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: Почечура Артемий Андреевич

Группа: М80-206Б-20

Преподаватель: Соколов Андрей Алексеевич

Дата: 28.12.2021

Оценка: 5

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

1. **Постановка задачи**

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

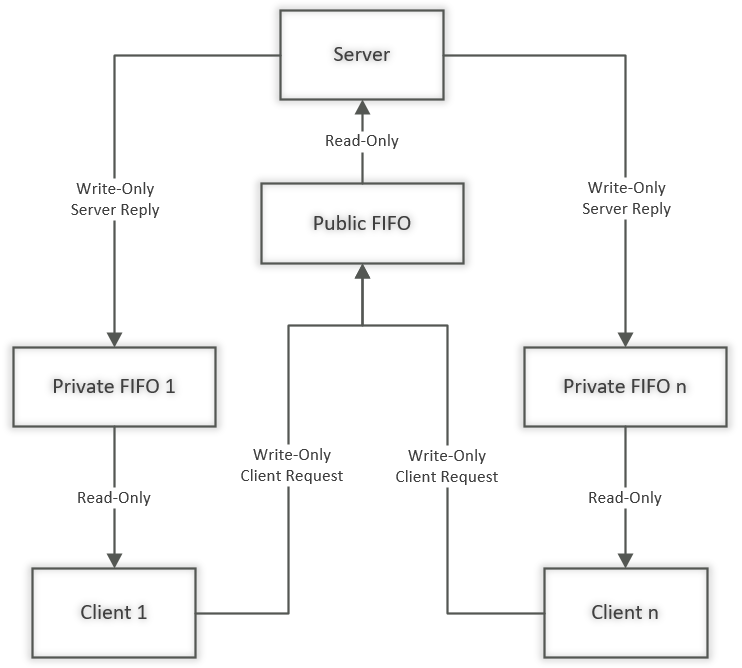
• Клиент может присоединиться к серверу, введя логин

• Клиент может отправить сообщение другому клиенту по его логину

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

Вариант 28: Необходимо предусмотреть возможность отправки отложенного сообщения или себе, или другому пользователю. При выходе с сервера отправка всё равно должна осуществиться. Связь между сервером и клиентом должна быть реализована при помощи pipe'ов.

1. **Схема работы программы**



1. **Листинг программы**

***funcs.hpp***

#include <string>

#include <vector>

#include <iostream>

using namespace std;

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

void send\_message\_to\_server(int fd, string curlogin, string user, int time, string message){

string s\_time;

char k1;

int q;

while(time>0){

q=time%10+'0';

k1=q;

s\_time=k1 + s\_time;

time=time/10;

}

if(s\_time.size()==0){

s\_time="0";

}

string text = curlogin + "$" + user + "$" + s\_time + "$" + 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, string message){

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

string recieve\_message\_client(int fd){

int size;

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

char messagec[size];

read(fd, messagec, size);

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

string recieve\_message\_server(int fd){

int size;

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

char messagec[size];

read(fd, messagec, size);

string recv;

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

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

}

return recv;

}

//получить логин из сообщения для сервера

string extract\_login(string message){

string login;

int i = 0;

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

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

i++;

}

return login;

}

//получить сообщение для клиента

string extract\_message(string message){

string text, text1, text2;

int i = 0;

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

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

i++;

}

i++;

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

i++;

}

i++;

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

i++;

}

while (i < message.size()){

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

i++;

}

text = text1 + text2;

return text;

}

//получить время отправки сообщения

string extract\_time(string message){

string text;

int i = 0;

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

i++;

}

i++;

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

i++;

}

i++;

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

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

i++;

}

return text;

}

//получить получателя сообщения

string extract\_addressee(string message){

string text;

int i = 0;

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

i++;

}

i++;

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

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

i++;

}

return text;

}

//получить текст сообщения

string extract\_text(string message){

string text;

int i = 0;

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

i++;

}

i++;

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

i++;

}

i++;

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

i++;

}

i++;

while (i < message.size()){

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

i++;

}

return text;

}

***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.h"

#include <thread>

using namespace std;

//функция приёма сообщений (для потока)

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

while (true){

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

cout << reply << "\n";

cout.flush();

cout << login << ">";

cout.flush();

}

}

int main(){

//подключение к входному FIFO сервера

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

if (fd\_send == -1){

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

exit(1);

}

//подготовка - инструкции, ввод логина

cout << "Welcome to Sdacha\_KP.\nTo create accounts launch ./server and insert logins.\n Than relaunch this application and enter your login.\n";

cout << "Input: LOGIN TIME MESSAGE. Example: anton 5 hey, how are you?\n";

cout << "Insert your login: ";

string login;

//подключение к персональному именованному пайпу

int fd\_recv = -1;

while (fd\_recv == -1){

cin >> login;

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

if (fd\_recv == -1){

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

}

};

//вход успешен, запуск потока принятия сообщений от сервера

string adressee, message;

cout << "Congrats! You have signed in Sdacha\_KP. Now you can send messages!\n";

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

//запуск цикла отправки сообщений на сервер

while (true){

cout << login << "> ";

cin >> adressee;

if (adressee == "quit")

break;

int time;

cin >> time;

while(time<0){

cout << "Time must be positive or 0\n";

cin >> time;

}

getline(cin, message);

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

}

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 <string>

#include <fcntl.h>

#include "funcs.h"

#include <map>

#include <vector>

#include <math.h>

#include <algorithm>

#include <thread>

using namespace std;

typedef struct{

string m;

int fd;

int time;

} vec\_el;

vector<vec\_el> vector\_mes;

void func(vector<int> fd){

while(true){

if(vector\_mes.size()!=0){

sort(vector\_mes.begin(),vector\_mes.end(),

[](const vec\_el a, const vec\_el b){

return a.time<b.time;

});

for(int i=0;i<vector\_mes.size();i++){

if(vector\_mes[i].time==0){

send\_message\_to\_client(fd[vector\_mes[i].fd], vector\_mes[i].m);

}

vector\_mes[i].time--;

}

for(int i=0;i<vector\_mes.size();i++){

if(vector\_mes[i].time==-1){

vector\_mes.erase(vector\_mes.begin());

}

}

}

if(vector\_mes.size()!=0){

sleep(1);

}

}

}

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;

string command;

string login;

//ввод логинов

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

while (true){

cin >> login;

if(login == "end"){

break;

}

vector<string> vec;

vec.push\_back(login);

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

logins.push\_back(login);

else

cout << "already exists!";

}

//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){

cout << "FIFO WAS NOT CREATED";

exit(1);

}

}

}

//создание входного FIFO

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

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

exit(1);

}

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

if (fd\_pub == -1){

cout << "INPUT FIFO WAS NOT OPENED";

exit(1);

}

//открытие всех FIFO на запись

int fd[logins.size()];

vector<int> fd(logins.size());

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

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

}

thread thr\_recieve(func, fd);

while (true){

string message;

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

cout << message << "\n";

string s\_usr = extract\_login(message); //от кого

string s\_adressee = extract\_addressee(message); //кому

string s\_time = extract\_time(message); //время

string s\_message = extract\_message(message); //что

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

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

int pos = -1;

int time = stoi(s\_time.data());

vec\_el q\_mes;

q\_mes.m=s\_message;

q\_mes.time=time;

q\_mes.fd=fd\_repl;

if (in(logins, s\_adressee) == -1){

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

} else{

vector\_mes.push\_back(q\_mes);

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

}

}

}

***Makefile***

all: client server

client: client.cpp

g++ client.cpp -o client -pthread

server: server.cpp

g++ server.cpp -o server –pthread

**4. Пример работы программы**

root@DESKTOP-5HM2HTK:~#./server  
Enter all user's logins. Insert 'end' to stop:  
artyom  
misha  
artemii  
end  
misha$artyom$10$ kopusta  
misha$artemii$0$ tikva  
artemii$misha$15$ ogurets varenii  
artyom$misha$17$ krijovnik  
misha$artyom$10$ arbuz  
misha$artemii$5$ arbuz

root@DESKTOP-5HM2HTK:~# ./client  
Wellcome to Sdacha\_KP.  
To create accounts launch ./server and insert logins.  
Than relaunch this application and enter your login.  
Input: LOGIN TIME MESSAGE. Example: anton 5 hey, how are you?  
Insert your login: misha  
Congrats! You have signed in Sdacha\_KP. Now you can send messages!  
misha> artyom 10 kopusta  
misha> artemii 0 tikva  
misha> artemii: ogurets varenii  
misha>artyom: krijovnik  
misha>artyom 10 arbuz  
misha> artemii 5 arbuz

misha> quit

root@DESKTOP-5HM2HTK:~# ./client  
Wellcome to Sdacha\_KP.  
To create accounts launch ./server and insert logins.  
Than relaunch this application and enter your login.  
Input: LOGIN TIME MESSAGE. Example: anton 5 hey, how are you?  
Insert your login: artemii  
Congrats! You have signed in Sdacha\_KP. Now you can send messages!  
artemii> misha: tikva  
artemii>misha 15 ogurets varenii  
artemii> misha: arbuz

artemii> quit

root@DESKTOP-5HM2HTK:~# ./client  
Wellcome to Sdacha\_KP.  
To create accounts launch ./server and insert logins.  
Than relaunch this application and enter your login.  
Input: LOGIN TIME MESSAGE. Example: anton 5 hey, how are you?  
Insert your login: artyom  
Congrats! You have signed in Sdacha\_KP. Now you can send messages!  
artyom> misha: kopusta  
artyom>misha 17 krijovnik  
artyom> misha: arbuz

artyom> quit

**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.