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.