Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Курсовой проект по курсу**

**«Операционные системы»**

Студент: Кажекин Д.А.

Группа: М8О-207Б-21

Вариант: 25

Преподаватель: Черемисинов Максим

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2022

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

**Репозиторий**

https://github.com/DKazhekin/OS

**Постановка задачи**

**Цель работы**

Приобретение практических навыков в использовании знаний, полученных в течении курса.

**Задание**

Необходимо создать 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

В соответствие с вариантом необходимо предусмотреть возможность хранения истории переписок (на сервере) и поиска по ним. Связь между сервером и клиентом должна быть реализована при помощи pipe'ов (FIFO).

**Общие сведения о программе**

Программа компилируется отдельно: сначала файл server.cpp, а затем client.cpp (Подробнее в листинге программы)

**Общий метод и алгоритм решения**

Логика программы заключается в том, что изначально на сервере нам необходимо зарегистрировать всех пользователей, участвующих в чате, предварительно создав для каждого отдельный приватный FIFO, а также один общий распределительный FIFO (input), в который со стороны клиентов будет записываться сообщения от пользователей, а сервер будет распределять в какой частный FIFO их отправить для получения адресатом. Также реализована возможность просмотра истории сообщений каждого пользователя, для этого организован вектор “history”.

**Исходный код**

**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 "functions.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 << "ALL RIGHT! CHAT IS READY. LAUNCH CLIENTS\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 (true)

{

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

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));

}

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);

}

}

}

}

**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 "functions.hpp"

#include <thread>

//приём сообщений

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 << "Welcome to chat.\n";

std::cout << "Follow the instruction below and notice: to switch the user you need to leave this session and re-login via './client'\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 chat.\n";

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

//запуск цикла отправки сообщений на сервер

while (true)

{

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();

}

**Functions.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] != '$')

{

++i;

}

++i;

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

{

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

++i;

}

return text;

}

//получить текст сообщения

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

{

std::string text;

int i = 0;

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

{

++i;

}

++i;

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

{

++i;

}

++i;

++i;

while (i < message.size())

{

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

++i;

}

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;

}

**Makefile**

all: client server

client:

g++ client.cpp -pthread -o client -std=c++11

-server:

g++ server.cpp -o server -std=c++11

**Демонстрация работы программы**



**Выводы**

В ходе выполнения курсового проекта у меня получилось подытожить знания, полученные в ходе курса «Операционные системы» и написать собственный мессенджер с историей сообщений, используя технологию FIFO.