Київський національний університет імені Тараса Шевченка

Факультет комп’ютерних наук та кібернетики

Кафедра системного аналізу та теорії прийняття рішень

Звіт

з лабораторної роботи № 1

на тему:

**«Взаємодія розподілених процесів через механізм сокетів»**

Варіант 12

Студента другого курсу

групи К-23(2)

Міщука Романа Андрійовича

Факультету комп’ютерних наук

та кібернетики

Київ – 2022

# Мета

В даній лабораторній роботі необхідно освоїти механізм (технологію) сокетів стеку протоколів TCP/IP, зокрема його реалізацію в MS Windows. Індивідуальний варіант роботи полягає в розробці двох програм (клієнта та сервера, які запускаються на різних станціях мережі), розробці протоколу обміну даними між ними та демонстрації роботи програм.

# Зміст індивідуального завдання

Гра "хрестики-нолики" на полі 5х5 клітинок. Один гравець - на сервері, другий - на клієнті. Поле (матрицю символів) до клієнта передає сервер. Сервер робить хід у будь-яку пусту клітинку, стратегія не потрібна, перемогу/програш не відслідковує. Після кожного ходу клієнта сервер передає поле, додавши свій хід. Клієнт в ході гри може її завершити в будь-який момент.

# Опис протоколу

Будь-яка комунікація між сервером та клієнтом здійснюється у такому вигляді:

|  |  |
| --- | --- |
| Довжина заголовка (в байтах) | Команда1 [; Команда2; ...] |
| 1 байт | до 255 байт |

При цьому, існують такі види команд:

* "CONNECT\_CLIENT" – команда, за допомогою якої здійснюється запит за з’єднання. Перед початком обміну повідомленнями, клієнт та сервер обмінюються цією командою.
* Команда розміром *25 байт* – команда, що містить інформацію про теперішній стан ігрового поля. Має вигляд послідовності символів *0*, *1* та *2*, де 0 – вільна клітинка, 1 – клітинка, зайнята клієнтом, 2 – клітинка, зайнята сервером. Коли хтось робить хід, він змінює значення відповідної комірки, та надсилає оновлений вигляд поля супротивнику.
* Команда розміром *1 байт* – команда, що може бути надіслана лише сервером, та містить у собі інформацію про результат гри. При цьому сервер виступає суддею, і надсилає символ *1* коли переміг клієнт, та *2* коли переміг сервер.
* Команда розміром *26 байт* – команда, що може бути надіслана лише сервером, і містить у собі інформацію про результат гри. Є комбінацією команд на *1 байт* та на *25 байтів*. Надсилається у випадку, коли сервер переміг, і в такому випадку клієнт не знає, як виглядає вирішальний хід.

Закриття з’єднання відбувається за допомогою відповідних функцій shutdown(socket, SD\_SEND); closesocket(socket);, що сповіщає як клієнта, так і сервер.

При надсиланні клієнтом неправильної команди, приймається рішення, що він програв, надсилається відповідна команда, та з’єднання закривається.

# Вигляд протоколу

# Код pch.h

// pch.h: This is a precompiled header file.

// Files listed below are compiled only once, improving build performance for future builds.

// This also affects IntelliSense performance, including code completion and many code browsing features.

// However, files listed here are ALL re-compiled if any one of them is updated between builds.

// Do not add files here that you will be updating frequently as this negates the performance advantage.

#ifndef PCH\_H

#define PCH\_H

// add headers that you want to pre-compile here

#include "framework.h"

#include <winsock2.h>

#include <ws2tcpip.h>

#endif //PCH\_H

# Код WinSockWrapper.h

#pragma once

#include "pch.h"

#include <string>

class WinSockWrapper

{

private:

static bool was\_initialised;

static void init();

public:

static void ensureInit();

static void close();

static SOCKET getSocketForAddress(std::string address, std::string port);

static std::string getIpStringFromAdress(addrinfo\* info);

};

# Код WinSockWrapper.cpp

#include "pch.h"

#include "WinSockWrapper.h"

#include <iostream>

#include <stdexcept>

using namespace std;

// Wrapper -----

bool WinSockWrapper::was\_initialised = false;

void WinSockWrapper::init()

{

WSADATA wsaData;

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

throw runtime\_error("WSAStartup failed.");

}

}

void WinSockWrapper::ensureInit()

{

if(was\_initialised) return;

init();

was\_initialised = true;

}

void WinSockWrapper::close()

{

if (!was\_initialised) return;

WSACleanup();

was\_initialised = false;

}

SOCKET WinSockWrapper::getSocketForAddress(string address, string port)

{

int ret;

addrinfo specs;

ZeroMemory(&specs, sizeof(specs));

specs.ai\_family = AF\_UNSPEC;

specs.ai\_socktype = SOCK\_STREAM;

specs.ai\_protocol = IPPROTO\_TCP;

// RESOLVING THE HOSTNAME/ADDRESS

addrinfo\* results\_list;

if ((ret = getaddrinfo(address.c\_str(), port.c\_str(), &specs, &results\_list)) != 0)

return INVALID\_SOCKET;

SOCKET buff = INVALID\_SOCKET;

for (addrinfo\* i = results\_list; i != NULL; i = i->ai\_next)

{

// SOCKET

buff = socket(i->ai\_family, i->ai\_socktype, i->ai\_protocol);

// CONNECT

if ((ret = connect(buff, i->ai\_addr, (int)i->ai\_addrlen)) != 0)

{

closesocket(buff);

buff = INVALID\_SOCKET;

continue;

}

break;

}

freeaddrinfo(results\_list);

return buff;

}

string WinSockWrapper::getIpStringFromAdress(addrinfo\* info)

{

string ret(INET\_ADDRSTRLEN, 0);

inet\_ntop(info->ai\_family, &(((sockaddr\_in\*)info->ai\_addr)->sin\_addr), (PSTR) ret.c\_str(), ret.size());

return ret;

}

# Код APeer.h

// ReSharper disable CppInconsistentNaming

#pragma once

#include <atomic>

#include <memory>

#include "WinSockWrapper.h"

static constexpr char PORT\_NUMBER[5] = "1037";

static constexpr int MAX\_COMMAND\_SIZE = 255;

static constexpr int RETRY\_NUM = 5;

static constexpr char START\_MESSAGE[15] = "CONNECT\_CLIENT";

static constexpr int START\_MESSAGE\_SIZE = sizeof(START\_MESSAGE);

class APeer // NOLINT(cppcoreguidelines-special-member-functions)

{

public:

struct AContext // NOLINT(cppcoreguidelines-special-member-functions)

{ virtual ~AContext() = 0; } \*\*context;

protected:

std::atomic<bool> working;

SOCKET socket;

struct ASockResult

{

enum Type { OK, SHUTDOWN, ERR } type;

ASockResult(Type type) : type(type) {}

};

struct ErrSockResult : ASockResult

{

int code;

ErrSockResult(int code) : ASockResult(ERR), code(code) {}

};

enum ContactType { SEND, RECEIVE };

std::unique\_ptr<ASockResult> contact(

ContactType type, char\* data, int data\_len) const;

friend class APeerManager;

static std::unique\_ptr<ASockResult> contact(

const SOCKET& socket, ContactType type, char\* data, int data\_len);

virtual std::unique\_ptr<ASockResult> loopStart(std::string& receiveBuffer, std::string& sendBuffer) = 0;

std::unique\_ptr<ASockResult> loopIterBody(std::string& receiveBuffer, std::string& sendBuffer);

virtual std::unique\_ptr<ASockResult> loopEnd(std::unique\_ptr<ASockResult> reason);

public:

APeer();

virtual ~APeer();

bool setSocket(SOCKET socket);

bool setSocket(std::string connectionAddress, std::string connectionPort);

std::unique\_ptr<ASockResult> loop();

struct HandlerResponse

{

std::string message;

enum Flags

{

EMPTY = 0,

MESSAGE = 1 << 0,

SHUTDOWN = 1 << 1

} flags;

HandlerResponse(std::string message);

HandlerResponse(bool is\_shutdown, std::string message);

};

virtual HandlerResponse messageHandler(AContext\*\* context, std::string message) = 0;

};

# Код APeer.cpp

// ReSharper disable CppClangTidyClangDiagnosticCastQual

#include "pch.h"

#include "APeer.h"

#include <iostream>

#include <stdexcept>

using namespace std;

DEFINE\_ENUM\_FLAG\_OPERATORS(APeer::HandlerResponse::Flags)

// https://www.geeksforgeeks.org/pure-virtual-destructor-c/

APeer::AContext::~AContext() = default;

unique\_ptr<APeer::ASockResult> APeer::contact(ContactType type, char\* data, int data\_len) const

{ return contact(socket, type, data, data\_len); }

unique\_ptr<APeer::ASockResult> APeer::contact(const SOCKET& socket, ContactType type, char\* data, int data\_len)

{

int result = 0, code, i = 0;

do

{

if (type == RECEIVE) result = recv(socket, data, data\_len, 0);

else if (type == SEND) result = send(socket, data, data\_len, 0);

code = WSAGetLastError();

#ifdef \_DEBUG

if (type == RECEIVE) cout << "r" << result << '\"' << data << '\"' << "\n";

else if (type == SEND) cout << "s" << result << '\"' << data << '\"' << "\n";

#endif

if (result == data\_len) return make\_unique<ASockResult>(ASockResult::OK);

// https://learn.microsoft.com/uk-ua/windows/win32/winsock/windows-sockets-error-codes-2?redirectedfrom=MSDN

if (result == 0 || result == WSAESHUTDOWN || code == WSAECONNRESET)

return make\_unique<ASockResult>(ASockResult::SHUTDOWN);

i++;

} while (i < RETRY\_NUM);

return make\_unique<ErrSockResult>(code);

}

unique\_ptr<APeer::ASockResult> APeer::loopIterBody(string& receiveBuffer, string& sendBuffer)

{

unique\_ptr<ASockResult> buff\_res;

// Receiving phase

if ((buff\_res = contact(RECEIVE, (char\*) receiveBuffer.c\_str(), 1)

)->type != ASockResult::OK || (unsigned char)receiveBuffer[0] < 1)

return buff\_res;

receiveBuffer.resize((unsigned char) receiveBuffer[0]);

if ((buff\_res = contact(RECEIVE, (char\*) receiveBuffer.c\_str(), (unsigned char) receiveBuffer[0])

)->type != ASockResult::OK)

return buff\_res;

// Processing phase

HandlerResponse response = messageHandler(context, receiveBuffer);

sendBuffer = move(response.message);

if (response.flags & HandlerResponse::SHUTDOWN)

{

// In case, we need to shut down the connection

if (response.flags & HandlerResponse::MESSAGE)

{

sendBuffer = (char) sendBuffer.size() + sendBuffer;

if ((buff\_res = contact(SEND, (char\*) sendBuffer.c\_str(), static\_cast<int>(sendBuffer.size()))

)->type != ASockResult::OK)

return buff\_res;

}

return make\_unique<ASockResult>(ASockResult::SHUTDOWN);

}

// Answering phase

sendBuffer = (char) sendBuffer.size() + sendBuffer;

if ((buff\_res = contact(SEND, (char\*) sendBuffer.c\_str(), static\_cast<int>(sendBuffer.size()))

)->type != ASockResult::OK)

return buff\_res;

return buff\_res;

}

std::unique\_ptr<APeer::ASockResult> APeer::loopEnd(std::unique\_ptr<ASockResult> reason)

{

// Shutting down the connection, since we're done

shutdown(socket, SD\_SEND);

closesocket(socket);

return reason;

}

APeer::APeer(): socket(INVALID\_SOCKET)

{

context = new AContext \* (nullptr);

working = false;

}

APeer::~APeer()

{

delete \*context;

delete context;

}

bool APeer::setSocket(SOCKET socket) // NOLINT(clang-diagnostic-shadow)

{

this->socket = socket;

if(socket == INVALID\_SOCKET) return false;

// https://stackoverflow.com/questions/30395258/setting-timeout-to-recv-function

// setsockopt(socket, SOL\_SOCKET, SO\_RCVTIMEO, (char\*)&TIMEOUT\_MS, sizeof(TIMEOUT\_MS));

return true;

}

bool APeer::setSocket(std::string connectionAddress, std::string connectionPort)

{

WinSockWrapper::ensureInit();

const SOCKET buff = WinSockWrapper::getSocketForAddress(

move(connectionAddress), move(connectionPort));

return setSocket(buff);

}

std::unique\_ptr<APeer::ASockResult> APeer::loop()

{

if(socket == INVALID\_SOCKET)

return make\_unique<ErrSockResult>(INVALID\_SOCKET);

working = true;

unique\_ptr<ASockResult> buffRes;

string receiveBuffer, sendBuffer;

receiveBuffer.reserve(MAX\_COMMAND\_SIZE);

sendBuffer.reserve(MAX\_COMMAND\_SIZE);

receiveBuffer.resize(START\_MESSAGE\_SIZE);

// Starting

if ((buffRes = loopStart(receiveBuffer, sendBuffer)

)->type != ASockResult::OK)

return loopEnd(move(buffRes));

// The Loop

while(working)

{

if((buffRes = loopIterBody(receiveBuffer, sendBuffer)

)->type != ASockResult::OK)

break;

}

// The ending

return loopEnd(move(buffRes));

}

APeer::HandlerResponse::HandlerResponse(std::string message):

message(message), flags(MESSAGE) {}

APeer::HandlerResponse::HandlerResponse(bool is\_shutdown, std::string message): flags(EMPTY)

{

if (!message.empty())

{

this->message = message;

flags |= MESSAGE;

}

if (is\_shutdown) flags |= SHUTDOWN;

}

# Код AServer.h

#pragma once

#include <mutex>

#include <queue>

#include <set>

#include <thread>

#include "APeer.h"

#include "WinSockWrapper.h"

class AServer

{

protected:

std::atomic<bool> working;

SOCKET listenSocket = INVALID\_SOCKET;

class Peer : public APeer

{

private:

AServer\* server;

std::thread\* thread;

std::unique\_ptr<ASockResult> loopEnd(std::unique\_ptr<ASockResult> reason) override;

std::unique\_ptr<ASockResult> loopStart(std::string& receiveBuffer, std::string& sendBuffer) override;

public:

Peer(AServer\* server, SOCKET socket);

virtual ~Peer();

HandlerResponse messageHandler(AContext\*\* context, std::string message) override;

};

std::set<Peer\*> peers;

std::mutex finishedPeersMutex;

std::queue<Peer\*> finishedPeers;

static void resolveAddress(const std::string& address, const std::string& port, addrinfo\*& result);

public:

AServer(std::string address, std::string port);

void loop();

virtual APeer::HandlerResponse messageHandler(APeer::AContext\*\* context, std::string message) = 0;

};

# Код AServer.cpp

// ReSharper disable CppCStyleCast

// ReSharper disable CppClangTidyClangDiagnosticCastQual

#include "pch.h"

#include "AServer.h"

using namespace std;

std::unique\_ptr<APeer::ASockResult> AServer::Peer::loopEnd(std::unique\_ptr<ASockResult> reason)

{

reason = APeer::loopEnd(move(reason));

// Signalling work end

server->finishedPeersMutex.lock();

server->finishedPeers.push(this);

server->finishedPeersMutex.unlock();

return reason;

}

std::unique\_ptr<APeer::ASockResult> AServer::Peer::loopStart(std::string& receiveBuffer, std::string& sendBuffer)

{

unique\_ptr<ASockResult> buffRes;

// Receiving start message from the client

if ((buffRes = contact(RECEIVE, (char\*)receiveBuffer.c\_str(), 1)

)->type != ASockResult::OK || (unsigned char)receiveBuffer[0] != START\_MESSAGE\_SIZE)

{

if (buffRes->type != ASockResult::OK) return buffRes;

return make\_unique<ASockResult>(ASockResult::SHUTDOWN);

}

if ((buffRes = contact(RECEIVE, (char\*)receiveBuffer.c\_str(), START\_MESSAGE\_SIZE)

)->type != ASockResult::OK || strcmp(receiveBuffer.c\_str(), START\_MESSAGE) != 0)

{

if (buffRes->type != ASockResult::OK) return buffRes;

return make\_unique<ASockResult>(ASockResult::SHUTDOWN);

}

// Responding him with the confirmation message

if ((buffRes = contact(SEND, (char\*)&START\_MESSAGE\_SIZE, 1)

)->type != ASockResult::OK)

return buffRes;

if ((buffRes = contact(SEND, (char\*)START\_MESSAGE, START\_MESSAGE\_SIZE)

)->type != ASockResult::OK)

return buffRes;

return buffRes;

}

AServer::Peer::Peer(AServer\* server, SOCKET socket) : // NOLINT(clang-diagnostic-shadow-field)

APeer(), server(server)

{

// https://stackoverflow.com/questions/10998780/stdthread-calling-method-of-class

thread = new std::thread(&Peer::loop, this);

setSocket(socket);

}

AServer::Peer::~Peer()

{

thread->join();

delete thread;

}

APeer::HandlerResponse AServer::Peer::messageHandler(AContext\*\* context, std::string message) // NOLINT(clang-diagnostic-shadow-field)

{

return server->messageHandler(context, move(message));

}

void AServer::resolveAddress(const std::string& address, const std::string& port, addrinfo\*& result)

{

addrinfo specs; // NOLINT(cppcoreguidelines-pro-type-member-init)

ZeroMemory(&specs, sizeof(specs));

specs.ai\_family = AF\_INET;

specs.ai\_socktype = SOCK\_STREAM;

specs.ai\_protocol = IPPROTO\_TCP;

specs.ai\_flags = AI\_PASSIVE;

// https://docs.microsoft.com/en-us/windows/win32/api/ws2tcpip/nf-ws2tcpip-getaddrinfo#:~:text=If%20the-,pNodeName,-parameter%20points%20to%20a%20string%20equal%20to%20%22localhost

const int ret = getaddrinfo(

address.empty() ? NULL : address.c\_str(),

port.c\_str(), &specs, &result

);

}

AServer::AServer(std::string address, std::string port) : working(false)

{

int ret;

WinSockWrapper::ensureInit();

// Resolve the server address and port

addrinfo\* result = nullptr;

resolveAddress(address, port, result);

// Create a SOCKET for the server to listen connections

listenSocket = socket(result->ai\_family, result->ai\_socktype, result->ai\_protocol);

if (listenSocket == INVALID\_SOCKET) {

freeaddrinfo(result);

throw runtime\_error("listenSocket failed with error: " + to\_string(WSAGetLastError()));

}

// Setup the TCP listening listenSocket

ret = bind(listenSocket, result->ai\_addr, (int)result->ai\_addrlen);

freeaddrinfo(result);

if (ret == SOCKET\_ERROR) {

closesocket(listenSocket);

throw runtime\_error("bind failed with error: " + to\_string(ret));

}

}

void AServer::loop()

{

if (listen(listenSocket, SOMAXCONN) == SOCKET\_ERROR) {

closesocket(listenSocket);

throw runtime\_error("listenSocket failed with error: " + to\_string(WSAGetLastError()));

}

working = true;

while (working)

{

const SOCKET buffSocket = accept(listenSocket, nullptr, nullptr);

if (buffSocket == INVALID\_SOCKET) continue;

peers.insert(new Peer(this, buffSocket));

// Cleaning up finished connections

Peer\* buff;

finishedPeersMutex.lock();

while (!finishedPeers.empty())

{

buff = finishedPeers.front();

finishedPeers.pop();

delete buff;

peers.erase(buff);

}

finishedPeersMutex.unlock();

}

closesocket(listenSocket);

}

# Код AClient.h

#pragma once

#include "APeer.h"

#include "WinSockWrapper.h"

class AClient : public APeer

{

protected:

std::unique\_ptr<ASockResult> loopStart(std::string& receiveBuffer, std::string& sendBuffer) override;

public:

AClient(std::string connectionAddress = "localhost", std::string connectionPort = PORT\_NUMBER);

};

# Код AClient.cpp

// ReSharper disable CppCStyleCast

#include "pch.h"

#include "AClient.h"

#include <iostream>

#include <stdexcept>

using namespace std;

std::unique\_ptr<APeer::ASockResult> AClient::loopStart(std::string& receiveBuffer, std::string& sendBuffer)

{

unique\_ptr<ASockResult> buffRes;

// Exchanging START\_MESSAGEs

if ((buffRes = contact(SEND, (char\*) &START\_MESSAGE\_SIZE, 1)

)->type != ASockResult::OK)

return buffRes;

if ((buffRes = contact(SEND, (char\*) START\_MESSAGE, START\_MESSAGE\_SIZE)

)->type != ASockResult::OK)

return buffRes;

return buffRes;

}

AClient::AClient(std::string connectionAddress, std::string connectionPort): APeer()

{ setSocket(move(connectionAddress), move(connectionPort)); }

# Код Сервера

#include <iostream>

#include "AServer.h"

using namespace std;

class Server: public AServer

{

int cur\_id = 0;

public:

Server(std::string address = "", std::string port = PORT\_NUMBER) :

AServer(move(address), move(port)){}

struct Context: APeer::AContext

{

int id;

string prev\_grid;

Context(int id, string grid): AContext(), id(id), prev\_grid(grid) {}

int compare\_grid(const string& grid)

{

if (grid.size() != prev\_grid.size())

return -1;

int ret = 0;

for (int i = 0; i < grid.size(); i++)

if(grid[i] != prev\_grid[i])

{

ret++;

ret += prev\_grid[i] != '0';

}

return ret;

}

virtual ~Context()

{ cout << "Removed finished client!\n"; }

};

int checkWinLoose(const string& grid)

{

char vert[5], horis[5], d1 = grid[0], d2 = grid[4];

for(int i = 0; i < 5; i++)

{

if (grid[i] == '1' || grid[i] == '2') vert[i] = grid[i];

else vert[i] = 0;

if (grid[5\*i] == '1' || grid[5\*i] == '2') horis[i] = grid[5 \*i];

else horis[i] = 0;

}

for(int y = 0; y < 5; y++)

for(int x = 0; x < 5; x++)

{

if (x == y && d1 != grid[x + 5 \* y]) d1 = 0;

if (4 - x == y && d2 != grid[x + 5 \* y]) d2 = 0;

if (grid[x + 5 \* y] != vert[x]) vert[x] = 0;

if (grid[x + 5 \* y] != horis[y]) horis[y] = 0;

}

int was = 0;

if (d1 == '2' || d2 == '2') was = 2;

if (d1 == '1' || d2 == '1') was = 1;

for (int i = 0; i < 5 && was == 0; i++)

{

if (vert[i] == '2' || horis[i] == '2') was = 2;

if (vert[i] == '1' || horis[i] == '1') was = 1;

}

return was;

}

APeer::HandlerResponse messageHandler(APeer::AContext\*\* \_context, std::string message) override

{

if(\*\_context == nullptr)

{

cout << "New client!\n";

if (message.size() != 25) {

return APeer::HandlerResponse(true, "2");

}

\*\_context = new Context(cur\_id++, message);

}

else if(((Context\*)\*\_context)->compare\_grid(message) != 1)

return APeer::HandlerResponse(true, "2");

Context\* context = (Context\*)\*\_context;

int status = checkWinLoose(message);

if(status != 0)

{

if (status == 1) return APeer::HandlerResponse(true, "1");

if (status == 2) return APeer::HandlerResponse(true, "2");

}

int ind = -1;

for (int i = 0; i < 25 && ind == -1; i++)

if (message[i] == '0') ind = i;

if(ind == -1) return APeer::HandlerResponse(true, "0");

message[ind] = '2';

status = checkWinLoose(message);

if (status != 0)

{

if (status == 1) return APeer::HandlerResponse(true, "1" + message);

if (status == 2) return APeer::HandlerResponse(true, "2" + message);

}

context->prev\_grid = message;

return APeer::HandlerResponse(move(message));

}

};

int main()

{

Server server;

server.loop();

WinSockWrapper::close();

return 0;

}

# Код Клієнта

#include <iostream>

#include <string>

#include "AClient.h"

using namespace std;

class Client: public AClient

{

public:

Client(std::string connectionAddress = "localhost", std::string connectionPort = PORT\_NUMBER):

AClient(move(connectionAddress), move(connectionPort)) {}

void print\_grid(const string& grid)

{

cout << "\tGrid:\n\t\_\_\_\_\_\_\_\_\_\_\_\n";

for (int y = 0; y < 5; y++) {

cout << "\t|";

for (int x = 0; x < 5; x++)

{

if (grid[y \* 5 + x] == '1') cout << "@|";

else if (grid[y \* 5 + x] == '2') cout << "#|";

else cout << (char)('a' + y \* 5 + x) << "|";

}

cout << "\n\t-----------\n";

}

}

bool make\_move(string& grid)

{

cout << "Now enter a character, you want to place on a grid\n(that must not be @ or #, or ! to exit): ";

char buff;

cin >> buff;

while(buff != '!' && ('A' > buff || ('Y' < buff && buff < 'a') || buff > 'y'))

{

cout << "Reenter please: ";

cin >> buff;

}

if(buff == '!') return false;

int i;

if (buff < 'Y') i = buff - 'A';

else i = buff - 'a';

grid[i] = '1';

return true;

}

HandlerResponse messageHandler(AContext\*\* context, std::string message) override

{

if(strcmp(message.c\_str(), START\_MESSAGE) == 0)

{

cout << "Successfully connected to the server!\nMake your first move:\n";

string start\_grid(25, '0');

print\_grid(start\_grid);

bool res = make\_move(start\_grid);

if (res) return HandlerResponse(start\_grid);

return HandlerResponse(true, "");

}

if(message.size() == 25)

{

cout << "Received new move from a server. Now its time for yours:\n";

print\_grid(message);

bool res = make\_move(message);

if (res) return HandlerResponse(message);

return HandlerResponse(true, "");

}

if(message.size() == 1 || message.size() == 26)

{

if (message[0] == '2') cout << "!!! Server won\n";

else if (message[0] == '1') cout << "!!! You won!\n";

else if (message[0] == '0') cout << "!!! No one win(!\n";

if(message.size() == 26)

{

cout << "It responded with such a grid:\n";

print\_grid(message.substr(1));

}

return HandlerResponse(true, "");

}

return HandlerResponse(true, "");

}

std::unique\_ptr<ASockResult> loopEnd(std::unique\_ptr<ASockResult> reason) override

{

if (reason->type == ASockResult::SHUTDOWN)

cout << "Connection was shutdown\n";

else if (reason->type == ASockResult::ERR)

cout << "Connection closed because of an error: " << ((ErrSockResult\*)reason.get())->code;

return AClient::loopEnd(move(reason));

}

};

int main()

{

Client cl;

cl.loop();

WinSockWrapper::close();

cin.get(); cin.get();

}

# Текст системного журналу Сервера

r1""

r15"CONNECT\_CLIENT"

s1""

s15"CONNECT\_CLIENT"

r1"↓ONNECT\_CLIENT"

r25"1000000000000000000000000"

New client!

s26"↓1200000000000000000000000"

r1"↓000000000000000000000000"

r25"1200001000000000000000000"

s26"↓1220001000000000000000000"

r1"↓200001000000000000000000"

r25"1220001000001000000000000"

s26"↓1222001000001000000000000"

r1"↓220001000001000000000000"

r25"1222001000001000001000000"

s26"↓1222201000001000001000000"

r1"↓222001000001000001000000"

r25"1222201000001000001000001"

s2"☺1"

Removed finished client!

# Текст системного журналу Клієнта

s1""

s15"CONNECT\_CLIENT"

r1""

r15"CONNECT\_CLIENT"

Successfully connected to the server!

Make your first move:

Grid:

\_\_\_\_\_\_\_\_\_\_\_

|a|b|c|d|e|

-----------

|f|g|h|i|j|

-----------

|k|l|m|n|o|

-----------

|p|q|r|s|t|

-----------

|u|v|w|x|y|

-----------

Now enter a character, you want to place on a grid

(that must not be @ or #, or ! to exit): a

s26"↓1000000000000000000000000"

r1"↓ONNECT\_CLIENT"

r25"1200000000000000000000000"

Received new move from a server. Now its time for yours:

Grid:

\_\_\_\_\_\_\_\_\_\_\_

|@|#|c|d|e|

-----------

|f|g|h|i|j|

-----------

|k|l|m|n|o|

-----------

|p|q|r|s|t|

-----------

|u|v|w|x|y|

-----------

Now enter a character, you want to place on a grid

(that must not be @ or #, or ! to exit): g

s26"↓1200001000000000000000000"

r1"↓200000000000000000000000"

r25"1220001000000000000000000"

Received new move from a server. Now its time for yours:

Grid:

\_\_\_\_\_\_\_\_\_\_\_

|@|#|#|d|e|

-----------

|f|@|h|i|j|

-----------

|k|l|m|n|o|

-----------

|p|q|r|s|t|

-----------

|u|v|w|x|y|

-----------

Now enter a character, you want to place on a grid

(that must not be @ or #, or ! to exit): m

s26"↓1220001000001000000000000"

r1"↓220001000000000000000000"

r25"1222001000001000000000000"

Received new move from a server. Now its time for yours:

Grid:

\_\_\_\_\_\_\_\_\_\_\_

|@|#|#|#|e|

-----------

|f|@|h|i|j|

-----------

|k|l|@|n|o|

-----------

|p|q|r|s|t|

-----------

|u|v|w|x|y|

-----------

Now enter a character, you want to place on a grid

(that must not be @ or #, or ! to exit): s

s26"↓1222001000001000001000000"

r1"↓222001000001000000000000"

r25"1222201000001000001000000"

Received new move from a server. Now its time for yours:

Grid:

\_\_\_\_\_\_\_\_\_\_\_

|@|#|#|#|#|

-----------

|f|@|h|i|j|

-----------

|k|l|@|n|o|

-----------

|p|q|r|@|t|

-----------

|u|v|w|x|y|

-----------

Now enter a character, you want to place on a grid

(that must not be @ or #, or ! to exit): y

s26"↓1222201000001000001000001"

r1"☺222201000001000001000000"

r1"1"

!!! You won!

Connection was shutdown