Gg Data: new_chat\n\n", 23) = 23

write(1, "Current number of users: 2\n\n", 28) = 28

rt_sigaction(SIGUSR1, {sa_handler=0x5628ba3e331b, sa_mask=[USR1], sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x7f7cc5642520}, {sa_handler=0x5628ba3e331b, sa_mask=[USR1], sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x7f7cc5642520}, 8) = 0

--- SIGUSR1 {si_signo=SIGUSR1, si_code=SI_USER, si_pid=52587, si_uid=1000} ---

rt_sigreturn({mask=[]}) = 0

openat(AT_FDCWD, "/dev/shm/reader", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0600) = 3

ftruncate(3, 368) = 0

mmap(NULL, 368, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f7cc5e3c000

munmap(0x7f7cc5e3c000, 368) = 0

close(3) = 0

write(1, "New_msg: type: 4 from: <Gg>52587"..., 33) = 33

write(1, "To: \r Data: \n\n", 14) = 14

write(1, "Close server. There are no activ"..., 40) = 40

unlink("/dev/shm/myserver_pid") = 0

unlink("/dev/shm/reader") = 0

unlink("/dev/shm/writer") = 0

exit_group(0) = ?

+++ exited with 0 +++

client: strace -oclient_log.log ./client Arsenii

execve("./client", ["./client", "Arsenii"], 0x7fff78f9ffd0 /* 56 vars */) = 0

brk(NULL) = 0x55aaf3aad000

arch_prctl(0x3001 /* ARCH_??? */, 0x7fffce3d36f0) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f57a48d6000

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=81715, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 81715, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f57a48c2000

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"..., 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) = 0x7f57a4600000

mprotect(0x7f57a469a000, 1576960, PROT_NONE) = 0

mmap(0x7f57a469a000, 1118208, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x9a000) = 0x7f57a469a000

mmap(0x7f57a47ab000, 454656, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1ab000) = 0x7f57a47ab000

mmap(0x7f57a481b000, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x21a000) = 0x7f57a481b000

mmap(0x7f57a4829000, 10432, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f57a4829000

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"..., 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) = 0x7f57a48a2000

mmap(0x7f57a48a5000, 94208, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x7f57a48a5000

mmap(0x7f57a48bc000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1a000) = 0x7f57a48bc000

mmap(0x7f57a48c0000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1d000) = 0x7f57a48c0000

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 =\340\2563\265?\356\25x\261\27\313A#\350"..., 68,
896) = 68

newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2216304, ...}, 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, 2260560, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f57a4200000

mmap(0x7f57a4228000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f57a4228000

mmap(0x7f57a43bd000, 360448, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7f57a43bd000

mmap(0x7f57a4415000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x214000) = 0x7f57a4415000

mmap(0x7f57a441b000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f57a441b000

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) =
0x7f57a4519000

mmap(0x7f57a4527000, 507904, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe000) = 0x7f57a4527000

mmap(0x7f57a45a3000, 372736, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x8a000) = 0x7f57a45a3000

mmap(0x7f57a45fe000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe4000) = 0x7f57a45fe000

close(3)                                = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7f57a48a0000
```

```
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f57a489d000

arch_prctl(ARCH_SET_FS, 0x7f57a489d740) = 0

set_tid_address(0x7f57a489da10)         = 52123

set_robust_list(0x7f57a489da20, 24)     = 0

rseq(0x7f57a489e0e0, 0x20, 0, 0x53053053) = 0

mprotect(0x7f57a4415000, 16384, PROT_READ) = 0

mprotect(0x7f57a45fe000, 4096, PROT_READ) = 0

mprotect(0x7f57a48c0000, 4096, PROT_READ) = 0

mprotect(0x7f57a481b000, 45056, PROT_READ) = 0

mprotect(0x55aaf1e31000, 4096, PROT_READ) = 0

mprotect(0x7f57a4910000, 8192, PROT_READ) = 0

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

munmap(0x7f57a48c2000, 81715)           = 0

getrandom("\xae\xb0\x77\x17\xe7\xde\x4b\xbb", 8, GRND_NONBLOCK) = 8

brk(NULL)                               = 0x55aaf3aad000

brk(0x55aaf3ace000)                     = 0x55aaf3ace000

futex(0x7f57a482977c, FUTEX_WAKE_PRIVATE, 2147483647) = 0

openat(AT_FDCWD, "/dev/shm/myserver_pid", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0600) = 3

ftruncate(3, 4)                         = 0

mmap(NULL, 4, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f57a490f000

munmap(0x7f57a490f000, 4)               = 0

close(3)                                = 0

getpid()                                = 52123

newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}, AT_EMPTY_PATH) = 0

write(1, "Conecting...\n", 13)          = 13

rt_sigaction(SIGRT_1, {sa_handler=0x7f57a4291870, sa_mask=[], sa_flags=SA_RESTORER|SA_ONSTACK|SA_RESTART|SA_SIGINFO, sa_restorer=0x7f57a4242520}, 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) = 0x7f57a39ff000

mprotect(0x7f57a3a00000, 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|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f57a41ff910, parent_tid=0x7f57a41ff910, exit_signal=0, stack=0x7f57a39ff000, stack_size=0x7fff00, tls=0x7f57a41ff640} => {parent_tid=[52124]}, 88) = 52124

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

openat(AT_FDCWD, "/dev/shm/reader", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0600) = 3

ftruncate(3, 368)   = 0

mmap(NULL, 368, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f57a490f000

munmap(0x7f57a490f000, 368)   = 0

close(3)   = 0

kill(52060, SIGUSR1)   = 0

clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=2, tv_nsec=0}, {tv_sec=1, tv_nsec=999724726}) = ? ERESTART_RESTARTBLOCK (Interrupted by signal)

--- SIGUSR1 {si_signo=SIGUSR1, si_code=SI_USER, si_pid=52060, si_uid=1000} ---

rt_sigreturn({mask=[]})   = -1 EINTR (Interrupted system call)

write(1, "Server_pid: 52060\n", 18)   = 18

write(1, "Welcome in our chat, Arsenii!\n", 30) = 30

futex(0x7f57a441ba70, FUTEX_WAKE_PRIVATE, 1) = 1

write(1, "Server:OK: Successful created!He"..., 97) = 97

write(1, "--For reading messages write:\n", 30) = 30

write(1, "To:'other_username' 'you message"..., 37) = 37

write(1, "To:chat:'chat_name' 'you message"..., 37) = 37

write(1, "-- Or if you want to join or cre"..., 50) = 50

write(1, "Join:'chat_name'\n", 17)   = 17

write(1, "--Write 'q' to close terminal\n", 30) = 30

write(1, " > ", 3)   = 3

```
newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...},
AT_EMPTY_PATH) = 0

read(0, "\n", 1024)                  = 1

write(1, "\n", 1)                    = 1

write(1, " > ", 3)                   = 3

read(0, "To:Gg Hi!\n", 1024)         = 10

openat(AT_FDCWD, "/dev/shm/reader", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC,
0600) = 3

ftruncate(3, 368)                    = 0

mmap(NULL, 368, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f57a490f000

munmap(0x7f57a490f000, 368)          = 0

close(3)                             = 0

kill(52060, SIGUSR1)                 = 0

clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0}, 0x7fffce3d3320) = 0

write(1, " > ", 3)                   = 3

read(0, 0x55aaf3abf3f0, 1024)        = ? ERESTARTSYS (To be restarted if SA_RESTART is set)

--- SIGUSR1 {si_signo=SIGUSR1, si_code=SI_USER, si_pid=52060, si_uid=1000} ---

rt_sigreturn({mask=[]})              = 0

read(0, "\n", 1024)                  = 1

write(1, "Gg:Hello\n", 9)            = 9

write(1, " > ", 3)                   = 3

read(0, "Join:new_chat\n", 1024)     = 14

openat(AT_FDCWD, "/dev/shm/reader", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC,
0600) = 3

ftruncate(3, 368)                    = 0

mmap(NULL, 368, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f57a490f000

munmap(0x7f57a490f000, 368)          = 0

close(3)                             = 0

kill(52060, SIGUSR1)                 = 0

clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0}, {tv_sec=0,
tv_nsec=999779195}) = ? ERESTART_RESTARTBLOCK (Interrupted by signal)
```

--- SIGUSR1 {si_signo=SIGUSR1, si_code=SI_USER, si_pid=52060, si_uid=1000} ---

rt_sigreturn({mask=[]}) = -1 EINTR (Interrupted system call)

futex(0x7f57a441ba70, FUTEX_WAIT_PRIVATE, 2, NULL) = -1 EAGAIN (Resource temporarily unavailable)

write(1, "Chat:new_chat:Arsenii Chat has b"..., 57) = 57

futex(0x7f57a441ba70, FUTEX_WAKE_PRIVATE, 1) = 0

read(0, "\n", 1024) = 1

write(1, "\n", 1) = 1

write(1, " > ", 3) = 3

read(0, "q\n", 1024) = 2

rt_sigaction(SIGRTMIN, {sa_handler=0x7f57a4292b30, sa_mask=[], sa_flags=SA_RESTORER|SA_RESTART|SA_SIGINFO, sa_restorer=0x7f57a4242520}, NULL, 8) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

getpid() = 52123

tgkill(52123, 52124, SIGRTMIN) = 0

futex(0x7f57a41fffb4, FUTEX_WAKE_PRIVATE, 1) = 1

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

openat(AT_FDCWD, "/dev/shm/reader", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0600) = 3

ftruncate(3, 368) = 0

mmap(NULL, 368, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f57a490f000

munmap(0x7f57a490f000, 368) = 0

close(3) = 0

kill(52060, SIGUSR1) = 0

write(1, "Break;\n", 7) = 7

exit_group(0) = ?

+++ exited with 0 +++

Вывод

Данный курсовой проект направлен на закрепление навыков, полученных за курс Операционных систем. Конкретно мой проект был направлен на закрепление знаний в области разделяемой памяти.

В итоге я написал исправно работающий сервер для обмена сообщений, и поэтому считаю, что с поставленной задачей справился успешно.