Приложение Ж Листинг исходного кода

CMakeLists.txt

cmake\_minimum\_required(VERSION 3.0.0)

project(one-way-sync VERSION 0.1.0)

set(CMAKE\_CXX\_STANDARD 20)

set (source\_dir "${PROJECT\_SOURCE\_DIR}/src/")

SET(CMAKE\_INCLUDE\_PATH ${CMAKE\_INCLUDE\_PATH} "C:/CLI\_STUFF/mingw64/boost\_1\_78\_0/")

SET(CMAKE\_LIBRARY\_PATH ${CMAKE\_LIBRARY\_PATH} "C:/CLI\_STUFF/mingw64/boost\_1\_78\_0/lib/")

set(BOOST\_ROOT "C:/CLI\_STUFF/mingw64/boost\_1\_78\_0/")

set(Boost\_USE\_STATIC\_LIBS ON)

find\_package(Boost COMPONENTS system)

find\_package( Threads )

include\_directories(${Boost\_INCLUDE\_DIR})

file (GLOB source\_files

"${source\_dir}/\*.cpp"

"${source\_dir}/helpers/\*.cpp"

"${source\_dir}/models/\*.cpp"

"${source\_dir}/networking/\*.cpp"

)

include\_directories(

"${source\_dir}/"

)

add\_compile\_options(-Wall -Wextra -pedantic -Werror -pthread)

add\_executable(one-way-sync ${source\_files})

target\_link\_libraries(one-way-sync ws2\_32)

target\_link\_libraries(one-way-sync Threads::Threads)

main.cpp

#include <boost/asio.hpp>

#include <boost/array.hpp>

#include <boost/bind/bind.hpp>

#include <thread>

#include <iostream>

#define IPADDRESS "127.0.0.1" // "192.168.1.64"

#define UDP\_PORT 13253

using boost::asio::ip::udp;

using boost::asio::ip::address;

void Sender(std::string in) {

boost::asio::io\_service io\_service;

udp::socket socket(io\_service);

udp::endpoint remote\_endpoint = udp::endpoint(address::from\_string(IPADDRESS), UDP\_PORT);

socket.open(udp::v4());

boost::system::error\_code err;

auto sent = socket.send\_to(boost::asio::buffer(in), remote\_endpoint, 0, err);

socket.close();

std::cout << "Sent Payload --- " << sent << "\n";

}

struct Client {

boost::asio::io\_service io\_service;

udp::socket socket{io\_service};

boost::array<char, 1024> recv\_buffer;

udp::endpoint remote\_endpoint;

int count = 3;

void handle\_receive(const boost::system::error\_code& error, size\_t bytes\_transferred) {

if (error) {

std::cout << "Receive failed: " << error.message() << "\n";

return;

}

std::cout << "Received: '" << std::string(recv\_buffer.begin(), recv\_buffer.begin()+bytes\_transferred) << "' (" << error.message() << ")\n";

if (--count > 0) {

std::cout << "Count: " << count << "\n";

wait();

}

}

void wait() {

socket.async\_receive\_from(boost::asio::buffer(recv\_buffer),

remote\_endpoint,

boost::bind(&Client::handle\_receive, this, boost::asio::placeholders::error, boost::asio::placeholders::bytes\_transferred));

}

void Receiver()

{

socket.open(udp::v4());

socket.bind(udp::endpoint(address::from\_string(IPADDRESS), UDP\_PORT));

wait();

std::cout << "Receiving\n";

io\_service.run();

std::cout << "Receiver exit\n";

}

};

int main(int argc, char \*argv[])

{

Client client;

std::thread r([&] { client.Receiver(); });

std::string input = argc>1? argv[1] : "hello world";

std::cout << "Input is '" << input.c\_str() << "'\nSending it to Sender Function...\n";

for (int i = 0; i < 3; ++i) {

std::this\_thread::sleep\_for(std::chrono::milliseconds(200));

Sender(input);

}

r.join();

}