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);
        if (login.find("chat") == -1){
            //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";

    //запуск потока принятия сообщений от сервера
    thread thr_recieve(func, fd_recv, login);

    //запуск цикла отправки сообщений на сервер
    if (login.find("chat") == -1){//user
        cout << "USAGE: <recipient's login> <your message>\n\tquit - completion of work\n";

        while (true) {
            cout << login <<"> ";
            cin >> adressee;

            if (adressee == "quit")
                break;
            getline(cin, message);
            c_send(fd_send, login, adressee, message);
        }
    }
    else{//chat
        cout << "\tquit - completion of work\n\n";

        while (true) {
            cin >> adressee;
            if (adressee == "quit")
                break;
        }
    }

    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 main(){
```

```
vector<string> logins;
vector<string> chats;

cout << "Enter number of users\n";
int n; cin >> n;
cout << "\nEnter all user's logins or chat's names.\n";

while (n>0) {
   string login;
   cin >> login;
   if (login == "") { break; }
   if (login.find("chat") == 0){
      if (in(chats, login) == -1){
         chats.push_back(login);
      }
      else{
         cout << "Chat already exists!\n\n";
      }
   }
   else{
      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);
   }
}
for (int i = 0; i < chats.size(); ++i)    {
   if (mkfifo(chats[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);
    }
}

int fd_ch[chats.size()];
for (int i = 0; i < chats.size(); ++i) {
    fd_ch[i] = open(chats[i].c_str(), O_WRONLY);
    if (fd_ch[i] == -1) {
        cout << "FIFO chat 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 fd_sender = in(logins, f_sender);

    int fd_recipient;
    if (in(logins, f_recipient) != -1){
        fd_recipient = in(logins, f_recipient);
        s_send(fd_l[fd_recipient], f_message_info);
    }
    else if (in(chats, f_recipient) != -1){
        fd_recipient = in(chats, f_recipient);
        s_send(fd_ch[fd_recipient], f_message_info);
    }
    else{
        s_send(fd_l[fd_sender], "Login does not exists!");
    }

    // cout << "-----------------------------------------\n";
    // cout << "f_sender  " << f_sender << endl;
    // cout << "f_recipient  " << f_recipient << endl;
    // cout << "f_message_info  " << f_message_info << endl << endl;
    // cout << "fd_recipient  " << fd_recipient << endl;
```

```
        // cout << "fd_sender  " << fd_sender << endl;
        // cout << "-------------------------------------------\n\n";
    }
}
```

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;
}

//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/OS/kp$ ./client

Insert login or chat name: **mama**

You have successfully signed!

USAGE: <recipient's login> <your message>

quit - completion of work

mama> papa posmotri na koshku

mama> papa: kakaya krasivaya koshka

mama> chat_family kto poidet za hlebom

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

Insert login or chat name: **papa**

You have successfully signed!

USAGE: <recipient's login> <your message>

quit - completion of work

papa> mama: posmotri na koshku

papa> mama kakaya krasivaya koshka

papa> chat_family nu davai ya

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

Insert login or chat name: **koshka**

You have successfully signed!

USAGE: <recipient's login> <your message>

quit - completion of work

koshka> chat_family vsem spasibo za komplimenti

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

Insert login or chat name: **chat_family**

You have successfully signed!

quit - completion of work

koshka: vsem spasibo za komplimenti

mama: kto poidet za hlebom

papa: nu davai ya

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.