Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Курсовой проект по курсу**

**«Операционные системы»**

**Клиент-серверная система для передачи мгновенных сообщений**

Студент:

Группа:

Вариант: 22

Преподаватель:

Дата: 27.12.2023

Оценка:

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2023

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 №22.

22. Необходимо предусмотреть возможность создания «групповых чатов». Связь между сервером и клиентом должна быть реализована при помощи pipe'ов

Листингпрограммы

client.cpp

#include "funcs.h"

#include <thread>

//функция приёма сообщений (для потока)

void func(int fd\_recv, string login){

while (true){

string reply = c\_recieve(fd\_recv);

//cout<< reply << "\n";

cout.flush();

cout<< login << ">";

cout.flush();

}

}

int main(){

//подключение к входному FIFO сервера

int fd\_send = open("input", O\_WRONLY);

if (fd\_send == -1) {

cout<< "ERROR: MAIN FIFO WAS NOT OPENED\n";

exit(1);

}

cout<< "Insert login or chat name: ";

string login;

//подключение к персональному именованному пайпу

int fd\_recv = -1;

while (fd\_recv == -1) {

cin>> login;

fd\_recv = open(login.c\_str(), O\_RDONLY);

if (fd\_recv == -1) {

cout<< "Wrong login!\nInsert your login: ";

}

};

string adressee, message;

cout<< "You have successfully signed!\n\n";

//запуск потока принятия сообщений от сервера

thread thr\_recieve(func, fd\_recv, login);

cout<< "USAGE: <recipient's login or chat name><your message>\n";

cout<< "\tcreate<chat's name><user's quantity><names of users>\n";

cout<< "\tEXAMPLE: create family\_chat 3 mama papa kot\n\n";

cout<< "\tquit - completion of work\n\n";

while (true) {

cout<< login <<"> ";

cin>>adressee;

if (adressee == "quit"){ //quit client

break;

}

if (adressee == "create"){ //create new chat

string cur;

cin>> cur;

message += cur + "$";

int n; cin>> n;

while (n > 0){

cin>> cur;

message += cur + "$";

n--;

}

}

else{

getline(cin, message);

}

c\_send(fd\_send, login, adressee, message);

}

thr\_recieve.detach();

}

server.cpp

#include "funcs.h"

#include <asm-generic/errno-base.h>

#include <atomic>

#include <csignal>

#include <unistd.h>

int in(vector<string> logins, string str) {

for (int i = 0; i < logins.size(); ++i) {

if (logins[i] == str)

return i;

}

return -1;

}

int in\_chat(vector<vector<string>> chats, string str) {

for (int i = 0; i < chats.size(); ++i) {

if (chats[i][0] == str)

return i;

}

return -1;

}

int main() {

vector<string> logins;

vector<vector<string>> chats;

cout << "Enter number of users\n";

int n;

cin >> n;

cout << "\nEnter all user's logins.\n";

while (n > 0) {

string login;

cin >> login;

if (login == "") {

break;

}

if (in(logins, login) == -1) {

logins.push\_back(login);

} else {

cout << "Login already exists!\n\n";

}

n--;

}

cout << "Instant messaging system is started!\n";

// создание и открытие входного FIFO

if (mkfifo("input", 0777) == -1) {

if (errno == EEXIST) {

cout << "MAIN INPUT FIFO WAS ALREADY CREATED" << endl;

} else {

cout << "MAIN INPUT FIFO WAS NOT CREATED";

exit(1);

}

}

int fd\_recv = open("input", O\_RDONLY);

if (fd\_recv == -1) {

cout << "INPUT FIFO WAS NOT OPENED";

exit(1);

}

// создание и открытие выходных FIFO для всех логинов

for (int i = 0; i < logins.size(); ++i) {

if (mkfifo(logins[i].c\_str(), 0777) == -1) {

if (errno == EEXIST) {

cout << "USER FIFO WAS ALREADY CREATED" << endl;

} else {

cout << "FIFO WAS NOT CREATED";

exit(1);

}

}

}

int fd\_l[logins.size()];

for (int i = 0; i < logins.size(); ++i) {

fd\_l[i] = open(logins[i].c\_str(), O\_WRONLY);

if (fd\_l[i] == -1) {

cout << "FIFO login WAS NOT OPENED";

exit(1);

}

}

// обработка сообщений, полученных от клиентов

while (true) {

string message;

message = s\_recieve(fd\_recv); // читаем из input файла

// cout << "message " << message << endl;

string f\_sender = find\_sender(message);

string f\_recipient = find\_recipient(message);

string f\_message\_info = find\_message\_info(message);

int sender = in(logins, f\_sender);

if (f\_recipient == "create") { // processing a request to create a chat

vector<string> v\_cur;

string tmp = find\_text(message);

string s\_cur;

for (int i = 0; i < tmp.size(); i++) {

if (tmp[i] == '$') {

v\_cur.push\_back(s\_cur);

cout << s\_cur;

s\_cur = "";

} else {

s\_cur += tmp[i];

}

}

bool correct\_usernames = true;

for (int i = 1; i < v\_cur.size();

i++) { // check for correct usernames in chat

if (in(logins, v\_cur[i]) == -1) {

correct\_usernames = false;

}

}

if (correct\_usernames) {

chats.push\_back(v\_cur);

s\_send(fd\_l[sender], "Chat created successfully!");

} else {

s\_send(fd\_l[sender], "Login does not exists!");

}

} else { // send messenges

int recipient;

if (in(logins, f\_recipient) != -1) { // to users

recipient = in(logins, f\_recipient);

s\_send(fd\_l[recipient], f\_message\_info);

} else if (in\_chat(chats, f\_recipient) != -1) { // to chats

bool sender\_can\_write = false;

int chat\_num = in\_chat(chats, f\_recipient);

for (int i = 1; i < chats[chat\_num].size();

i++) { // checking whether the sender is in the chat

string rec\_name = chats[chat\_num][i];

if (f\_sender == rec\_name) {

sender\_can\_write = true;

}

}

if (sender\_can\_write) { // success check

for (int i = 1; i < chats[chat\_num].size(); i++) {

string rec\_name = chats[chat\_num][i];

int rec = in(logins, rec\_name);

if (sender != rec) {

s\_send(fd\_l[rec], f\_message\_info);

}

}

} else { // unsuccess

s\_send(fd\_l[sender], "You can't write to this chat!");

}

} else { // wrong login or chat name

s\_send(fd\_l[sender], "Login does not exists!");

}

}

}

};

funcs.h

#include <iostream>

#include <stdlib.h>

#include <unistd.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <fcntl.h>

#include <string>

#include <vector>

using namespace std;

//отправка сообщения от клиента серверу

void c\_send(int fd\_send, string login, string user, string message) {

string text = login + "$" + user + "$" + message;

int k = text.size();

write(fd\_send, &k, sizeof(k));

char c\_message[k];

for (int i = 0; i< k; ++i) {

c\_message[i] = text[i];

}

write(fd\_send, c\_message, k);

}

//получения сообщения клиентом от сервера

string c\_recieve(int fd\_recieve) {

int size;

read(fd\_recieve, &size, sizeof(size));

char s\_message[size];

read(fd\_recieve, s\_message, size);

string recv;

for (int i = 0; i< size; ++i) {

if (s\_message[i] == '$') {

recv = recv + ": ";

}

else {

recv.push\_back(s\_message[i]);

}

}

cout<<recv<<endl;

return recv;

}

//отправка сообщения от сервера клиенту

void s\_send(int fd, string message) {

string text = message;

int k = text.size();

write(fd, &k, sizeof(k));

char s\_message[k];

for (int i = 0; i< k; ++i) {

s\_message[i] = text[i];

}

write(fd, s\_message, k);

}

//получение сообщения сервером от клиента

string s\_recieve(int fd) {

int size;

read(fd, &size, sizeof(size));

char c\_message[size];

read(fd, c\_message, size);

string recv;

for (int i = 0; i< size; ++i) {

recv.push\_back(c\_message[i]);

}

cout<<recv<<endl;

return recv;

}

//-----------Парсинг сообщения-----------

//поиск в сообщении отправителя

string find\_sender(string message){

string login;

int i = 0;

while (message[i] != '$') {

login.push\_back(message[i]);

++i;

}

return login;

}

//поиск в сообщении получателя

string find\_recipient(string message) {

string text;

int i = 0;

while (message[i] != '$') { ++i; } ++i;

while (message[i] != '$') {

text.push\_back(message[i]);

++i;

}

return text;

}

string find\_text(string message) {

string text;

int i = 0;

while (message[i] != '$') { ++i; } ++i;

while (message[i] != '$') { ++i; } ++i;

while (i<message.size()) {

text.push\_back(message[i]);

++i;

}

return text;

}

//поиск в сообщении информации для отправки получателю - текст + отправитель

string find\_message\_info(string message){

string res, sender, mess;

int i = 0;

while (message[i] != '$') {

sender.push\_back(message[i]);

++i;

} ++i;

while (message[i] != '$') { ++i; }

while (i<message.size()) {

mess.push\_back(message[i]);

++i;

}

res = sender + mess;

return res;

}

makefile

all: client server

client:

g++ client.cpp -o client -pthread

server:

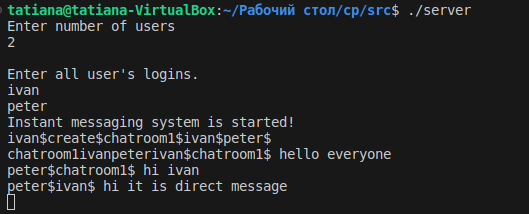
g++ server.cpp -o server

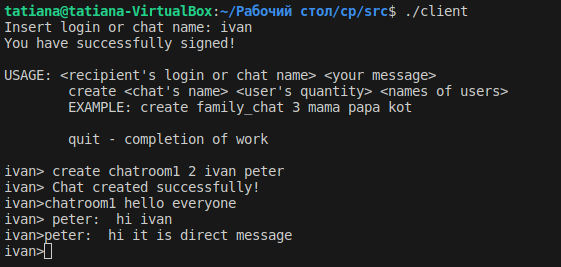
clean:

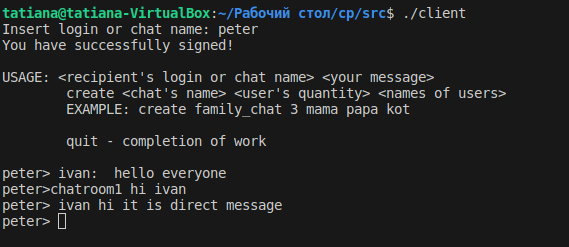
rm -rf client server

Примеры работы программы:

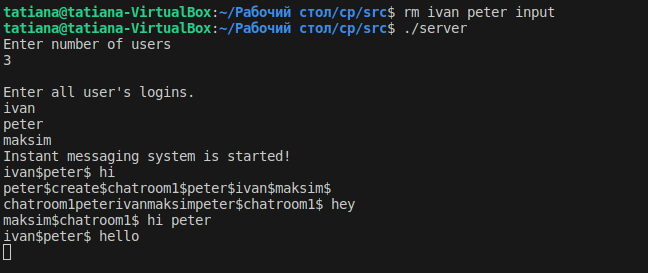
1)

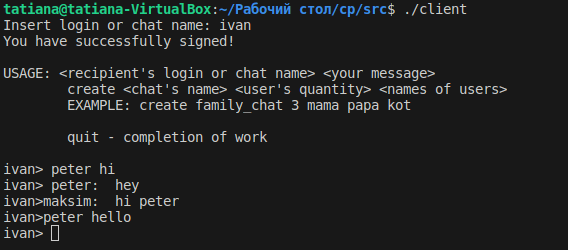


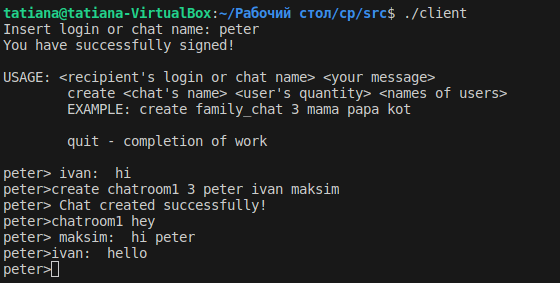


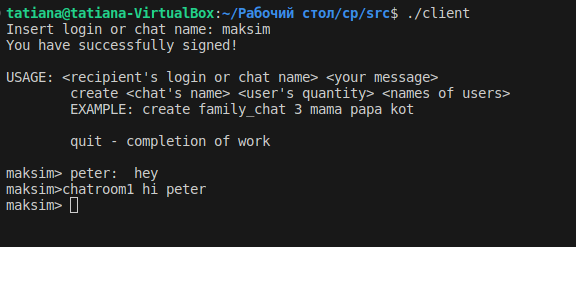


2)









Вывод

В процессе выполнения данного курсового проекта мной была реализована клиент-серверная система для передачи мгновенных сообщений. Написана эта система на пайпах (именованных каналах). Через них довольно удобно устроить общение между двумя процессами, которые не являются “родственниками”, то есть не использовался системный вызов fork. Но эти каналы нужно позже удалять.