**МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)**

**Факультет №8 «Прикладной математики и физики»**

**Кафедра вычислительной математики и программирования**

**Курсовой проект**

**по курсу**

**«Операционные системы»**

|  |  |
| --- | --- |
| **Студент:** | Сорокин Д.М. |
| **Группа:** | 8О-207Б, №13 |
| **Преподаватели:** | Миронов Е.С. |
|  |  |
| **Оценка:** |  |
| **Дата:** |  |

2 курс, 1 семестр

Москва 2016

# Цель курсового проекта

1. Приобретение практических навыков в использовании знаний, полученных в течении курса
2. Проведение исследования в выбранной предметной области

# Задание

Необходимо спроектировать и реализовать программный прототип в соответствии с выбранным вариантом. Произвести анализ и сделать вывод на основании данных, полученных при работе программного прототипа.

# Проектирование консольной клиент-серверной игры

На основе любой из выбранных технологий:

* Pipes
* Sockets
* Сервера очередей
* И другие

Создать собственную игру более, чем для одного пользователя. Игра может быть устроена по принципу: клиент-клиент, сервер-клиент.

Возможные варианты игр:

1. Крестики-нолики
2. Морской бой
3. Кости
4. Шашки
5. Охота на лис
6. И другие (**Клавогонки**)

# Текст программы

### Server.cpp

#include <iostream>

#include <string>

#include <vector>

#include <conio.h>

#include <fstream>

#include <ctime>

#include <windows.h>

#include <queue>

using namespace std;

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

//#pragma warning( disable : 4996)

#define PORT 4055

#define MAXCONN 50

int clientsQty = 0;

priority\_queue<int> top;

typedef struct \_result{

double time;

int errors;

int allSyms;

} Result;

DWORD WINAPI WorkWithClient(LPVOID client\_socket);

int numOfFilesInDir()

{

bool x = true;

int i = 0;

WIN32\_FIND\_DATA FindFileData;

HANDLE hFind;

wchar\_t\* fold = L"Texts/\*.txt";

hFind = FindFirstFile(fold, &FindFileData);

if (hFind != INVALID\_HANDLE\_VALUE) {

i++;

while ((x = FindNextFile(hFind, &FindFileData)) == TRUE) {

i++;

}

}

return i;

}

void printCurrState() {

if(clientsQty)

cout << clientsQty << " users online" << endl; \

else

cout << "Nobody is here..." << endl;

}

string getRandomText() {

ifstream file;

int numOfFiles = numOfFilesInDir();

int rdm;

srand(time(NULL));

rdm = rand() % numOfFiles;

char fileName[16];

sprintf\_s(fileName, "Texts/text%i.txt", rdm);

file.open(fileName);

string text = "";

string tmp;

bool t = false;

while (getline(file, tmp))

{

if (t == false) {

text += tmp;

t = true;

}

else

text += '\n' + tmp;

}

return text;

}

int startServer() {

char buff[1024];

cout << "Server start" << endl;

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

{

cout << "WSAStart error " << WSAGetLastError() << endl;

return -1;

}

SOCKET mysocket;

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

{

cout << "Socket() error " << WSAGetLastError() << endl;

WSACleanup();

return -1;

}

sockaddr\_in local\_addr;

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

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

local\_addr.sin\_addr.s\_addr = 0;

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

sizeof(local\_addr)))

{

cout << "Bind error " << WSAGetLastError() << endl;

closesocket(mysocket);

WSACleanup();

return -1;

}

if (listen(mysocket, MAXCONN))

{

cout << "Listen error " << WSAGetLastError() << endl;

closesocket(mysocket);

WSACleanup();

return -1;

}

cout << "Waiting for connection" << endl;

cout << "++++++++++++++++++++++++++++++++" << endl;

SOCKET client\_socket;

sockaddr\_in client\_addr;

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

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

&client\_addr, &client\_addr\_size)))

{

clientsQty++;

HOSTENT \*hst;

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

cout << "++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++" << endl;

cout << "+ " << inet\_ntoa(client\_addr.sin\_addr) << "new player!" << endl;

printCurrState();

cout << "++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++" << endl;

DWORD thID;

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

}

return 0;

}

DWORD WINAPI WorkWithClient(LPVOID client\_socket) {

SOCKET my\_sock;

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

char buff[1024] = "";

while (1)

{

if (!clientsQty)

break;

int bytes\_recv = recv(my\_sock, buff, sizeof(buff) - 1, 0);

if (!strcmp(&buff[0], "text"))

{

string text = getRandomText();

strcpy\_s(buff, text.c\_str());

send(my\_sock, buff, sizeof(buff) - 1, 0);

}

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

clientsQty--;

cout << "- disconnect" << endl;

printCurrState();

cout << "++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++" << endl;

closesocket(my\_sock);

break;

}

if (buff[0] == '$') {

int i = 1;

int speed = 0;

while (buff[i] != '$') {

speed \*= 10;

speed += buff[i] - 48;

++i;

}

top.push(speed);

}

if (!strcmp(&buff[0], "top")) {

string topchik = "";

priority\_queue<int> tmp = top;

for (int i = 1; !tmp.empty() || i >= 10; ++i) {

int sp = tmp.top();

tmp.pop();

topchik += to\_string(i) + " place - " + to\_string(sp) + " sym/min\n";

}

strcpy\_s(buff, topchik.c\_str());

send(my\_sock, buff, sizeof(buff) - 1, 0);

}

}

return 0;

}

int main()

{

startServer();

return 0;

}

## Client.cpp

#include <iostream>

#include <string>

#include <vector>

#include <conio.h>

#include <fstream>

#include <ctime>

#include <windows.h>

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

#pragma warning( disable : 4996)

#define PORT 4055

#define SERVERADDR "127.0.0.1"

using namespace std;

typedef struct \_result {

double time;

int errors;

int allSyms;

} Result;

Result \*checkResult(string text) {

clock\_t time\_t;

char c;

int errors = 0;

cout << "---------------------------T-------E-------X-------T-----------------------------" << endl;

cout << text << endl;

cout << "---------------------------------------------------------------------------------" << endl;

for (size\_t i = 0; i < text.size(); ++i)

{

c = \_getch();

if (i == 0)

time\_t = clock();

if (c == ' ' && text[i] == '\n'

|| c == '\r' && text[i] == '\n'

|| c == '\n' && text[i] == '\n') {

cout << endl;

continue;

}

if (c != text[i]) {

Beep(1000, 200);

++errors;

--i;

}

else

cout << c;

}

time\_t = clock() - time\_t;

cout << endl;

double tme = (double)time\_t / CLOCKS\_PER\_SEC;

Result \*res = new Result;

res->time = tme;

res->errors = errors;

res->allSyms = text.length();

return res;

}

void hints() {

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "'text' to get text" << endl;

cout << "'top' to see fasters" << endl;

cout << "'quit' to end" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

}

int startSession() {

char buff[1024];

cout << "Start client" << endl;

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

{

cout << "WSAStart error " << WSAGetLastError() << endl;

return -1;

}

SOCKET my\_sock;

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

if (my\_sock < 0)

{

cout << "Socket() error " << WSAGetLastError() << endl;

return -1;

}

sockaddr\_in dest\_addr;

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

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

HOSTENT \*hst;

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

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

else

if (hst = gethostbyname(SERVERADDR))

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

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

else

{

cout << "Invalid address " << SERVERADDR << endl;

closesocket(my\_sock);

WSACleanup();

return -1;

}

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

sizeof(dest\_addr)))

{

cout << "Connect error " << WSAGetLastError() << endl;

return -1;

}

cout << "Connection with " << SERVERADDR << " successfully done" << endl;

hints();

while (1)

{

scanf("%s", &buff[0]);

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

{

send(my\_sock, buff, sizeof(buff) - 1, 0);

cout << "Good bye!" << endl;

closesocket(my\_sock);

WSACleanup();

return 0;

}

if (!strcmp(&buff[0], "text")) {

send(my\_sock, buff, sizeof(buff) - 1, 0);

int bytes\_recv = recv(my\_sock, buff, sizeof(buff) - 1, 0);

Result \*result = checkResult(string(buff));

cout << "++++++++++++++++++++++++++++++++" << endl;

cout << "Time: " << result->time << " s" << endl;

cout << "Errors: " << result->errors << endl;

cout << "Speed: " << result->allSyms \* 60.0 / result->time << " sym/min" << endl;

cout << "++++++++++++++++++++++++++++++++" << endl;

int speed = (int) result->allSyms \* 60.0 / result->time;

string rapid = "$" + to\_string(speed) + '$';

strcpy\_s(buff, rapid.c\_str());

send(my\_sock, buff, sizeof(buff) - 1, 0);

delete result;

}

if (!strcmp(&buff[0], "top")) {

send(my\_sock, buff, sizeof(buff) - 1, 0);

int bytes\_recv = recv(my\_sock, buff, sizeof(buff) - 1, 0);

cout << string(buff);

cout << "++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++" << endl;

}

}

}

int main () {

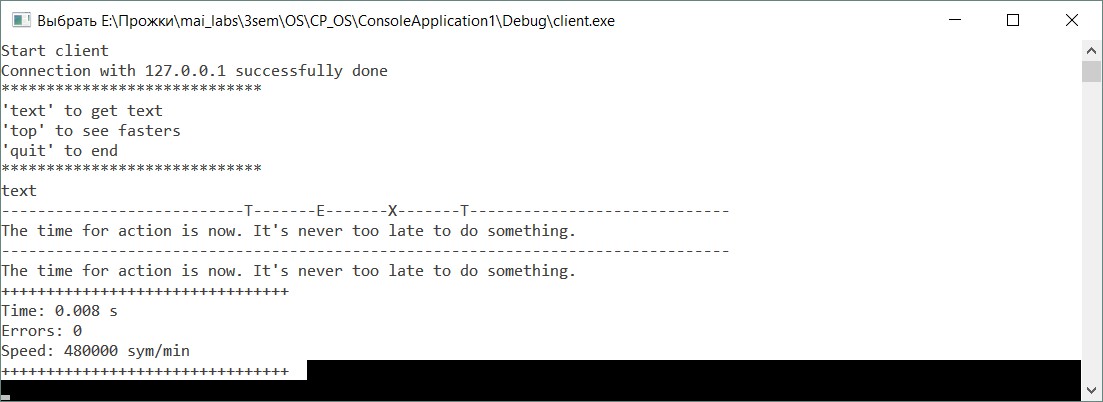
startSession();

return 0;

}

# Пример работы программы



# Выводы

Темой данной курсовой работы я выбрал проектирование клиент-серверной игры «Клавогонки». Суть игры – быстрее всех соперников набрать текст. Данную игру я выбрал не случайно, провел в ней много времени в детстве. Благодаря этому сейчас я быстро стучу по клавишам и освоил слепую печать…

Для взаимодействия клиента с сервером я использовал технологию Socket. Сокеты позволяют процессам обмениваться данным, причем как на одном компьютере, так и на разных, но связанных сетью. С помощью сокетов мои процессы обмениваются командами и реакциями на них. Например, клиент пишет: “***text***”, а сервер в ответ присылает текст для набора.

Эта курсовая ставит точку в изучении курса операционных систем. Он был нелегким, но точно интересным и полезным. Я многому научился в процессе выполнения заданий. Например, писать многопоточные приложения, взаимодействовать между процессами, диагностировать ПО и создавать проекты, подобные данному. Полученный опыт безусловно пригодится в дальнейшей профессиональной деятельности.