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 №6 | | | | | | | | | | |
| Межсетевое взаимодействие между процессами | | | | | | | | | | |
| 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:

Организация межсетевого взаимодействия средствами WinAPI и POSIX.

#### Задание:

Организовать взаимодействие типа клиент-сервер средствами WinAPI и POSIX в соответствии с индивидуальным заданием.

1. Вычислить номер варианта по списку в журнале и сохранить его в файл [TASKID.txt](https://github.com/suai-os-2022/os-task6-Kateilc/blob/master/TASKID.txt) в репозитории.
2. Выбрать индивидуальное задание в соответствии с номером варианта. По номеру варианта также определить:

* протокол, с использованием которого должен происходить обмен данными между сервером и клиентом;
* операционную систему, под которую необходимо разработать приложение сервера;
* операционную систему, под которую необходимо разработать приложение клиента.

1. В разделе [Issues](https://github.com/suai-os-2022/os-task6-Kateilc/issues) репозитория создать не менее трех задач. Например: "Разработка приложения сервера", "Разработка приложения клиента", "Отладка клиент-серверного взаимодействия". Последовательно выполнить эти задачи, написав код и разместив его в репозитории. Решению каждой задачи должен соответствовать свой отдельный коммит, который должен быть [привязан](https://github.blog/2011-04-09-issues-2-0-the-next-generation/#commits-issues) к конкретной задаче. Итого в репозитории должно быть не менее трех коммитов. При необходимости создать дополнительные задачи.
2. Код приложения клиента необходимо разместить в файле client.cpp в корне репозитория, код приложения сервера - в файле server.cpp там же. При необходимости использовать дополнительные заголовочные файлы. Код, предназначенный для выполнения в ОС Linux, должен собираться командной g++ client.cpp или g++ server.cpp.
3. В репозитории имеется тест, проверяющий корректность оформления кода в соответствии с [Google C++ Style Guide](https://google.github.io/styleguide/cppguide.html). Данный тест запускает [линтер](https://ru.wikipedia.org/wiki/Lint), который проверяет соответствие кода стандарту языка и правилам оформления (отступы, разделители, комментарии и т.п.). В некоторых случаях линтер может дать совет, как улучшить код используя общепринятые практики.
4. Автоматическое тестирование работоспособности кода отсутствует. Необходимо загрузить рабочий код в репозиторий, а затем защитить лабораторную работу у преподавателя, продемонстрировав корректную совместную работу клиентского и серверного приложений.
5. Допускается выполнение задания не в полном объеме. Правила начисления рейтинга для этого случая приведены в разделе [Рейтинг](https://github.com/suai-os-2022/os-task6-Kateilc#%D0%A0%D0%B5%D0%B9%D1%82%D0%B8%D0%BD%D0%B3).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер варианта | Индивидуальное задание | Протокол | Сервер | Клиент |
| 11 | 11 | |  |  | | --- | --- | | TCP |  | | Windows | Linux |

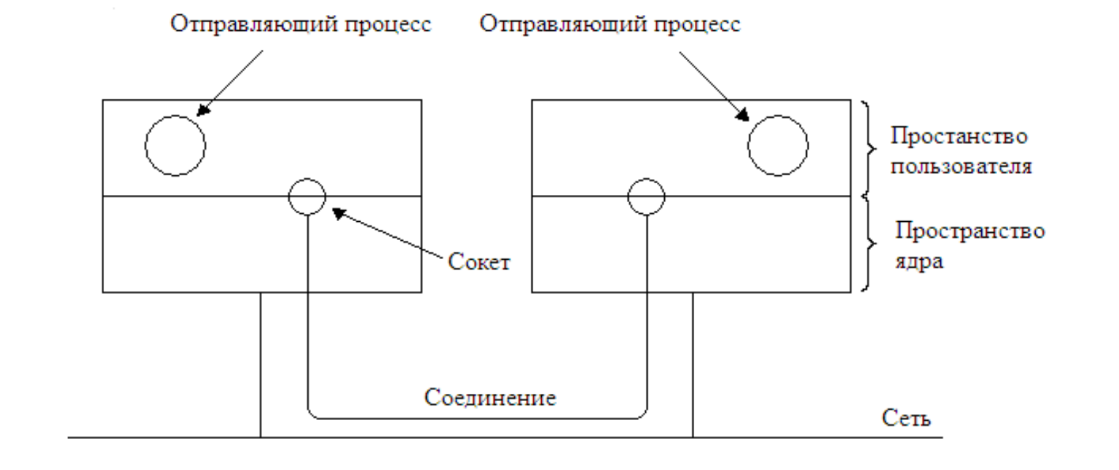
Общие требования для всех заданий:

* клиент и сервер являются консольными приложениями;
* номер порта, на котором работает сервер, указывается при его запуске в качестве аргумента командной строки;
* доменное имя (ip-адрес), на котором работает сервер, а также его номер порта указываются в качестве аргументов командной строки при запуске клиента;
* сервер выводит в консоль все сообщения, которые получает от клиента;
* клиент выводит в консоль все сообщения, которые получает от сервера.

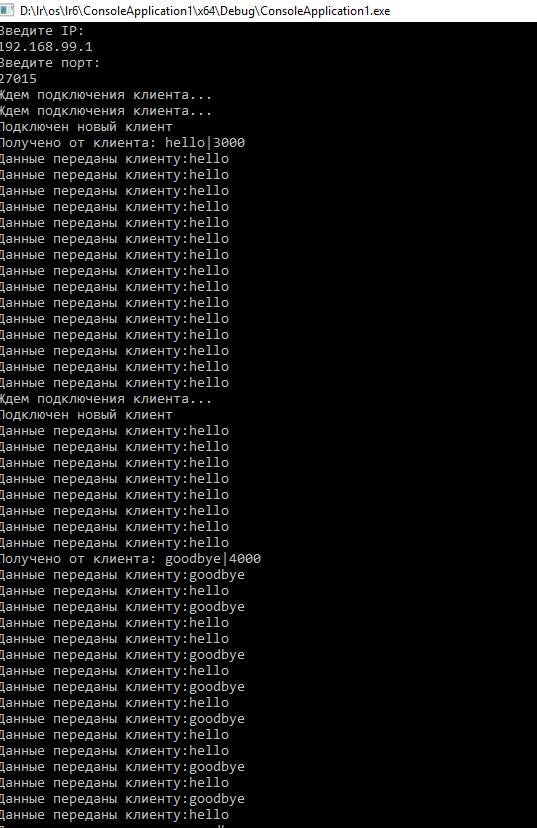
**Индивидуальное задание:**

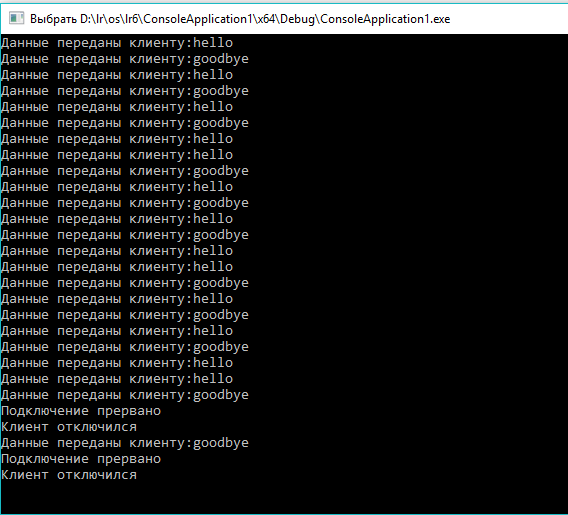
**Напоминание**. Приложение-клиент запрашивает у пользователя ввод строки и целого числа N и передает их на сервер. Сервер начинает отправлять клиенту полученную строку каждые N секунд до тех пор, пока клиент не отключится.

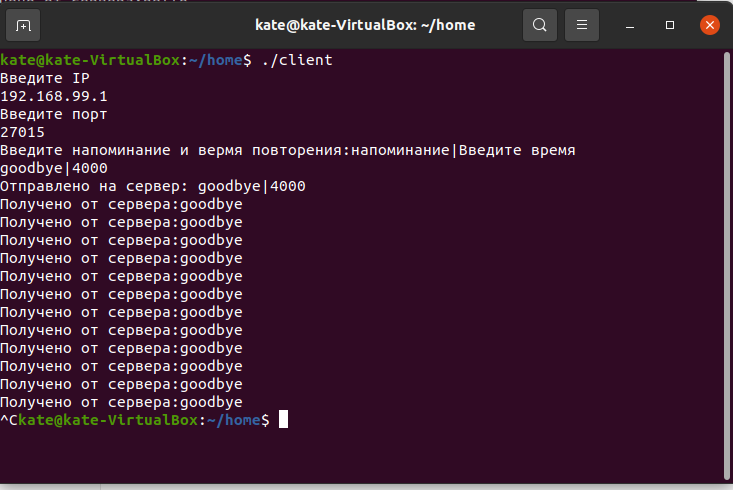
**Схема взаимодействия между клиентом и сервером:**

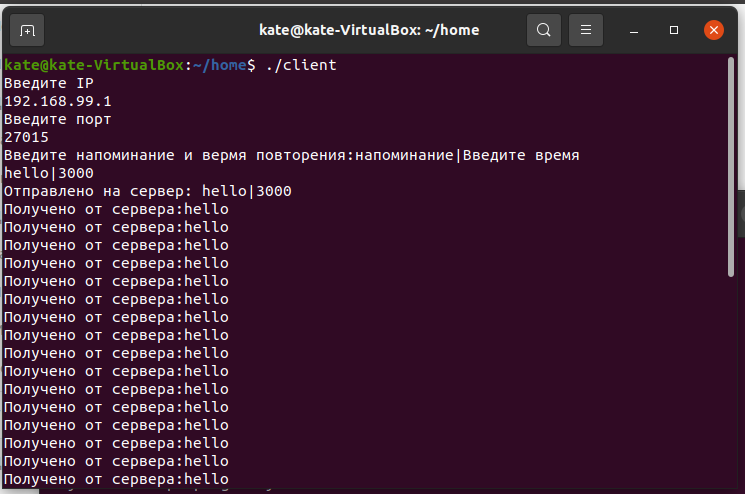
****

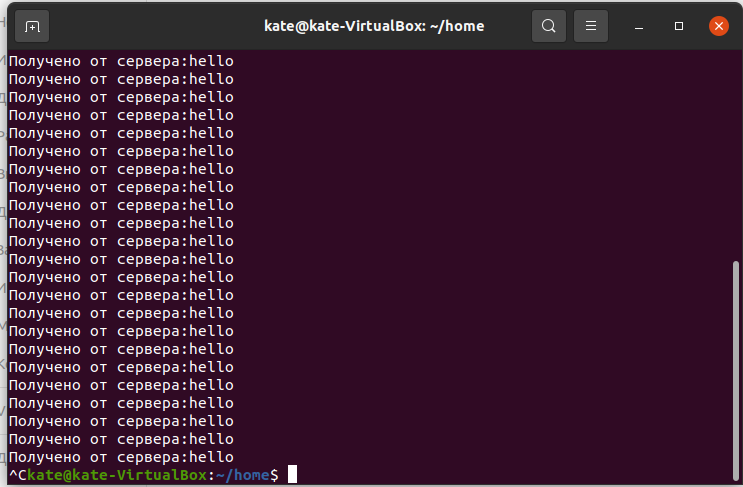
**Результат выполнения лабораторной работы:**

****

****

****

****

****

**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 <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);

// 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;

unsigned short p;

printf("Введите IP:\n");

std::cin >> IP;

printf("Введите порт:\n");

std::cin >> p;

// привязываем сокет

sockaddr\_in ser;

ser.sin\_family = AF\_INET;

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("Ждем подключения клиента...\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);

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("Ошибка при создании потока!\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("Подключен новый клиент\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("Получено от клиента: ");

printf(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("Подключение прервано\n");

break;

}

printf("Данные переданы клиенту:");

printf(sbuf);

printf("\n");

Sleep(time);

}

printf("Клиент отключился\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 закончено" << std::endl;

return 1; }

if (getb < 0)

return 0;

}

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

}

close(m\_socket);

return 0;

} **Выводы:** В процессе выполнения лабораторной работы ознакомились с принципами организации межсетевого взаимодействия средствами WinAPI и POSIX.