#include <iostream>

#include <WS2tcpip.h>

#include <string>

#include <fstream>

#include <time.h>

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

using namespace std;

void swearwordfilter(string& array, int size, string nextbadword) {

int Bwordpointer = 0, Bwordpos = 0;

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

// tolower converts the input to lowercase, as badwords in file are lowercase

if (tolower(array.at(i)) == nextbadword[Bwordpointer]) {

if (Bwordpointer == 0) {

Bwordpos = i;

}

Bwordpointer++;

}

else

Bwordpointer = 0;

if (Bwordpointer == nextbadword.length()) {

for (int i = Bwordpos; i < Bwordpos + nextbadword.length(); i++) {

array.at(i) = '\*';

}

Bwordpointer = 0;

}

}

}

void main()

{

// Initialze winsock

WSADATA wsData;

WORD ver = MAKEWORD(2, 2);

int wsOk = WSAStartup(ver, &wsData);

if (wsOk != 0)

{

cerr << "Can't Initialize winsock! Quitting" << endl;

return;

}

// Create a socket

SOCKET listening = socket(AF\_INET, SOCK\_STREAM, 0);

if (listening == INVALID\_SOCKET)

{

cerr << "Can't create a socket! Quitting" << endl;

return;

}

// Bind the ip address and port to a socket

sockaddr\_in hint;

hint.sin\_family = AF\_INET;

hint.sin\_port = htons(54000);

hint.sin\_addr.S\_un.S\_addr = INADDR\_ANY; // Could also use inet\_pton ....

bind(listening, (sockaddr\*)&hint, sizeof(hint));

// Tell Winsock the socket is for listening

listen(listening, SOMAXCONN);

// Wait for a connection

sockaddr\_in client;

int clientSize = sizeof(client);

SOCKET clientSocket = accept(listening, (sockaddr\*)&client, &clientSize);

char host[NI\_MAXHOST]; // Client's remote name

char service[NI\_MAXSERV]; // Service (i.e. port) the client is connect on

ZeroMemory(host, NI\_MAXHOST); // same as memset(host, 0, NI\_MAXHOST);

ZeroMemory(service, NI\_MAXSERV);

if (getnameinfo((sockaddr\*)&client, sizeof(client), host, NI\_MAXHOST, service, NI\_MAXSERV, 0) == 0)

{

cout << host << " connected on port " << service << endl;

}

else

{

inet\_ntop(AF\_INET, &client.sin\_addr, host, NI\_MAXHOST);

cout << host << " connected on port " <<

ntohs(client.sin\_port) << endl;

}

// Close listening socket

closesocket(listening);

// While loop: accept and echo message back to client

char buf[4096];

//File handling part

ofstream serverfile;

serverfile.open("ServerChatlog.txt", std::ios\_base::app);

string align; //aligning

for (int i = 0; i < 30; i++) align += '-';

serverfile << align << "New Chat log" << align << endl;

// Loop begin

while (true)

{

string Uinput;

ZeroMemory(buf, 4096);

// Time

time\_t result = time(NULL);

char myTime[26];

ctime\_s(myTime, sizeof myTime, &result);

// Wait for client to send data

int bytesReceived = recv(clientSocket, buf, 4096, 0);

serverfile << buf << endl << " Sent on: " << myTime << endl;

int Frecv = 0;

if (string(buf, 0, bytesReceived).substr(0, 5) == "$file") {

ofstream file;

file.open("transferedfile.txt", std::ios\_base::out);

while (string(buf, 0, bytesReceived) != "end") {

int bytesReceived = recv(clientSocket, buf, 4096, 0);

if (string(buf, 0, bytesReceived) != "end") send(clientSocket, "RECV", 5, 0);

file << string(buf, 0, bytesReceived)<<endl;

}

Frecv = 1;

send(clientSocket, "FILE RECV", 10, 0);

}

if (Frecv == 1) continue; // Restarts the loop once File is recieved

if (bytesReceived == SOCKET\_ERROR)

{

cerr << "Error in recv(). Quitting" << endl;

break;

}

if (bytesReceived == 0)

{

cout << "Client disconnected " << endl;

break;

}

cout << string(buf, 0, bytesReceived) << endl;

cout << "> ";

getline(cin, Uinput);

ifstream pFile; // from here

pFile.open("filteredwords.txt", std::ios\_base::in);

string buffer;

while (getline(pFile, buffer)) {

swearwordfilter(Uinput, Uinput.size(), buffer);

}

// Time for server

ZeroMemory(myTime, 26);

time\_t result1 = time(NULL);

ctime\_s(myTime, sizeof myTime, &result1);

serverfile << "SERVER >" << Uinput << endl << " sent on: " << myTime << endl;

// Echo message back to client

send(clientSocket, Uinput.c\_str(), Uinput.size(), 0);

}

// Close the socket

closesocket(clientSocket);

// Cleanup winsock

WSACleanup();

system("pause");

}