Учреждение образования

«Белорусский государственный технологический университет»

Кафедра информационных систем и технологий

Лабораторная работа №1

«Синхронизация времени»

Выполнил:

Студент 3 курса 4 группы ФИТ

Мелентьев Никита

2018 г.

**Цель**

Изначально в данной лабораторной работе были созданы клиент и сервер, способные синхронизировать время на удаленных машинах. Для этого использовались клиенты и параллельный сервер, способный обслуживать несколько клиентов одновременно.

Второй частью является синхронизация времени клиента и сервера с помощью протокола NTP.

**Синхронизация часов (I)**

Клиент периодически шлет серверу запросы следующей структуры.

struct GETSINCHRO

{

string cmd;

int curvalue;

};

Периодичность клиентского запроса ***Tc*** является параметром программы-клиента и задается в тиках, равных **1/1000 сек**.

Сервер без задержки отвечает клиенту на каждый запрос ответом, имеющим следующую структуру.

struct SETSINCRO

{

string cmd;

int correction;

};

Для каждого эксперимента принимается устанавливается значение ***Tc***. В каждом эксперименте клиент должен делать по 10 запросов

**Код структуры Сервера**

#include "stdafx.h"

#include "iostream"

#include "string.h"

#include "locale"

#include "time.h"

#include "Winsock2.h"

#pragma comment(lib, "WS2\_32.lib")

using namespace std;

struct SETSINCRO

{

string cmd;

int correction;

};

string GetErrorMsgText (int code)

{

string msgText;

switch (code)

{

case WSAEINTR: msgText = "Работа функции прервана\n"; break;

case WSAEACCES: msgText = "Разрешение отвергнуто\n"; break;

case WSAEFAULT: msgText = "Ошибочный адрес\n"; break;

case WSAEINVAL: msgText = "Ошибка в аргументе\n"; break;

case WSAEMFILE: msgText = "Слишком много файлов открыто\n"; break;

case WSAEWOULDBLOCK: msgText = "Ресурс временно недоступен\n"; break;

case WSAEINPROGRESS: msgText = "Операция в процессе развития\n"; break;

case WSAEALREADY: msgText = "Операция уже выполняется\n"; break;

case WSAENOTSOCK: msgText = "Сокет задан неправильно\n"; break;

case WSAEDESTADDRREQ: msgText = "Требуется адрес расположения\n"; break;

case WSAEMSGSIZE: msgText = "Сообщение слишком длинное\n"; break;

case WSAEPROTOTYPE: msgText = "Неправильный тип протокола для сокета\n"; break;

case WSAENOPROTOOPT: msgText = "Ошибка в опции протокола\n"; break;

case WSAEPROTONOSUPPORT: msgText = "Протокол не поддерживается\n"; break;

case WSAESOCKTNOSUPPORT: msgText = "Тип сокета не поддерживается\n"; break;

case WSAEOPNOTSUPP: msgText = "Операция не поддерживается\n"; break;

case WSAEPFNOSUPPORT: msgText = "Тип протоколов не поддерживается\n"; break;

case WSAEAFNOSUPPORT: msgText = "Тип адресов не поддерживается протоколом\n"; break;

case WSAEADDRINUSE: msgText = "Адрес уже используется\n"; break;

case WSAEADDRNOTAVAIL: msgText = "Запрошенный адрес не может быть использован\n"; break;

case WSAENETDOWN: msgText = "Сеть отключена\n"; break;

case WSAENETUNREACH: msgText = "Сеть не достижима\n"; break;

case WSAENETRESET: msgText = "Сеть разорвала соединение\n"; break;

case WSAECONNABORTED: msgText = "Программный отказ связи\n"; break;

case WSAECONNRESET: msgText = "Связь восстановлена\n"; break;

case WSAENOBUFS: msgText = "Не хватает памяти для буферов\n"; break;

case WSAEISCONN: msgText = "Сокет уже подключен\n"; break;

case WSAENOTCONN: msgText = "Сокет не подключен\n"; break;

case WSAESHUTDOWN: msgText = "Нельзя выполнить send: сокет завершил работу\n"; break;

case WSAETIMEDOUT: msgText = "Закончился отведенный интервал времени\n"; break;

case WSAECONNREFUSED: msgText = "Соединение отклонено\n"; break;

case WSAEHOSTDOWN: msgText = "Хост в неработоспособном состоянии\n"; break;

case WSAEHOSTUNREACH: msgText = "Нет маршрута для хоста\n"; break;

case WSAEPROCLIM: msgText = "Слишком много процессов\n"; break;

case WSASYSNOTREADY: msgText = "Сеть не доступна\n"; break;

case WSAVERNOTSUPPORTED: msgText = "Данная версия недоступна\n"; break;

case WSANOTINITIALISED: msgText = "Не выполнена инициализация WS2\_32.DLL\n"; break;

case WSAEDISCON: msgText = "Выполняется отключение\n"; break;

case WSATYPE\_NOT\_FOUND: msgText = "Класс не найден\n"; break;

case WSAHOST\_NOT\_FOUND: msgText = "Хост не найден\n"; break;

case WSATRY\_AGAIN: msgText = "Неавторизированный хост не найден\n"; break;

case WSANO\_RECOVERY: msgText = "Неопределенная ошибка\n"; break;

case WSANO\_DATA: msgText = "Нет записи запрошенного типа\n"; break;

case WSA\_INVALID\_HANDLE: msgText = "Указанный дескриптор события с ошибкой\n"; break;

case WSA\_INVALID\_PARAMETER: msgText = "Один или более параметров с ошибкой\n"; break;

case WSA\_IO\_INCOMPLETE: msgText = "Объект ввода-вывода не в сигнальном состоянии\n"; break;

case WSA\_IO\_PENDING: msgText = "Операция завершится позже\n"; break;

case WSA\_NOT\_ENOUGH\_MEMORY: msgText = "Не достаточно памяти\n"; break;

case WSA\_OPERATION\_ABORTED: msgText = "Операция отвергнута\n"; break;

case WSAEINVALIDPROCTABLE: msgText = "Ошибочный сервис\n"; break;

case WSAEINVALIDPROVIDER: msgText = "Ошибка в версии сервиса\n"; break;

case WSAEPROVIDERFAILEDINIT: msgText = "Невозможно инициализировать сервис\n"; break;

case WSASYSCALLFAILURE: msgText = "Аварийное завершение системного вызова\n"; break;

default: msgText = "Error\n"; break;

};

return msgText;

}

string SetErrorMsgText (string msgText, int code)

{

return msgText + GetErrorMsgText (code);

};

int setAverageCorrection(int averageCorrection[], int length)

{

int value = 0;

for (int i = 0; i < length; i++)

value += averageCorrection[i];

return value / length;

}

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

{ setlocale(LC\_ALL, "Russian");

SETSINCRO setsincro, getsincro;

ZeroMemory(&setsincro, sizeof(setsincro));

ZeroMemory(&getsincro, sizeof(getsincro));

setsincro.cmd = "SINCRO";

setsincro.correction = 0;

SYSTEMTIME tm;

clock\_t c;

int averageCorrection[10];

cout << "Сервер запущен" << endl;

try

{

SOCKET sS;

WSADATA wsaData;

if (WSAStartup(MAKEWORD(2,0), &wsaData) != 0)

throw SetErrorMsgText("Startup: ", WSAGetLastError());

if ((sS = socket(AF\_INET, SOCK\_DGRAM, NULL)) == INVALID\_SOCKET)

throw SetErrorMsgText("Socket: ", WSAGetLastError());

SOCKADDR\_IN serv;

serv.sin\_family = AF\_INET;

serv.sin\_port = htons(2000);

serv.sin\_addr.s\_addr = INADDR\_ANY;

if (bind(sS, (LPSOCKADDR)&serv, sizeof(serv)) == SOCKET\_ERROR)

throw SetErrorMsgText("Bind\_Server: ", WSAGetLastError());

int count = 1;

while(count != 11)

{

SOCKADDR\_IN client;

int lc = sizeof(client);

int average = 0;

GetSystemTime(&tm);

recvfrom(sS, (char \*)&getsincro, sizeof(getsincro), NULL, (sockaddr\*)&client, &lc);

c = clock();

setsincro.correction = c - getsincro.correction;

averageCorrection[count - 1] = c - getsincro.correction;

average = setAverageCorrection(averageCorrection, count);

sendto(sS, (char \*)&setsincro, sizeof(setsincro), 0, (sockaddr\*)&client, sizeof(client));

cout << count << " " << " Дата и время " << tm.wMonth << "." << tm.wDay << ".2018" << " " << endl << tm.wHour + 3 << " Часов " << tm.wMinute << " Минут " << tm.wSecond << " Секунд " << tm.wMilliseconds << " Миллисекунд " << endl << inet\_ntoa(client.sin\_addr) << "Коррекция = " << setsincro.correction << ", Средняя коррекция = " << average << endl;

count++;

}

if(closesocket(sS) == SOCKET\_ERROR)

throw SetErrorMsgText("close socket: ", WSAGetLastError());

if(WSACleanup() == SOCKET\_ERROR)

throw SetErrorMsgText("Cleanup: ", WSAGetLastError());

}

catch (string errorMsgText)

{

cout << endl << errorMsgText;

}

return 0;

**Код структуры клиента**

#include "stdafx.h"

#include "Winsock2.h"

#include <iostream>

#include <string>

#include <ctime>

#pragma comment(lib, "WS2\_32.lib")

using namespace std;

struct GETSINCHRO

{

string cmd;

int curvalue;

};

string GetErrorMsgText (int code)

{

string msgText;

switch (code)

{

case WSAEINTR: msgText = "Работа функции прервана\n"; break;

case WSAEACCES: msgText = "Разрешение отвергнуто\n"; break;

case WSAEFAULT: msgText = "Ошибочный адрес\n"; break;

case WSAEINVAL: msgText = "Ошибка в аргументе\n"; break;

case WSAEMFILE: msgText = "Слишком много файлов открыто\n"; break;

case WSAEWOULDBLOCK: msgText = "Ресурс временно недоступен\n"; break;

case WSAEINPROGRESS: msgText = "Операция в процессе развития\n"; break;

case WSAEALREADY: msgText = "Операция уже выполняется\n"; break;

case WSAENOTSOCK: msgText = "Сокет задан неправильно\n"; break;

case WSAEDESTADDRREQ: msgText = "Требуется адрес расположения\n"; break;

case WSAEMSGSIZE: msgText = "Сообщение слишком длинное\n"; break;

case WSAEPROTOTYPE: msgText = "Неправильный тип протокола для сокета\n"; break;

case WSAENOPROTOOPT: msgText = "Ошибка в опции протокола\n"; break;

case WSAEPROTONOSUPPORT: msgText = "Протокол не поддерживается\n"; break;

case WSAESOCKTNOSUPPORT: msgText = "Тип сокета не поддерживается\n"; break;

case WSAEOPNOTSUPP: msgText = "Операция не поддерживается\n"; break;

case WSAEPFNOSUPPORT: msgText = "Тип протоколов не поддерживается\n"; break;

case WSAEAFNOSUPPORT: msgText = "Тип адресов не поддерживается протоколом\n"; break;

case WSAEADDRINUSE: msgText = "Адрес уже используется\n"; break;

case WSAEADDRNOTAVAIL: msgText = "Запрошенный адрес не может быть использован\n"; break;

case WSAENETDOWN: msgText = "Сеть отключена\n"; break;

case WSAENETUNREACH: msgText = "Сеть не достижима\n"; break;

case WSAENETRESET: msgText = "Сеть разорвала соединение\n"; break;

case WSAECONNABORTED: msgText = "Программный отказ связи\n"; break;

case WSAECONNRESET: msgText = "Связь восстановлена\n"; break;

case WSAENOBUFS: msgText = "Не хватает памяти для буферов\n"; break;

case WSAEISCONN: msgText = "Сокет уже подключен\n"; break;

case WSAENOTCONN: msgText = "Сокет не подключен\n"; break;

case WSAESHUTDOWN: msgText = "Нельзя выполнить send: сокет завершил работу\n"; break;

case WSAETIMEDOUT: msgText = "Закончился отведенный интервал времени\n"; break;

case WSAECONNREFUSED: msgText = "Соединение отклонено\n"; break;

case WSAEHOSTDOWN: msgText = "Хост в неработоспособном состоянии\n"; break;

case WSAEHOSTUNREACH: msgText = "Нет маршрута для хоста\n"; break;

case WSAEPROCLIM: msgText = "Слишком много процессов\n"; break;

case WSASYSNOTREADY: msgText = "Сеть не доступна\n"; break;

case WSAVERNOTSUPPORTED: msgText = "Данная версия недоступна\n"; break;

case WSANOTINITIALISED: msgText = "Не выполнена инициализация WS2\_32.DLL\n"; break;

case WSAEDISCON: msgText = "Выполняется отключение\n"; break;

case WSATYPE\_NOT\_FOUND: msgText = "Класс не найден\n"; break;

case WSAHOST\_NOT\_FOUND: msgText = "Хост не найден\n"; break;

case WSATRY\_AGAIN: msgText = "Неавторизированный хост не найден\n"; break;

case WSANO\_RECOVERY: msgText = "Неопределенная ошибка\n"; break;

case WSANO\_DATA: msgText = "Нет записи запрошенного типа\n"; break;

case WSA\_INVALID\_HANDLE: msgText = "Указанный дескриптор события с ошибкой\n"; break;

case WSA\_INVALID\_PARAMETER: msgText = "Один или более параметров с ошибкой\n"; break;

case WSA\_IO\_INCOMPLETE: msgText = "Объект ввода-вывода не в сигнальном состоянии\n"; break;

case WSA\_IO\_PENDING: msgText = "Операция завершится позже\n"; break;

case WSA\_NOT\_ENOUGH\_MEMORY: msgText = "Не достаточно памяти\n"; break;

case WSA\_OPERATION\_ABORTED: msgText = "Операция отвергнута\n"; break;

case WSAEINVALIDPROCTABLE: msgText = "Ошибочный сервис\n"; break;

case WSAEINVALIDPROVIDER: msgText = "Ошибка в версии сервиса\n"; break;

case WSAEPROVIDERFAILEDINIT: msgText = "Невозможно инициализировать сервис\n"; break;

case WSASYSCALLFAILURE: msgText = "Аварийное завершение системного вызова\n"; break;

default: msgText = "Error\n"; break;

};

return msgText;

}

string SetErrorMsgText (string msgText, int code)

{

return msgText + GetErrorMsgText(code);

};

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

{ setlocale(LC\_CTYPE, "Russian");

//string IP = "192.168.43.27";

string IP = "127.0.0.1";

int Tc = 14000;

int Cc = 0;

SYSTEMTIME tm;

GETSINCHRO getsincro, setsincro;

ZeroMemory(&setsincro, sizeof(setsincro));

ZeroMemory(&getsincro, sizeof(getsincro));

getsincro.cmd = "SINC";

getsincro.curvalue = 0;

cout << "Клиент запущен" << endl;

try

{

SOCKET cS;

WSADATA wsaData;

if (WSAStartup(MAKEWORD(2,0), &wsaData) != 0)

throw SetErrorMsgText("Startup: ", WSAGetLastError());

if ((cS = socket(AF\_INET, SOCK\_DGRAM, NULL)) == INVALID\_SOCKET)

throw SetErrorMsgText("Socket: ", WSAGetLastError());

SOCKADDR\_IN serv;

serv.sin\_family = AF\_INET;

serv.sin\_port = htons(2000);

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

int maxcor = 0;

int mincor = INT\_MAX;

int avgcorr = 0;

int lensockaddr = sizeof(serv);

sendto(cS, (char \*)&getsincro, sizeof(getsincro), 0, (sockaddr\*)&serv, sizeof(serv));

recvfrom(cS, (char \*)&setsincro, sizeof(setsincro), 0, (sockaddr\*)&serv, &lensockaddr);

getsincro.curvalue += setsincro.curvalue ;

for (int i = 0; i < 10; i++)

{

GetSystemTime(&tm);

sendto(cS, (char \*)&getsincro, sizeof(getsincro), 0, (sockaddr\*)&serv, sizeof(serv));

recvfrom(cS, (char \*)&setsincro, sizeof(setsincro), 0, (sockaddr\*)&serv, &lensockaddr);

maxcor = (maxcor < setsincro.curvalue) ? setsincro.curvalue : maxcor;

mincor = (mincor > setsincro.curvalue) ? setsincro.curvalue : mincor;

cout << " Дата и время " << tm.wMonth << "." << tm.wDay << ".2018" << " " << endl << tm.wHour + 3 << " Часов " << tm.wMinute << " Минут " << tm.wSecond << " Секунд " << tm.wMilliseconds << " Миллисекунд " << endl << i + 1 << " " << getsincro.curvalue << " Коррекция = " << setsincro.curvalue << " Максимальная/минимальная коррекции соответственно: " << maxcor << "/" << mincor << endl;

getsincro.curvalue += setsincro.curvalue + Tc;

avgcorr += setsincro.curvalue;

Sleep(Tc);

}

cout << "Средняя коррекция: " << avgcorr / 10 << endl;

if (closesocket(cS) == SOCKET\_ERROR)

throw SetErrorMsgText("Closesocket: ", WSAGetLastError());

if (WSACleanup() == SOCKET\_ERROR)

throw SetErrorMsgText("Cleanup: ", WSAGetLastError());

}

catch (string errorMsgText)

{

cout << endl << errorMsgText << endl;

}

return 0;

}

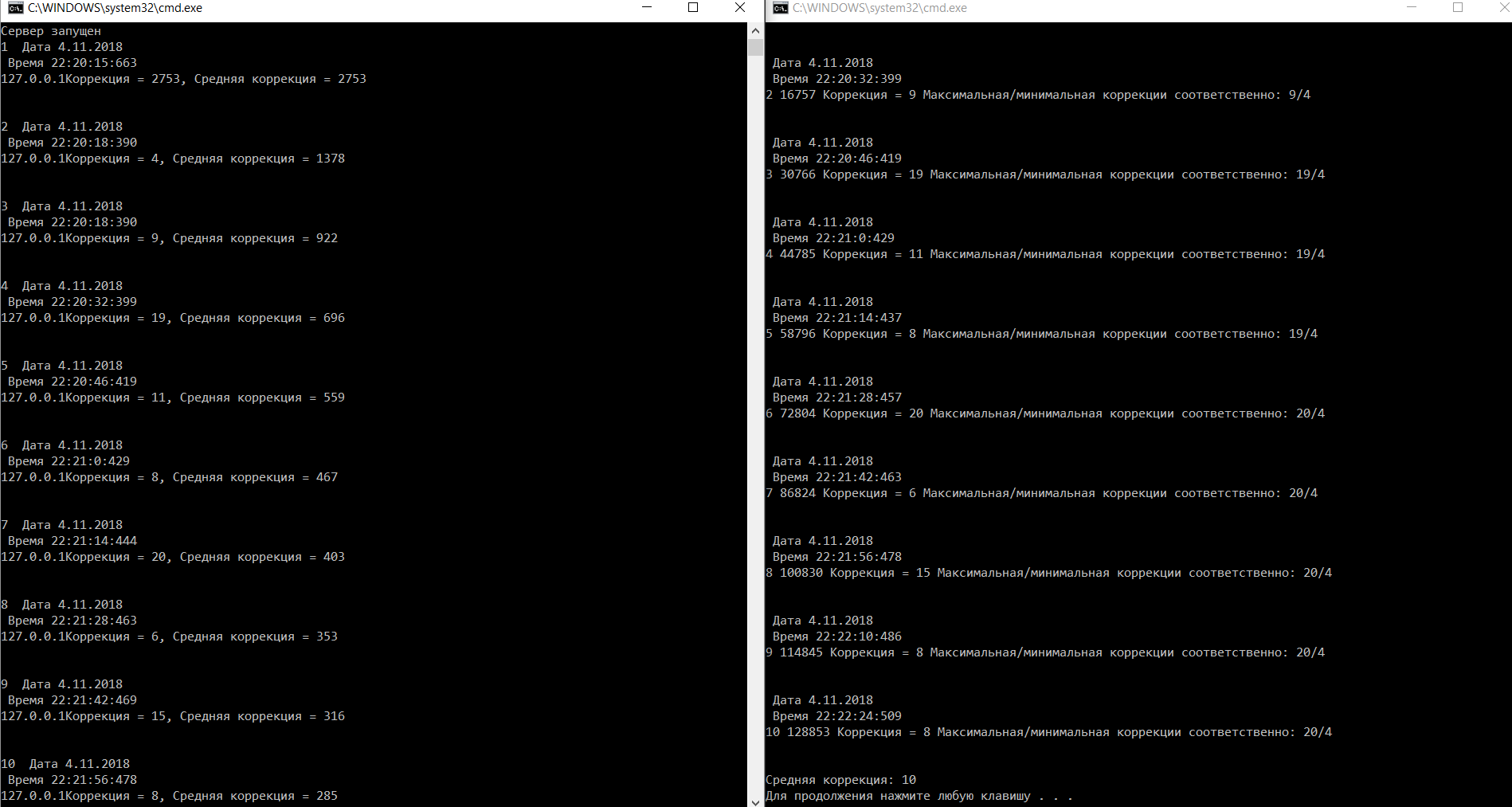
}

Схема работы сервера:

