МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА № 43

ОТЧЁТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Старший преподаватель | |  |  | | | |  | | М. Д. Поляк |
| должность, уч. степень, звание | |  | подпись, дата | | | |  | | инициалы, фамилия |
| ОТЧЁТ О ЛАБОРАТОРНОЙ РАБОТЕ №8 | | | | | | | | | | |
| Системы виртуализации и облачные технологии на примере платформы AWS или Yandex Cloud | | | | | | | | | | |
| по дисциплине: Операционные системы | | | | | | | | | | |
|  | | | | | | | | | | |
| РАБОТУ ВЫПОЛНИЛА | | | | | | | | | | |
| СТУДЕНТКА ГР. | 4931 | | |  | 20.05.2022 |  | | Е.Ю. Ильченко | | |
|  |  | | |  | подпись, дата |  | | инициалы, фамилия | | |
|  |  | | |  |  |  | |  | | |

Санкт-Петербург 2022

## Вариант 11

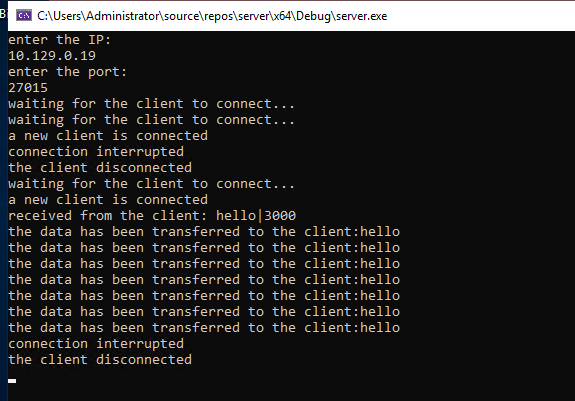
## Цель:

Изучение принципов работы с системами виртуализации в облаке.

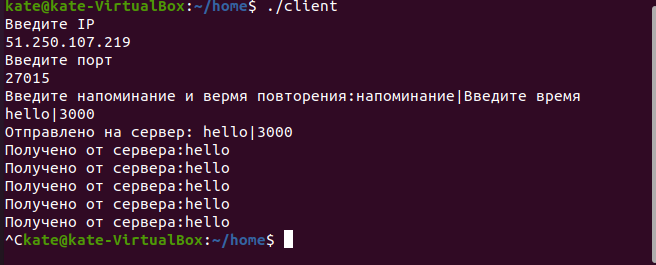
**Задание на лабораторную работу**:   
1. Зарегистрировать аккаунт AWS или Yandex Cloud. При создании платежного аккаунта в  
 Yandex Cloud включить пробный период.  
2. В соответствии с вариантом задания на лабораторную работу No6 определить  
 операционную систему для создаваемой виртуальной машины. Если в лабораторной  
 работе No6 серверная часть приложения разрабатывалась под ОС Linux, то для  
 виртуальной машины также следует выбрать ОС Linux. Если же серверная часть  
 приложения разрабатывалась под ОС Windows, то для виртуальной машины следует  
 выбрать ОС Windows.  
3. Создать виртуальную машину:  
 a. AWS. В рамках уровня бесплатного использования AWS создать виртуальную  
 машину на Amazon EC2 с использованием инстанса t2.micro под управлением ОС  
 Linux или ОС Windows (в зависимости от варианта). Использовать любой из  
 дистрибутивов ОС Linux или ОС Windows, предлагаемых Amazon в рамках  
 программы бесплатного использования AWS.  
 b. Yandex Cloud. В рамках пробного периода и предоставленного гранта Yandex Cloud  
 создать виртуальную машину Yandex Compute Cloud под управлением ОС Linux или  
 ОС Windows (в зависимости от варианта). Использовать любой из дистрибутивов  
 ОС Linux или ОС Windows, предлагаемых Yandex Cloud в рамках пробного периода.  
4. Разместить на виртуальной машине в облаке серверную часть приложения из  
 лабораторной работы No6.  
5. Настроить брандмауэр и открыть порт, на котором работает сервер.  
6. Продемонстрировать корректную работу по сети клиентской части приложения с  
 запущенной ранее в облаке серверной частью. Это значит, что сервер должен быть  
 запущен в облаке на виртуальной машине, а подключившийся к нему клиент должен  
 работать на стационарном компьютере (или ноутбуке) в аудитории.  
7. После успешной защиты лабораторной работы остановить и удалить виртуальную машину, чтобы избежать автоматического списания денежных средств с банковской карты по окончании бесплатного периода использования AWS через 12 месяцев или окончании пробного периода Yandex Cloud через 60 дней.

**Результат выполнения работы**

**server.cpp**

****

**client.cpp**

****

**Исходный код программы с комментариями**

**server.cpp**

// This code is not protected by any copyright as far as I know

// It is based on misc public sources on the Internet

// Prepared by Ekaterina Ilchenko in 2022

#include <stdio.h>

#include <winsock2.h>

#include <locale.h>

#include <iostream>

#include <string>

#pragma comment(lib, "ws2\_32.lib")

#pragma warning(disable: 4996)

DWORD idthr;

//Функция для проверки данных, введенных пользователем

bool checkdata(char\* str);

//Функция для напоминаний

DWORD WINAPI hint(void\* g);

struct forclient {

SOCKET getsock;

struct sockaddr\_in soccl;

};

void main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "Russian");

// инициализируем

WSADATA wsaData;

int iResult = WSAStartup(MAKEWORD(2, 2), &wsaData);

if (iResult != NO\_ERROR)

printf("Ошибка в WSAStartup()\n");

//создание сокета

SOCKET soc;

soc = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

if (soc == INVALID\_SOCKET) {

printf("Error at socket(): %ld\n", WSAGetLastError());

WSACleanup();

return;

}

std::string IP;

int p;

printf("enter the IP:\n");

std::cin >> IP;

printf("enter the port:\n");

std::cin >> p;

// привязываем сокет

sockaddr\_in ser;

ser.sin\_family = AF\_INET; // структура для хранения информации о сокете

ser.sin\_addr.s\_addr = inet\_addr(IP.c\_str());

ser.sin\_port = htons(p);

if (bind(soc, reinterpret\_cast<SOCKADDR\*>(&ser),

sizeof(ser)) == SOCKET\_ERROR) {

printf("bind() failed\n");

closesocket(soc);

return;

}

// слушаем

if (listen(soc, 1) == SOCKET\_ERROR)

printf("Error listening on socket\n");

while (1) {

// принимаем соединение

SOCKET Sock;

printf("waiting for the client to connect...\n");

while (1) {

Sock = SOCKET\_ERROR;

while (Sock == SOCKET\_ERROR) {

struct sockaddr\_in client;

int len = sizeof(client);

Sock = accept(soc, (struct sockaddr\*)&client,

&len); //ожидание подключения

forclient\* data = new forclient{ Sock,

client }; //создаем клиента

CreateThread(NULL, 0, hint,

reinterpret\_cast<void\*>(data), 0,

&idthr); // создаем поток для клиента

if (idthr == NULL) {

printf("error when creating a stream!\n");

return;

}

}

break;

}

}

return;

}

//Проверка данных, введенных пользователем

bool checkdata(char\* str) {

std::string line(str);

std::string time;

int c = 0;

size\_t pos = 0;

for (int i = 0; i < line.length(); i++) {

if (line[i] == '|')

c++;

}

if (c == 1) {

pos = line.find("|");

time = line.substr(pos + 1);

for (int i = 0; i < time.length(); i++) {

if (!isdigit(time[i])) {

return false;

}

}

}

else {

return false;

}

return true;

}

//Функция для напоминаний

DWORD WINAPI hint(void\* g) {

forclient\* data = reinterpret\_cast<forclient\*>(g);

SOCKET getsock = data->getsock;

struct sockaddr\_in soccl = data->soccl;

printf("a new client is connected\n");

while (1) {

int postb; //сколько байтов отправили

int getb = SOCKET\_ERROR; //сколько байтов получили

char sbuf[32] = "";

char bpost[32] = "";

std::string message = "";

int time = 0;

std::string r;

getb = recv(getsock, bpost, 32, 0);

if (getb == SOCKET\_ERROR)

break;

if (bpost[0] == '\0')

break;

//проверим полученные данные

bool check = checkdata(bpost);

if (!check) {

r = "Data entered incorrectly!";

}

else {

printf("received from the client: ");

printf("%s", bpost);

printf("\n");

std::string str(bpost);

size\_t pos = 0;

pos = str.find("|");

message = str.substr(0, pos);

time = std::stoi(str.substr(pos + 1));

r = message;

}

while (1) {

postb = SOCKET\_ERROR;

r.copy(sbuf, 32);

postb = send(getsock, sbuf, strlen(sbuf), 0);

if (postb < 0 || postb == SOCKET\_ERROR ||

r == "Data entered incorrectly!") {

printf("connection interrupted\n");

break;

}

printf("the data has been transferred to the client:");

printf("%s", sbuf);

printf("\n");

Sleep(time);

}

printf("the client disconnected\n");

closesocket(getsock);

delete(data);

return 0;

}

}

client.cpp

// This code is not protected by any copyright as far as I know

// It is based on misc public sources on the Internet

// Prepared by Ekaterina Ilchenko in 2022

#include <unistd.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <iostream>

#include <string>

#include <signal.h>

int m\_socket;

void signal\_callback\_handler(int signum);

void signal\_callback\_handler(int signum) {

close(m\_socket);

//Прерывание програмы

exit(signum);

}

int main() {

signal(SIGINT, signal\_callback\_handler);

setlocale(LC\_ALL, "rus");

struct sockaddr\_in fordata;

m\_socket = socket(AF\_INET, SOCK\_STREAM, 0);

if (m\_socket < 0) {

perror("socket");

exit(1);

}

std::string IP;

unsigned short P;

std::cout << "Введите IP" << std::endl;

std::cin >> IP;

std::cout << "Введите порт" << std::endl;

std::cin >> P;

std::cin.clear();

std::cin.ignore();

fordata.sin\_family = AF\_INET;

fordata.sin\_port = htons(P);

fordata.sin\_addr.s\_addr = inet\_addr(IP.c\_str());

if (connect(m\_socket, reinterpret\_cast<struct sockaddr\*>(&fordata), sizeof(fordata)) < 0) {

perror("connect");

exit(2);

}

bool m = false;

while (1) {

char sbuf[32] = "";

char bpost[32] = "";

int postb;

int getb;

getb = SO\_ERROR;

if (!m) {

std::string messages;

printf("Введите напоминание и вермя повторения:напоминание|Введите время\n");

getline(std::cin, messages);

messages.copy(sbuf, 32);

postb = send(m\_socket, sbuf, sizeof(sbuf), 0);

std::cout << "Отправлено на сервер: " << sbuf << std::endl;

m = true;

}

while (getb == SO\_ERROR) {

getb = recv(m\_socket, bpost, sizeof(bpost), 0);

if (getb == 0 || getb == ECONNRESET) {

std::cout << "Соединение прервано" << std::endl;

return 1; }

if (getb < 0)

return 0;

}

std::cout << "Получено от сервера:" << bpost << std::endl;

}

close(m\_socket);

return 0;

}

**Выводы:**

Изучили принципы работы с системами виртуализации в облаке.