TCP Server Client Chat Using Boost:

Client:

#include<iostream>

#include<string>

#include<boost/asio.hpp>

#include<boost/array.hpp>

using boost::asio::ip::tcp;

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

{

try

{

if(argc != 2)

{

std::cerr << "Usage: client localhost" << std::endl;

return 1;

}

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

tcp::resolver resolver(io\_service);

tcp::resolver::query query(argv[1], "daytime");

tcp::resolver::iterator endpoint\_iterator = resolver.resolve(query);

tcp::socket socket(io\_service);

boost::asio::connect(socket, endpoint\_iterator);

for(;;)

{

boost::array<char, 128>buf;

boost::system::error\_code error;

size\_t len = socket.read\_some(boost::asio::buffer(buf), error);

if (buf.data()=="Bye")

break;

else if (error)

throw boost::system::system\_error(error);

std::cout<<"\nServer: ";

std::cout.write(buf.data(), len);

boost::system::error\_code ignored\_error;

std::cout<<"\nClient: ";

std::string mess;

std::cin>>mess;

boost::asio::write(socket, boost::asio::buffer(mess), ignored\_error);

}

}

catch(std::exception& e)

{

std::cerr << e.what() << std::endl;

}

return 0;

}

Server:

#include<iostream>

#include<ctime>

#include<string>

#include<boost/asio.hpp>

#include<boost/array.hpp>

using boost::asio::ip::tcp;

std::string make\_daytime\_string()

{

using namespace std;

time\_t now = time(0);

return ctime(&now);

}

int main()

{

try

{

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

tcp::acceptor acceptor(io\_service, tcp::endpoint(tcp::v4(), 13));

tcp::socket socket(io\_service);

acceptor.accept(socket);

for(;;)

{

std::string message;

std::cout<<"\nServer: ";

std::cin>>message;

boost::system::error\_code ignored\_error;

boost::asio::write(socket, boost::asio::buffer(message), ignored\_error);

boost::array<char, 128>buf;

boost::system::error\_code error;

size\_t len = socket.read\_some(boost::asio::buffer(buf), error);

if (error == boost::asio::error::eof)

break;

else if (error)

throw::boost::system::system\_error(error);

std::cout<<"\nClient: ";

std::cout.write(buf.data(), len);

}

}

catch (std::exception& e)

{

std::cerr << e.what() << std::endl;

}

return 0;

}