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: Попов Илья Павлович

Группа: М80-206Б-20

Вариант: 22

Преподаватель: Соколов Андрей Алексеевич

Дата: 27.12.2021

Оценка:

Подпись: _____

Москва, 2021

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

Листинг программы

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"

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)  {
        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) {
            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

```
lunidep@lunidep-VirtualBox:~/Desktop/kp $ ./server

Enter number of users

4


Enter all user's logins or chat's names.

mama

papa

kot

ilya
```

Instant messaging system is started!

lunidep@lunidep-VirtualBox:~/Desktop/kp$ ./client

Insert login or chat name: **ilya**

You have successfully signed!

USAGE: <recipient's login or chat name> <your message>

create <chat's name> <user's quantity> <names of users>

EXAMPLE: create family_chat 3 mama papa kot

quit - completion of work

ilya> mama: hi from mama

ilya>papa: hi from papa

ilya>create chat 3 ilya mama papa

ilya> Chat created successfully!

ilya>chat hi from ilya, parents

ilya> kot: pochemu bez menya chat sozdali

ilya>chat_s_kotom sorry

lunidep@lunidep-VirtualBox:~/Desktop/kp$ ./client

Insert login or chat name: **mama**

You have successfully signed!

USAGE: <recipient's login or chat name> <your message>

create <chat's name> <user's quantity> <names of users>

EXAMPLE: create family_chat 3 mama papa kot

quit - completion of work

mama> ilya hi from mama

mama> ilya: hi from ilya, parents

mama>kot: pochemu bez menya chat sozdali

mama>ilya: sorry

lunidep@lunidep-VirtualBox:~/Desktop/kp$ ./client

Insert login or chat name: **papa**

You have successfully signed!

USAGE: <recipient's login or chat name> <your message>

create <chat's name> <user's quantity> <names of users>

EXAMPLE: create family_chat 3 mama papa kot

quit - completion of work

papa> ilya hi from papa

papa> ilya: hi from ilya, parents

papa>kot: pochemu bez menya chat sozdali

papa>ilya: sorry

lunidep@lunidep-VirtualBox:~/Desktop/kp$ ./client

Insert login or chat name: **kot**

You have successfully signed!

USAGE: <recipient's login or chat name> <your message>

create <chat's name> <user's quantity> <names of users>

EXAMPLE: create family_chat 3 mama papa kot

quit - completion of work

kot> create chat_s_kotom 4 ilya mama papa kot

kot> Chat created successfully!

kot>chat_s_kotom pochemu bez menya chat sozdali

kot> ilya: sorry

Вывод

В процессе выполнения данного курсового проекта мною была реализована клиент-серверная система для передачи мгновенных сообщений. Написана эта система на пайпах (именованных каналах). Они являются удобным инструментом, потому что, сохраняя логику пайпов, являются по сути ммап файлами. 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.

Из минусов можно выделить то, что впоследствии именованные каналы необходимо удалять.