МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Череповецкий государственный университет»

**Лабораторная работа № 3**

**ТЕХНОЛОГИИ ПЕРЕДАЧИ ДАННЫХ В ЛОКАЛЬНЫХ СЕТЯХ. РАЗРАБОТКА СЕТЕВОГО ПРИЛОЖЕНИЯ ДЛЯ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ**

**Выполнил:**

студент гр. 1ИВТпб-01-31оп

Климов А.Г.  
**Проверил:** преподаватель

Селяничев О.Л.  
Отметка о зачете:

Череповец

2018 год

Цель работы: Определить правила взаимодействия приложений, построенных по технологии клиент-сервер.

**Задание**

Создать приложения, работающие по технологии клиент-сервер.

Серверная часть должна выполнять следующие функции:

1. по запросу с клиентской части определить локальные параметры:

 собственный IP-адрес

 доменное имя

 текущего пользователя

 тип операционной системы

2. в случае успеха послать сформированные данные клиентскому приложению.

Клиентская часть должна выполнять следующие функции:

1. послать запрос серверной части о формировании данных

2. получить сформированные данные

3. отобразить полученные данные

4. обеспечивать работу с несколькими серверными частями.

**Текст программы Client.cpp**

// consolClient.cpp: определяет точку входа для консольного приложения.

//

//etcp. h

#include "stdafx.h"

#include <stdio.h>

#include <string.h>

#include <winsock2.h>

#include <windows.h>

#define PORT 666

#define SERVERADDR "127.0.0.1"

int \_tmain(int argc, \_TCHAR\* argv[])

{

char buff[BUFSIZ];

printf("TCP DEMO CLIENT\n");

// Шаг 1 - инициализация библиотеки Winsock

if (WSAStartup(0x202, (WSADATA \*)&buff[0]))

{

printf("WSAStart error %d\n", WSAGetLastError());

return -1;

}

// Шаг 2 - создание сокета

SOCKET my\_sock;

my\_sock = socket(AF\_INET, SOCK\_STREAM, 0);

if (my\_sock < 0)

{

printf("Socket() error %d\n", WSAGetLastError());

return -1;

}

// Шаг 3 - установка соединения

// заполнение структуры sockaddr\_in - указание адреса и порта сервера

sockaddr\_in dest\_addr;

dest\_addr.sin\_family = AF\_INET;

dest\_addr.sin\_port = htons(PORT);

HOSTENT \*hst;

// преобразование IP адреса из символьного в сетевой формат

if (inet\_addr(SERVERADDR) != INADDR\_NONE)

dest\_addr.sin\_addr.s\_addr = inet\_addr(SERVERADDR);

else

{

// попытка получить IP адрес по доменному имени сервера

if (hst = gethostbyname(SERVERADDR))

// hst->h\_addr\_list содержит не массив адресов,

// а массив указателей на адреса

((unsigned long \*)&dest\_addr.sin\_addr)[0] =

((unsigned long \*\*)hst->h\_addr\_list)[0][0];

else

{

printf("Invalid address %s\n", SERVERADDR);

closesocket(my\_sock);

WSACleanup();

return -1;

}

}

// адрес сервера получен - пытаемся установить соединение

if (connect(my\_sock, (sockaddr \*)&dest\_addr, sizeof(dest\_addr)))

{

printf("Connect error %d\n", WSAGetLastError());

return -1;

}

printf("Soedinenie s %s uspeshno ustanovlenno\n \

Type quit for quit\n\n", SERVERADDR);

// Шаг 4 - чтение и передача сообщений

int nsize;

while ((nsize = recv(my\_sock, &buff[0], sizeof(buff) - 1, 0)) != SOCKET\_ERROR)

{

// ставим завершающий ноль в конце строки

buff[nsize] = 0;

// выводим на экран

printf("S=>C:%s", buff);

// читаем пользовательский ввод с клавиатуры

printf("S<=C:"); fgets(&buff[0], sizeof(buff) - 1, stdin);

// проверка на "quit"

if (!strcmp(&buff[0], "quit\n"))

{

// Корректный выход

printf("Exit...");

closesocket(my\_sock);

WSACleanup();

return 0;

}

// передаем строку клиента серверу

send(my\_sock, &buff[0], strlen(&buff[0]), 0);

}

printf("Recv error %d\n", WSAGetLastError());

closesocket(my\_sock);

WSACleanup();

return -1;

//return 0;

}

**Текст программы Server.cpp**

// server.cpp: определяет точку входа для консольного приложения.

//

#include "stdafx.h"

#define WIN32\_LEAN\_AND\_MEAN

#define UNLEN 256

//#include <windows.h>

//#include <winsock2.h>

//#include <ws2tcpip.h>

//#include <stdio.h>

// Need to link with Ws2\_32.lib

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

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

#include <iostream>

#include <stdio.h>

#include <winsock2.h> // Wincosk2.h должен быть раньше windows!

#include <windows.h>

using namespace std;

#define MY\_PORT 666 // Порт, который слушает сервер 666

// макрос для печати количества активных пользователей

#define PRINTNUSERS if (nclients) printf("%d User on-line\n", nclients); \

else printf("No User on line\n");

// прототип функции, обслуживающий подключившихся пользователей

DWORD WINAPI SexToClient(LPVOID client\_socket);

//прототип функции обмена слов

void changeWords();

// глобальная переменная - количество активных пользователей

int nclients = 0;

//глобальная переменная, в которой храняться принятые от сервера данные

char \*tmp\_buff = new char;

char \*tmp\_buff\_obmen = new char;

char \*tmp\_buff\_obmen2 = new char;

//char buff[20 \* 1024];

int \_tmain(int argc, \_TCHAR\* argv[])

{

char buff[1024]; // Буфер для различных нужд

printf("TCP SERVER\n");

// Шаг 1 - Инициализация Библиотеки Сокетов

// т.к. возвращенная функцией информация не используется

// ей передается указатель на рабочий буфер, преобразуемый к указателю

// на структуру WSADATA.

// Такой прием позволяет сэкономить одну переменную, однако, буфер

// должен быть не менее полкилобайта размером (структура WSADATA

// занимает 400 байт)

if (WSAStartup(0x0202, (WSADATA \*)&buff[0]))

{

// Ошибка!

printf("Error WSAStartup %d\n", WSAGetLastError());

return -1;

}

// Шаг 2 - создание сокета

SOCKET mysocket;

// AF\_INET - сокет Интернета

// SOCK\_STREAM - потоковый сокет (с установкой соединения)

// 0 - по умолчанию выбирается TCP протокол

if ((mysocket = socket(AF\_INET, SOCK\_STREAM, 0)) < 0)

{

// Ошибка!

printf("Error socket %d\n", WSAGetLastError());

WSACleanup(); // Деиницилизация библиотеки Winsock

return -1;

}

// Шаг 3 - связывание сокета с локальным адресом

sockaddr\_in local\_addr;

local\_addr.sin\_family = AF\_INET;

local\_addr.sin\_port = htons(MY\_PORT); // не забываем о сетевом порядке!!!

local\_addr.sin\_addr.s\_addr = 0; // сервер принимает подключения

// на все свои IP-адреса

// вызываем bind для связывания

if (bind(mysocket, (sockaddr \*)&local\_addr, sizeof(local\_addr)))

{

// Ошибка

printf("Error bind %d\n", WSAGetLastError());

closesocket(mysocket); // закрываем сокет!

WSACleanup();

return -1;

}

// Шаг 4 - ожидание подключений

// размер очереди - 0x100

if (listen(mysocket, 0x100))

{

// Ошибка

printf("Error listen %d\n", WSAGetLastError());

closesocket(mysocket);

WSACleanup();

return -1;

}

char ac[80];

if (gethostname(ac, sizeof(ac)) == SOCKET\_ERROR) {

cerr << "Error " << WSAGetLastError() <<

" when getting local host name." << endl;

}

cout << "Host name is " << ac << "." << endl;

struct hostent \*phe = gethostbyname(ac);

if (phe == 0) {

cerr << "Yow! Bad host lookup." << endl;

return 1;

}

for (int i = 0; phe->h\_addr\_list[i] != 0; ++i) {

struct in\_addr addr;

memcpy(&addr, phe->h\_addr\_list[i], sizeof(struct in\_addr));

cout << "Address " << i << ": " << inet\_ntoa(addr) << endl;

}

char buffer[UNLEN+1]; // буфер

DWORD size; // размер

size=sizeof(buffer); // размер буфера

GetUserName(buffer,&size);

cout<< "User: " << buffer<<endl;

//getchar();

char\* ostype = getenv("OSTYPE");

if (ostype == NULL)

{

// не угадали. попробуем, а вдруг это виндовс!?

ostype = getenv("windir");

// Ищет в списке переменных среды, возвращаемом окружением (операционной системой) строку, соответствующую

// указателю на строку в стиле Си, указанной в env\_var, и возвращает указатель на строку в стиле Си, которая

// содержит значение найденной переменной среды.

if (ostype != NULL)

{ printf ("windows catalog is found!!!\n");}

}

else

{

// переменная окружения есть, попробуем вызнать что же это такое.

if(strcmp(ostype, "linux") == 0)

printf("linux\n");

else if(strcmp(ostype, "hpux") == 0)

printf("hpux\n");

else if(strcmp(ostype, "solaris") == 0)

printf("solaris\n");

else if(strcmp(ostype, "darwin") == 0)

printf("darwin\n");

}

printf("ozidanie podkluceniy...\n");

// Шаг 5 - извлекаем сообщение из очереди

SOCKET client\_socket; // сокет для клиента

sockaddr\_in client\_addr; // адрес клиента (заполняется системой)

// функции accept необходимо передать размер структуры

int client\_addr\_size = sizeof(client\_addr);

// цикл извлечения запросов на подключение из очереди

while ((client\_socket = accept(mysocket, (sockaddr \*)&client\_addr, \

&client\_addr\_size)))

{

nclients++; // увеличиваем счетчик подключившихся клиентов

// пытаемся получить имя хоста

HOSTENT \*hst;

hst = gethostbyaddr((char \*)&client\_addr. sin\_addr.s\_addr, 4, AF\_INET);

// вывод сведений о клиенте

printf("+%s [%s] new connect!\n",

(hst) ? hst->h\_name : "", inet\_ntoa(client\_addr.sin\_addr));

PRINTNUSERS

// Вызов нового потока для обслужвания клиента

// Да, для этого рекомендуется использовать \_beginthreadex

// но, поскольку никаких вызовов функций стандартной Си библиотеки

// поток не делает, можно обойтись и CreateThread

DWORD thID;

CreateThread(NULL, NULL, SexToClient, &client\_socket, NULL, &thID);

cout<<"Thread for new client created!" <<endl;

}

return 0;

}

// Эта функция создается в отдельном потоке

// и обсуживает очередного подключившегося клиента независимо от остальных

DWORD WINAPI SexToClient(LPVOID client\_socket)

{

SOCKET my\_sock;

my\_sock = ((SOCKET \*)client\_socket)[0];

char buff[BUFSIZ];//char buff[20 \* 1024];

#define sHELLO "SOCKET PODKLUCHEN\r\n"

// отправляем клиенту приветствие

send(my\_sock, sHELLO, sizeof(sHELLO), 0);

// цикл эхо-сервера: прием строки от клиента и возвращение ее клиенту

int bytes\_recv;

while ((bytes\_recv = recv(my\_sock, &buff[0], sizeof(buff), 0)) &&

bytes\_recv != SOCKET\_ERROR)

//\*tmp\_buff = buff[0];

//changeWords();

send(my\_sock, &buff[0], bytes\_recv, 0);

// если мы здесь, то произошел выход из цикла по причине

// возращения функцией recv ошибки - соединение с клиентом разорвано

nclients--; // уменьшаем счетчик активных клиентов

printf("-disconnect\n"); PRINTNUSERS

// закрываем сокет

closesocket(my\_sock);

return 0;

}

void changeWords()

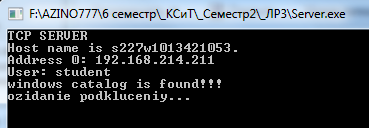
{

cout << tmp\_buff;

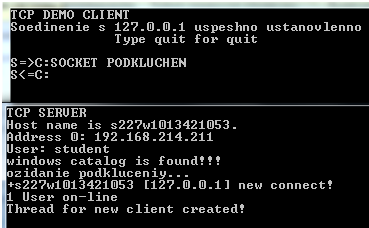
}

Тестирование:

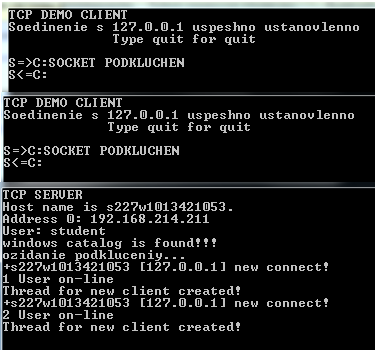
Информация о сервере



Подключение 1-го клиента



Подключение 2-го клиента



Обновление статуса сервера при отключении клиентов

