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: Лисин Р.С.

Группа: М80-206Б-20

Преподаватель: Соколов Андрей Алексеевич

Дата: 25.12.2021

Оценка: 5

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

1. **Постановка задачи**

Написать клиент-серверную систему для передачи мгновенных сообщений. Базовый функционал должен быть следующим:

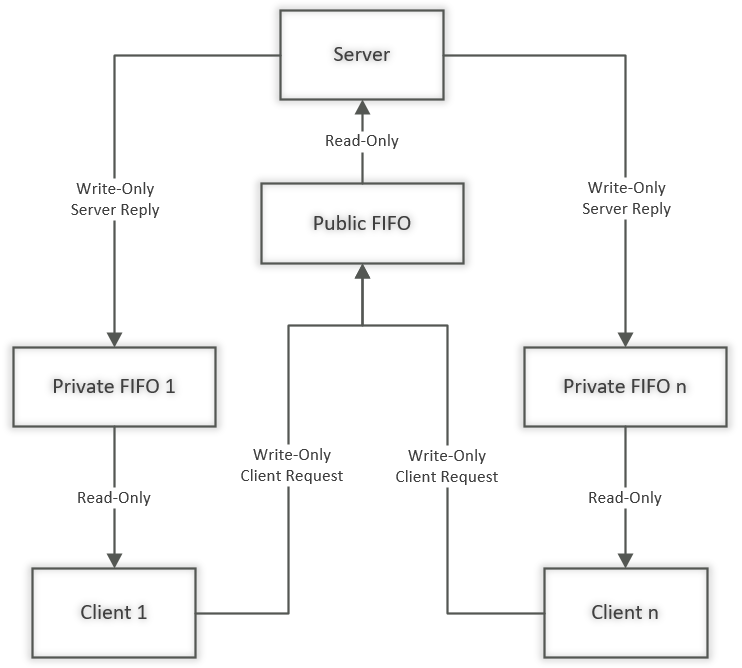
• Клиент может присоединиться к серверу, введя логин

• Клиент может отправить сообщение другому клиенту по его логину

• Клиент в реальном времени принимает сообщения от других клиентов (то есть в отдельном потоке)

Вариант 25: Необходимо предусмотреть возможность хранения истории переписок (на сервере) и поиска по ним. Связь между сервером и клиентом должна быть реализована при помощи pipe'ов (именнованных) - FIFO.

1. **Схема работы программы**



1. **Листинг программы**

***funcs.hpp***

#include <string>

#include <vector>

//отправить сообщение серверу в удобной форме - логин$получатель$сообщение

void send\_message\_to\_server(int fd, std::string curlogin, std::string user, std::string message)

{

std::string text = curlogin + "$" + user + "$" + message;

int k = text.size();

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

char messagec[k];

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

{

messagec[i] = text[i];

}

write(fd, messagec, k);

}

//отправить сообщение клиенту

void send\_message\_to\_client(int fd, std::string message)

{

std::string text = message;

int k = text.size();

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

char messagec[k];

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

{

messagec[i] = text[i];

}

write(fd, messagec, k);

}

//получить сообщение в удобной для клиента форме

std::string recieve\_message\_client(int fd)

{

int size;

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

char messagec[size];

read(fd, messagec, size);

std::string recv;

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

{

if (messagec[i] != '$')

{

recv.push\_back(messagec[i]);

}

else

{

recv = recv + ": ";

}

}

return recv;

}

//получить сообщение в удобной для сервера форме

std::string recieve\_message\_server(int fd)

{

int size;

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

char messagec[size];

read(fd, messagec, size);

std::string recv;

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

{

recv.push\_back(messagec[i]);

}

return recv;

}

//получить логин из сообщения для сервера

std::string extract\_login(std::string message)

{

std::string login;

int i = 0;

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

{

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

++i;

}

return login;

}

//получить сообщение для клиента

std::string extract\_message(std::string message)

{

std::string text, text1, text2;

int i = 0;

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

{

text1.push\_back(message[i]);

++i;

}

++i;

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

{

++i;

}

while (i < message.size())

{

text2.push\_back(message[i]);

++i;

//std::cout << "TESTSSSS";

}

text = text1 + text2;

return text;

}

//получить получателя сообщения

std::string extract\_addressee(std::string message)

{

std::string text;

int i = 0;

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

{

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

++i;

}

++i;

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

{

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

++i;

//std::cout << "TESTSSSS";

}

return text;

}

//получить текст сообщения

std::string extract\_text(std::string message)

{

std::string text;

int i = 0;

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

{

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

++i;

}

++i;

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

{

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

++i;

}

++i;

++i;

while (i < message.size())

{

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

++i;

//std::cout << "TESTSSSS";

}

return text;

}

//поиск подстроки в строке алгоритмом Кнута-Морриса-Пратта

std::vector<int> prefix\_function(std::string s) {

int n = s.size();

std::vector<int> pi(n);

for (int i = 1; i < n; i++) {

int j = pi[i - 1];

while (j > 0 && s[i] != s[j]) {

j = pi[j - 1];

}

if (s[i] == s[j]) {

pi[i] = j + 1;

}

}

return pi;

}

bool search(std::string t, std::string p) {

std::vector<int> v = prefix\_function(p);

int l = 0;

int k = 0;

while (k < t.size()) {

if (t[k] == p[l]) {

k++;

l++;

if (l == p.size()) {

return true;

}

}

else if (l == 0) {

k++;

}

else {

l = v[l - 1];

}

}

return false;

}

***client.cpp***

#include <iostream>

#include <stdlib.h>

#include <unistd.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <errno.h>

#include <vector>

#include <fcntl.h>

#include "funcs.hpp"

#include <thread>

//функция приёма сообщений (для потока)

void func(int fd\_recv, std::string login)

{

while (1)

{

std::string reply = recieve\_message\_client(fd\_recv);

std::cout << reply << "\n";

std::cout.flush();

std::cout << login << ">";

std::cout.flush();

}

}

int main()

{

//подключение к входному FIFO сервера

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

if (fd\_send == -1)

{

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

exit(1);

}

//подготовка - инструкции, ввод логина

std::cout << "Wellcome to VMAI.\nTo create accounts launch ./server and insert logins.\n Than relaunch this application and enter your login.\n";

std::cout << "Input: login message. Example: anton\n hey, how are you?\n";

std::cout << "Insert your login: ";

std::string login;

//подключение к персональному именованному пайпу

int fd\_recv = -1;

while (fd\_recv == -1)

{

std::cin >> login;

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

if (fd\_recv == -1)

{

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

}

};

//вход успешен, запуск потока принятия сообщений от сервера

std::string adressee, message;

std::cout << "Congrats! You have signed in VMAI. Now you can send messages!\n";

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

//запуск цикла отправки сообщений на сервер

while (1)

{

std::cout << login << "> ";

std::cin >> adressee;

if (adressee == "history")

{

std::string pattern;

std::getline(std::cin, pattern);

send\_message\_to\_server(fd\_send, login, adressee, pattern);

}

else

{

if (adressee == "quit")

break;

std::getline(std::cin, message);

send\_message\_to\_server(fd\_send, login, adressee, message);

}

}

//return 0;

thr\_recieve.detach();

}

***server.cpp***

#include <iostream>

#include <stdlib.h>

#include <unistd.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <errno.h>

#include <vector>

#include <fcntl.h>

#include "funcs.hpp"

#include <map>

#include <vector>

int in(std::vector<std::string> logins, std::string str)

{

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

{

if (logins[i] == str)

return i;

}

return -1;

}

int main()

{

std::vector<std::vector<std::string>> history; // здесь хранится история переписок

// соответсвие с вектором logins по принципу ассоциативных векторов

std::vector<std::string> logins;

std::string command;

std::string login;

//ввод логинов

std::cout << "Enter all user's logins. Insert 'end' to stop:\n";

while (login != "end")

{

std::cin >> login;

std::vector<std::string> vec;

vec.push\_back(login);

history.push\_back(vec);

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

logins.push\_back(login);

else

std::cout << "already exists!";

}

std::cout << "TEST3\n";

//создание выходных FIFO для всех логинов

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

{

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

{

if (errno != EEXIST)

{

std::cout << "FIFO WAS NOT CREATED";

exit(1);

}

}

}

//создание входного FIFO

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

{

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

exit(1);

}

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

if (fd\_recv == -1)

{

std::cout << "INPUT FIFO WAS NOT OPENED";

exit(1);

}

//открытие всех FIFO на запись

int fd[logins.size()];

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

{

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

}

while (3)

{

std::string message;

message = recieve\_message\_server(fd\_recv);

std::cout << message;

std::string rcvd\_usr = extract\_login(message); //от кого

std::string rcvd\_adressee = extract\_addressee(message); //кому

std::string rcvd\_message = extract\_message(message); //что

int fd\_repl = in(logins, rcvd\_adressee); //id получателя

int fd\_usr = in(logins, rcvd\_usr); //id отправителя

//std::cout << rcvd\_adressee;

int pos = -1;

if (rcvd\_adressee == "history")

{

std::string reply = "No matches found\n";

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

{

if (i == fd\_usr)

{

for (int j = 0; j < history[i].size(); ++j)

{

if (search(history[i][j], extract\_text(message)))

{

reply = history[i][j];

pos = i;

}

}

}

}

if (reply != "No matches found\n")

{

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

{

if (i != pos)

{

for (int j = 0; j < history[i].size(); ++j)

{

if (search(history[i][j], extract\_text(message)))

{

reply = "[To " + logins[i] + "] " + history[i][j];

}

}

}

}

}

send\_message\_to\_client(fd[fd\_usr], reply);

}

else

{

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

{

if (logins[i] == rcvd\_usr)

history[i].push\_back(extract\_text(message));

if (logins[i] == rcvd\_adressee && rcvd\_usr != rcvd\_adressee)

history[i].push\_back(extract\_text(message));

}

// for (int i = 0; i < history.size(); ++i)

// {

// for (int j = 0; j < history[i].size(); ++j)

// {

// std::cout << history[i][j] << " ";

// }

// std::cout << "\n";

// }

if (in(logins, rcvd\_adressee) == -1)

{

send\_message\_to\_client(fd[fd\_usr], "Login does not exists!\n");

}

else

{

send\_message\_to\_client(fd[fd\_repl], rcvd\_message);

}

}

}

}

***Makefile***

all: client server

client:

g++ client.cpp -o client -pthread

server:

g++ server.cpp -o server

clean:

rm -rf client server

**4. Пример работы программы**

roma@DESKTOP-JD58QU2:~/os\_labs/final\_kp$ ./server

Enter all user's logins. Insert 'end' to stop:

vasya

petya

andrey

master

end

TEST3

roma@DESKTOP-JD58QU2:~/os\_labs/final\_kp$ ./client

Wellcome to VMAI.

To create accounts launch ./server and insert logins.

Than relaunch this application and enter your login.

Input: login message. Example: anton

hey, how are you?

Insert your login: sergey

Wrong login!

Insert your login: vasya

Congrats! You have signed in VMAI. Now you can send messages!

vasya> petya hello from vasya!

vasya> petya: I recieved your message, vasya!

vasya>goblin hi

vasya> Login does not exists!

vasya>quit

roma@DESKTOP-JD58QU2:~/os\_labs/final\_kp$ ./client

Wellcome to VMAI.

To create accounts launch ./server and insert logins.

Than relaunch this application and enter your login.

Input: login message. Example: anton

hey, how are you?

Insert your login: petya

Congrats! You have signed in VMAI. Now you can send messages!

petya> vasya: hello from vasya!

petya>vasya I recieved your message, vasya!

petya> andrey: Have you forgotten about me?

petya>andrey what was your message about?

petya> history message

petya> [To andrey] what was your message about?

petya>

roma@DESKTOP-JD58QU2:~/os\_labs/final\_kp$ ./client

Wellcome to VMAI.

To create accounts launch ./server and insert logins.

Than relaunch this application and enter your login.

Input: login message. Example: anton

hey, how are you?

Insert your login: andrey

Congrats! You have signed in VMAI. Now you can send messages!

andrey> petya Have you forgotten about me?

andrey> wefas

andrey> Login does not exists!

andrey>quit

roma@DESKTOP-JD58QU2:~/os\_labs/final\_kp$ ./client

Wellcome to VMAI.

To create accounts launch ./server and insert logins.

Than relaunch this application and enter your login.

Input: login message. Example: anton

hey, how are you?

Insert your login: andrey

Congrats! You have signed in VMAI. Now you can send messages!

andrey> history me

andrey> [To petya] Have you forgotten about me?

andrey>petya: what was your message about?

andrey>

**5. Выводы**

В данной работе я написал базовый чат на пайпах. Именованные каналы являются хорошим инструментом для обмена данными между двумя процессами, которые не являются “родственниками”, то есть не использовался системный вызов fork. Но эти каналы нужно позже удалять. Также они занимают память, не сравнимую с сокетами. Благодаря выбранной схеме построения, взаимодействие между сервером и клиентами получилось реализовать на высоком уровне.