Министерство образования и науки Российской Федерации

Югорский государственный университет

Институт цифровой экономики

Отчет о лабораторной работе №5

По дисциплине «Разработка системных приложений»

Выполнил: студент группы 1161б

Стародубов Юрий

Проверил: Шицелов А.В.

Ханты-Мансийск

2019

Цель: Изучить основы работы с сетевым стеком на примере сокетов.

Задание

Написать HTTP сервер, который на любой запрос отдает страницу index.html, которая лежит рабом с исполняемым файлом сервера.

Сервер должен уметь:

* Исполнять каждый запрос в отдельном потоке
* Поддерживать одновременный доступ нескольких клиентов
* Реализовывать стандарт общения между клиентом (браузером) и сервером как минимум по стандарту HTTP 1.0
* Не требовать перезагрузки для принятия изменений в файле index.html (не кэшировать index.html)

Index.html должен содержать:

* Правильную базовую разметку HTML страницы
* Название страницы (окна, вкладки браузера) «Разработка системных приложений»
* В body должно быть написано:

Привет мир!

Я <ваше ФИО>

Ход работы:

Листинг 1. Код сервера.

#include <netinet/in.h> //socket\_in

#include <sys/socket.h>

#include <sys/types.h>

// #include <errno.h>

#include <arpa/inet.h> //inet\_pton

#include <chrono>

#include <ctime>

#include <fstream>

#include <iostream>

#include <signal.h>

#include <stdio.h>

#include <string.h>

#include <unistd.h> //close

using namespace std;

int mSocketFd;

int cSocketFd;

void ER(int s) {

close(cSocketFd);

close(mSocketFd);

cout << "Program was closed" << endl;

exit(1);

}

string answerMessage() {

string resultRead = "";

string answer;

string temp;

ifstream file;

file.open("/home/piratemaster/CLionProjects/untitled1/index.html");

if (!file.is\_open()) {

cerr << "File not open" << endl;

answer = "HTTP/1.0 404 OK\nConnection: close\nServer: MyServer\nContent-Length: 11\nContent-Type: text/html\n\n Not Found";

}else{

while (!file.eof()) {

getline(file, temp);

resultRead += temp;

answer = "HTTP/1.0 200 OK\nConnection: close\nServer: MyServer\nContent-Type: text/html\nContent-Length: ";

answer += to\_string(resultRead.length());

answer += "\n\n";

answer += resultRead;

}

}

file.close();

return answer;

}

int main(int argc, char \*argv[]) {

signal(SIGINT, ER);

signal(SIGTSTP, ER);

mSocketFd = socket(AF\_INET, SOCK\_STREAM, 0); //домен, тип, протокол

if (mSocketFd == -1) {

cerr << "socket open error" << endl;

}

const char address[] = "127.0.0.1";

int port = 8084;

int clientSize = 5;

int messageSize = 1024;

/\*именование сокета\*/

sockaddr\_in addr\_in; // or sockaddr and more paint

addr\_in.sin\_family = AF\_INET;

addr\_in.sin\_port = htons(port);

// store this IP address in sa:

inet\_pton(AF\_INET, address, &(addr\_in.sin\_addr));

int socketBind = bind(mSocketFd, (sockaddr \*) &addr\_in, sizeof(addr\_in));

if (socketBind == -1) {

cerr << "bind error" << endl;

ER(0);

}

/\*установка соединения\*/

/\*очередь запросов на оединение\*/

if (listen(mSocketFd, clientSize) == -1) {

cerr << "listen error" << endl;

}

//создание нового отдельного сокета для клиента

sockaddr clientAdds;

socklen\_t clientAddsLen;

// Client Socket File Descriptor

while (true) {

cSocketFd = accept(mSocketFd, &clientAdds, &clientAddsLen);

pid\_t nameFork = fork();

if (nameFork == 0) {

//cout << "It is Fork. My pid is " << getpid() << endl;

// cSocketFd = accept(mSocketFd, NULL, NULL);

if (cSocketFd == -1) {

cerr << "accept error" << endl;

}

/\*принять данные\*/

char message[messageSize];

if (recv(cSocketFd, &message, 2000, 0) == -1) {

cerr << "recv error" << endl;

} else {

printf("%s", message);

}

/\*отправить данные \*/

// char message[] = "one";

char st[messageSize];

strcpy(st, answerMessage().c\_str());

if (send(cSocketFd, st, sizeof(st), 0) == -1) {

cerr << "Error send messge " << endl;

} else {

cout << "message send" << endl;

}

close(cSocketFd);

close(mSocketFd);

return 0;

}

}

}

Вывод: в ходе данной лабораторной работы были изучены основы работы с сетевым стеком на примере сокетов