Ôn tập LTM

* 13/3

//Khoi tao 1 Wínsock

#include <iostream>

#include <WinSock2.h>

using namespace std;

using namespace std;

int main()

{

WSADATA data;

WORD version = MAKEWORD(2, 2);

int r = WSAStartup(version, &data); //LPWSADATA = \*WSADATA

if (r == 0) {

printf("Khoi tao thanh cong!\n");

}

else {

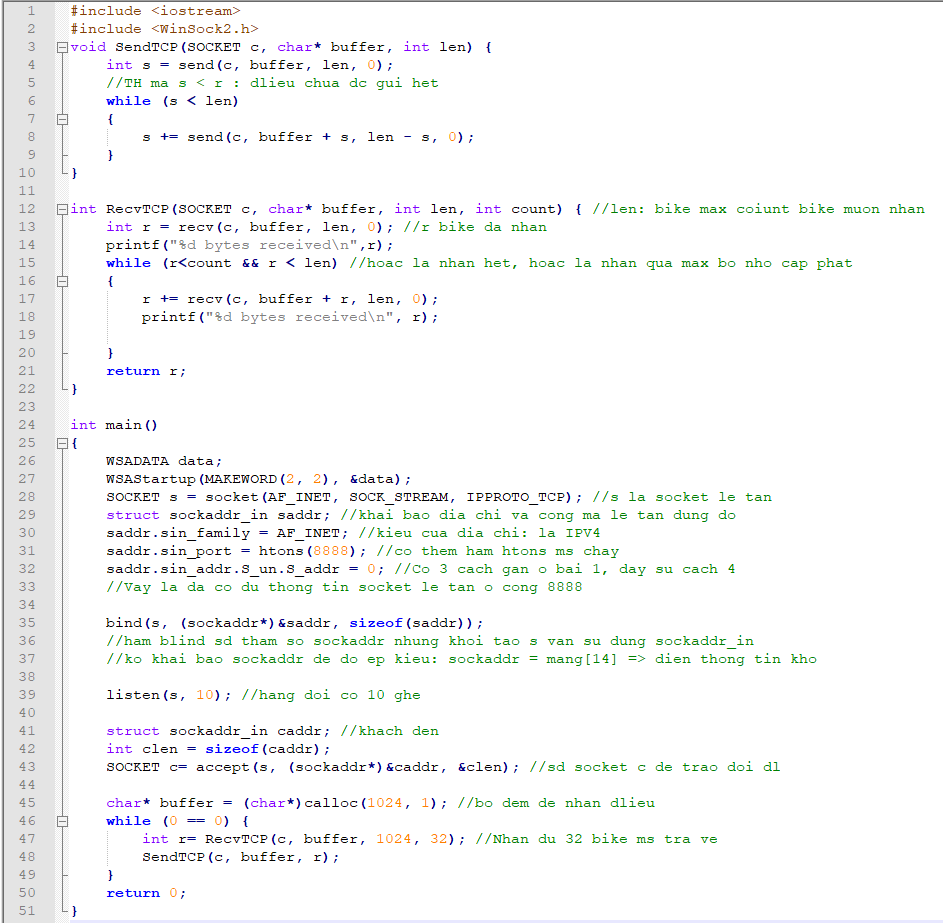
printf("Khoi tao that bai!\n");

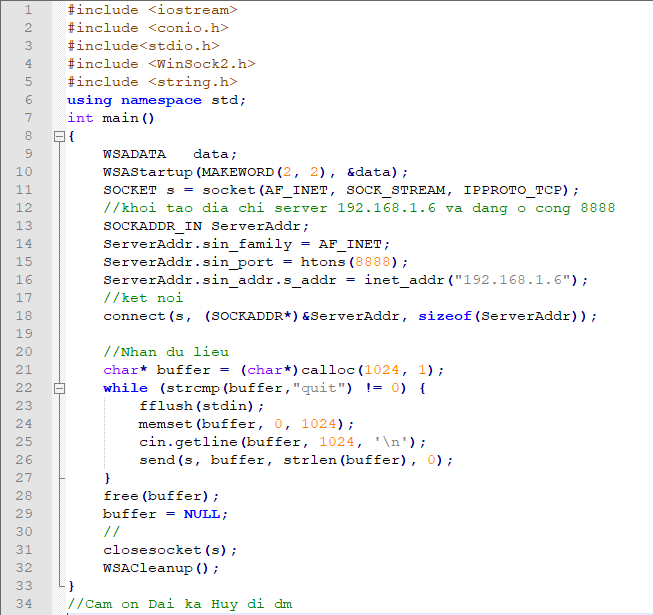
}

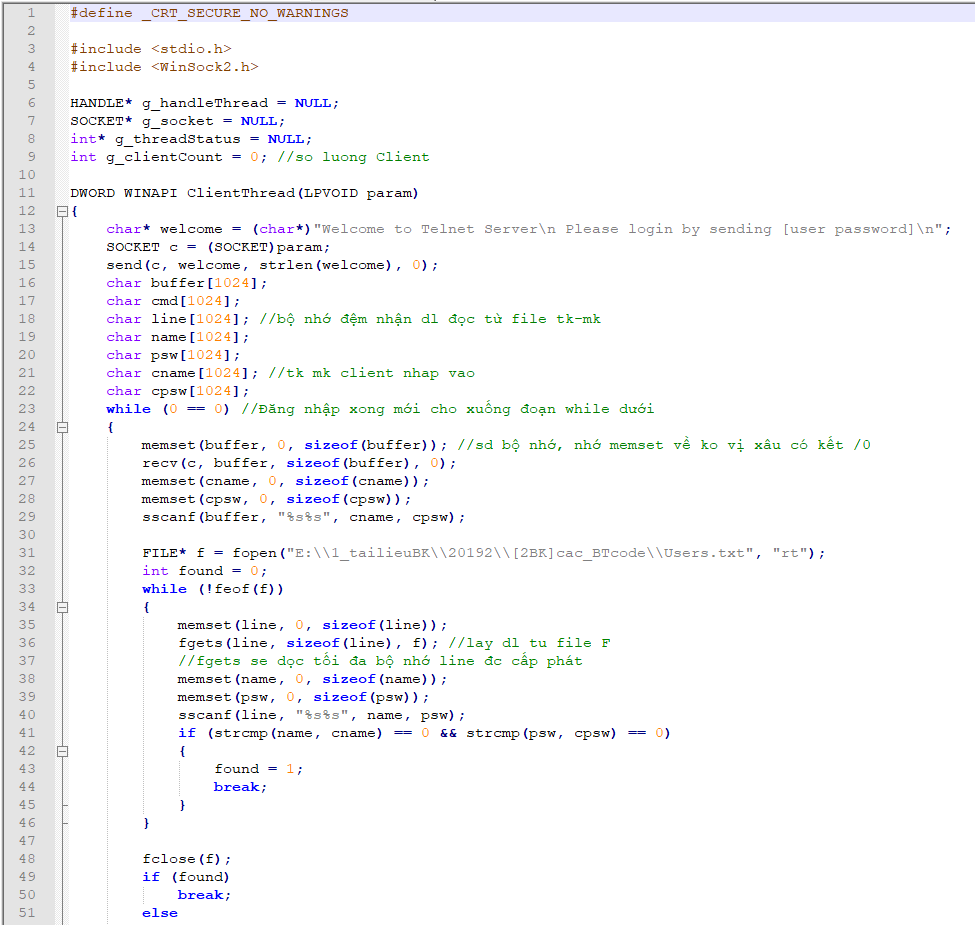
WSACleanup();

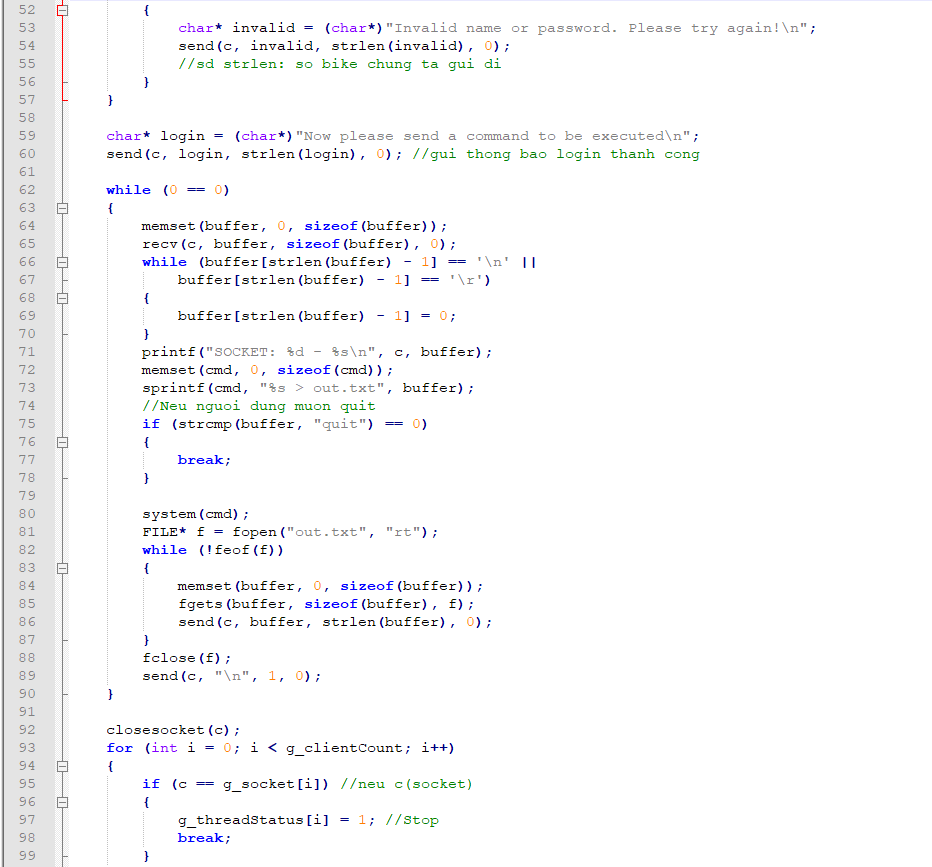
}

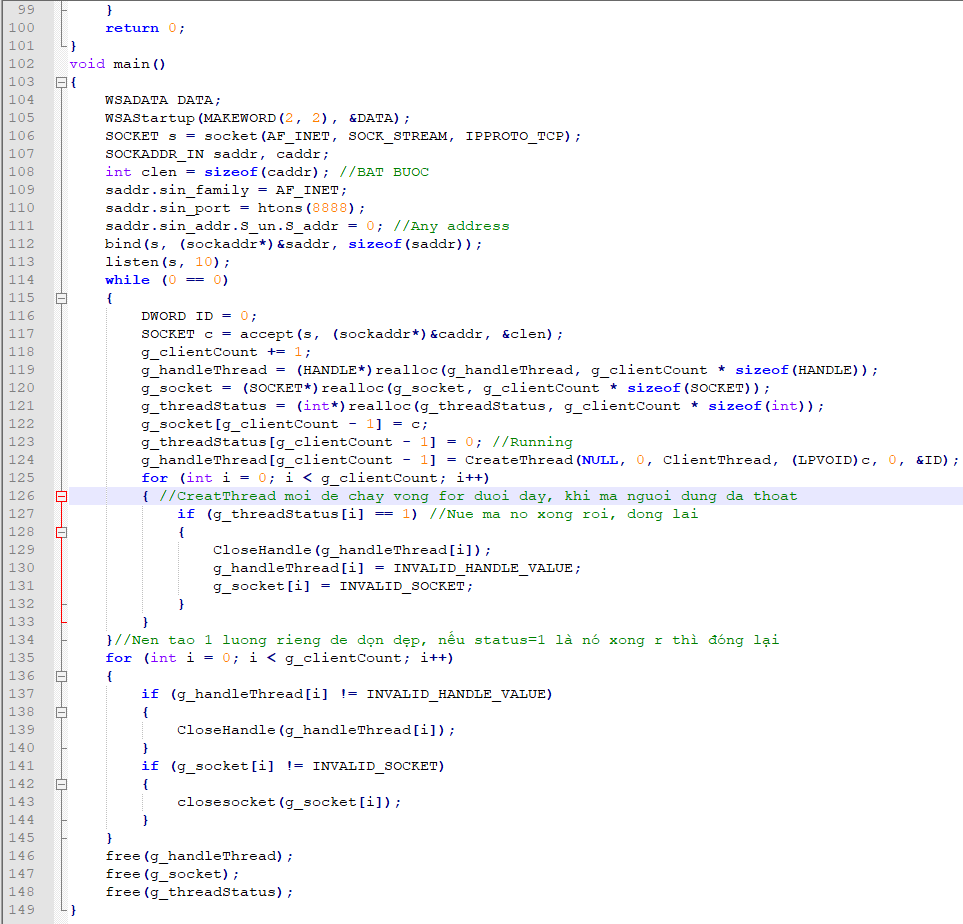
* 19/3: TCP Socket
* Server



* Client
* Telnet Server – 2/4







* http server

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <WinSock2.h>

char\* g\_html = NULL;

void SendFile(SOCKET c, char\* path)

{

    char header[1024];

    memset(header, 0, 1024);

    strcpy(header, "HTTP/1.1 200 OK\n");

    char\* pattern = strstr(path, "VINHLT");

    pattern[0] = 0;

    if (\_stricmp(path + strlen(path) - 4, ".jpg") == 0)

    {

        sprintf(header + strlen(header), "Content-Type: image/jpeg\n");

    }

    else if (\_stricmp(path + strlen(path) - 4, ".mp4") == 0)

    {

        sprintf(header + strlen(header), "Content-Type: video/mp4\n");

    }

    else if (\_stricmp(path + strlen(path) - 4, ".mp3") == 0)

    {

        sprintf(header + strlen(header), "Content-Type: audio/mp3\n");

    }

    else

    {

        sprintf(header + strlen(header), "Content-Type: application/octet-stream\n");

    }

    char buffer[1024];

    FILE\* f = fopen(path, "rb");

    fseek(f, 0, SEEK\_END);

    int flen = ftell(f);

    fseek(f, 0, SEEK\_SET);

    sprintf(header + strlen(header), "Content-Length: %d\n\n", flen);

    send(c, header, strlen(header), 0);

    while (!feof(f))

    {

        int r = fread(buffer, 1, sizeof(buffer), f);

        send(c, buffer, r, 0);

    }

    fclose(f);

}

void ScanFolder(char\* path)

{

    WIN32\_FIND\_DATAA DATA;

    char fullpath[1024];

    memset(fullpath, 0, sizeof(fullpath));

    if (path[strlen(path) - 1] == '/')

        sprintf(fullpath, "C:%s\*.\*", path);

    else

        sprintf(fullpath, "C:%s/\*.\*", path);

    HANDLE hFind = FindFirstFileA(fullpath, &DATA);

    char tmpHtml[1024];

    memset(tmpHtml, 0, sizeof(tmpHtml));

    if (DATA.dwFileAttributes & FILE\_ATTRIBUTE\_DIRECTORY)

    {

        if (path[strlen(path) - 1] == '/')

        {

            sprintf(tmpHtml, "<a href=\"%s%s\"><b>%s</b></a><br>", path, DATA.cFileName, DATA.cFileName);

        }

        else

        {

            sprintf(tmpHtml, "<a href=\"%s/%s\"><b>%s</b></a><br>", path, DATA.cFileName, DATA.cFileName);

        }

    }

    else

    {

        if (path[strlen(path) - 1] == '/')

            sprintf(tmpHtml, "<a href=\"%s%sVINHLT\">%s</a><br>", path, DATA.cFileName, DATA.cFileName);

        else

            sprintf(tmpHtml, "<a href=\"%s/%sVINHLT\">%s</a><br>", path, DATA.cFileName, DATA.cFileName);

    }

    int oldlen = g\_html != NULL ? strlen(g\_html) : 0;

    g\_html = (char\*)realloc(g\_html, oldlen + strlen(tmpHtml) + 1);

    sprintf(g\_html + oldlen, "%s", tmpHtml);

    if (hFind != INVALID\_HANDLE\_VALUE)

    {

        while (FindNextFileA(hFind, &DATA) == TRUE)

        {

            memset(tmpHtml, 0, sizeof(tmpHtml));

            if (DATA.dwFileAttributes & FILE\_ATTRIBUTE\_DIRECTORY)

            {

                if (path[strlen(path) - 1] == '/')

                    sprintf(tmpHtml, "<a href=\"%s%s\"><b>%s</b></a><br>", path, DATA.cFileName, DATA.cFileName);

                else

                    sprintf(tmpHtml, "<a href=\"%s/%s\"><b>%s</b></a><br>", path, DATA.cFileName, DATA.cFileName);

            }

            else

            {

                if (path[strlen(path) - 1] == '/')

                    sprintf(tmpHtml, "<a href=\"%s%sVINHLT\">%s</a><br>", path, DATA.cFileName, DATA.cFileName);

                else

                    sprintf(tmpHtml, "<a href=\"%s/%sVINHLT\">%s</a><br>", path, DATA.cFileName, DATA.cFileName);

            }

            int oldlen = g\_html != NULL ? strlen(g\_html) : 0;

            g\_html = (char\*)realloc(g\_html, oldlen + strlen(tmpHtml) + 1);

            sprintf(g\_html + oldlen, "%s", tmpHtml);

        }

    }

}

int FindBoundaryPattern(char\* pattern, char\* data, int len, int from)

//Tim mau pattern trong du lieu data bat dau tu vi tri from, len la tong so byte cua data

{

    for (int i = from; i < len; i++)

    {

        if (pattern[0] == data[i])

        {

            int j = 0;

            for (j = 0; j < strlen(pattern); j++)

            {

                if (pattern[j] != data[j + i])

                {

                    break;

                }

            }

            if (j == strlen(pattern))

            {

                return i;

            }

        }

    }

    return -1;

}

DWORD WINAPI ClientThread(LPVOID param)

{

    SOCKET c = (SOCKET)param;

    char\* buffer = (char\*)calloc(1024, 1); //Cap phat dong 1024 byte

    memset(buffer, 0, sizeof(buffer));

    int r = recv(c, buffer, 1024, 0);

    char ACTION[1024];

    char PATH[1024];

    char VER[1024];

    char BOUNDARY[1024];

    char LENGTH[1024];

    memset(ACTION, 0, sizeof(ACTION));

    memset(PATH, 0, sizeof(PATH));

    memset(VER, 0, sizeof(VER));

    memset(BOUNDARY, 0, sizeof(BOUNDARY));

    memset(LENGTH, 0, sizeof(LENGTH));

    sscanf(buffer, "%s%s%s", ACTION, PATH, VER);

    if (g\_html != NULL)

    {

        free(g\_html);

        g\_html = NULL;

    }

    if (strcmp(ACTION, "POST") == 0)

    {

        FILE\* f = NULL;

        char\* tmp = strstr(buffer, "boundary=") + strlen("boundary=");

        sscanf(tmp, "%s", BOUNDARY);

        tmp = strstr(buffer, "Content-Length:") + strlen("Content-Length:");

        sscanf(tmp, "%s", LENGTH);

        int length = atoi(LENGTH);

        buffer = (char\*)realloc(buffer, 1024 + length); //Chac chan du boi vi 1024 byte thua de chua header

        int r1 = 0;

        int received = r;

        do

        {

            r1 = recv(c, buffer + received, 1024, 0);

            if (r1 > 0)

            {

                received += r1;

            }

            else

                break;

        } while (received < length);

        length = received; //Tong so byte nhan trong ca hai lan goi recv

        int found = FindBoundaryPattern(BOUNDARY, buffer, length, 0); //Boundary o tren phan HEADER cua HTTP

        while (found)

        {

            found = FindBoundaryPattern(BOUNDARY, buffer, length, found + 1); //Boundary bat dau phan du lieu

            if (found >= 0)

            {

                //Tach ten file tu metadata sau boundary

                char\* filename = strstr(buffer + found, "filename=\"");

                if (filename != NULL)

                {

                    char fnamevalue[1024];

                    memset(fnamevalue, 0, sizeof(fnamevalue));

                    strcpy(fnamevalue, "C:\\Temp\\");

                    sscanf(filename + strlen("filename=\""), "%s", fnamevalue + strlen(fnamevalue));

                    fnamevalue[strlen(fnamevalue) - 1] = 0;

                    f = fopen(fnamevalue, "wb");

                }

                char\* linebreak = strstr(buffer + found, "\r\n\r\n");

                if (linebreak != NULL)

                    found = linebreak - buffer + 4; //Bo qua doan du lieu bao gom 4 dau \r\n\r\n

                else

                    found += strlen(BOUNDARY);

                int next = FindBoundaryPattern(BOUNDARY, buffer, length, found + 1); //Vi tri tiep theo

                if (next >= 0)

                {

                    fwrite(buffer + found, 1, next - found, f);

                }

                else

                {

                    next = length - 1;

                    fwrite(buffer + found, 1, next - found, f);

                    break;

                }

            }

        }

        free(buffer); //Neu dung calloc/malloc/realloc thi phai free de thu hoi bo nho

        fclose(f);

        char\* response = (char\*)"HTTP/1.1 200 OK\n\n";

        send(c, response, strlen(response), 0);

    }

    if (strcmp(ACTION, "GET") == 0)

    {

        while (strstr(PATH, "%20") != NULL)

        {

            char\* found = strstr(PATH, "%20");

            found[0] = ' ';

            strcpy(found + 1, found + 3);

        }

        if (strstr(PATH, "VINHLT") != NULL)

        {

            SendFile(c, PATH);

        }

        else

        {

            char\* header = (char\*)"HTTP/1.1 200 OK\nServer: LTM\nContent-Type: text/html\n\n";

            send(c, header, strlen(header), 0);

            char\* input = (char\*)"<form action=\"/\" method=\"post\" enctype=\"multipart/form-data\"><input type=\"file\" id=\"txtmyid\" name=\"txtmyname\"><br><input type=\"submit\"></form><br>";

            send(c, input, strlen(input), 0);

            ScanFolder(PATH);

            g\_html = (char\*)realloc(g\_html, strlen(g\_html) + 8);

            sprintf(g\_html + strlen(g\_html), "</html>");

            send(c, g\_html, strlen(g\_html), 0);

        }

    }

    closesocket(c);

    return 0;

}

void main()

{

    WSADATA data;

    WSAStartup(MAKEWORD(2, 2), &data);

    SOCKET s = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

    struct sockaddr\_in saddr;

    saddr.sin\_family = AF\_INET;

    saddr.sin\_port = htons(8888);

    saddr.sin\_addr.S\_un.S\_addr = 0; //htons(INADDR\_ANY)

    bind(s, (sockaddr\*)&saddr, sizeof(saddr));

    listen(s, 10);

    while (0 == 0)

    {

        DWORD ID = 0;

        SOCKADDR\_IN caddr;

        int clen = sizeof(caddr);

        SOCKET c = accept(s, (sockaddr\*)&caddr, &clen);

        CreateThread(NULL, 0, ClientThread, (LPVOID)c, 0, &ID);

    }

}

* 24/4

#include <stdio.h>

#include <WinSock2.h>

#define MAX\_SOCKET\_PER\_THREAD 3

//Mang de chua socket cua cac client dang ket noi den server

SOCKET g\_client[1024];

int g\_count = 0;

DWORD WINAPI ClientThread(LPVOID param)

{

    fd\_set read;

    int startIndex = (int)param;

    int threadID = GetCurrentThreadId();

    while (true)

    {

        FD\_ZERO(&read);

        int number = 0;

        for (int i = startIndex; i < g\_count && number < MAX\_SOCKET\_PER\_THREAD; i++)

        {

            printf("Thread %d: Add socket %d(%d) to the read set\n", threadID, i, g\_client[i]);

            FD\_SET(g\_client[i], &read);

            number += 1;

        }

        timeval t;

        t.tv\_sec = 1;

        t.tv\_usec = 0;

        select(0, &read, NULL, NULL, &t);

        number = 0;

        for (int i = startIndex; i < g\_count && number < MAX\_SOCKET\_PER\_THREAD; i++, number++)

        {

            if (FD\_ISSET(g\_client[i], &read))

            {

                char buffer[1024];

                memset(buffer, 0, sizeof(buffer));

                recv(g\_client[i], buffer, sizeof(buffer), 0);

                printf("Socket %d: %s\n", g\_client[i], buffer);

                for (int j = 0; j < g\_count; j++)

                {

                    if (j != i)

                    {

                        send(g\_client[j], buffer, strlen(buffer), 0);

                    }

                }

            }

        }

    }

    return 0;

}

int main()

{

    WSADATA DATA;

    WSAStartup(MAKEWORD(2, 2), &DATA);

    SOCKET s = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

    struct sockaddr\_in saddr;

    saddr.sin\_family = AF\_INET;

    saddr.sin\_port = htons(8888);

    saddr.sin\_addr.S\_un.S\_addr = INADDR\_ANY;

    bind(s, (sockaddr\*)&saddr, sizeof(saddr));

    listen(s, 10);

    fd\_set read;

    while (true)

    {

        FD\_ZERO(&read);

        FD\_SET(s, &read); //Them cac socket da co vao trong tap tham do

        for (int i = 0; i < g\_count && i < MAX\_SOCKET\_PER\_THREAD; i++)

        {

            printf("Thread %d: Add socket %d (%d) to the read set\n", GetCurrentThreadId(), i, g\_client[i]);

            FD\_SET(g\_client[i], &read);

        }

        select(0, &read, NULL, NULL, NULL);

        if (FD\_ISSET(s, &read))

        {

            SOCKET c;

            struct sockaddr\_in caddr;

            int clen = sizeof(caddr);

            c = accept(s, (sockaddr\*)&caddr, &clen);

            printf("Socket %d connected\n", c);

            g\_client[g\_count] = c;

            g\_count += 1;

            if (g\_count > 1 && ((g\_count - 1) % MAX\_SOCKET\_PER\_THREAD == 0))

            {

                DWORD ID = 0;

                CreateThread(NULL, 0, ClientThread, (LPVOID)(g\_count - 1), 0, &ID);

            }

        }

        for (int i = 0; i < g\_count; i++)

        {

            if (FD\_ISSET(g\_client[i], &read))

            {

                char buffer[1024];

                memset(buffer, 0, sizeof(buffer));

                recv(g\_client[i], buffer, sizeof(buffer), 0);

                printf("Socket %d: %s\n", g\_client[i], buffer);

                for (int j = 0; j < g\_count; j++)

                {

                    if (j != i)

                    {

                        send(g\_client[j], buffer, strlen(buffer), 0);

                    }

                }

            }

        }

    }

    return 0;

}

* AsyncSelect Example.cpp 14/5

#define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS

#define \_CRT\_SECURE\_NO\_WARNINGS

// AsyncSelect Example.cpp : Defines the entry point for the application.

//

#include "framework.h"

#include "AsyncSelect Example.h"

#include <WinSock2.h>

#include <stdio.h>

#define MAX\_LOADSTRING 100

#define WM\_ACCEPT WM\_USER + 1

#define WM\_READ\_CLOSE WM\_USER + 2

#define WM\_DISCONNECT WM\_USER + 3

// Global Variables:

HINSTANCE hInst; // current instance

WCHAR szTitle[MAX\_LOADSTRING]; // The title bar text

WCHAR szWindowClass[MAX\_LOADSTRING]; // the main window class name

// Forward declarations of functions included in this code module:

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

LRESULT CALLBACK MyWndProc(HWND, UINT, WPARAM, LPARAM);

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

SOCKET s, data;

HWND hWnd;

SOCKET clients[1024];

int status[1024]; //0 neu da disconnect, 1 neu van dang connect

int count = 0;

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// TODO: Place code here.

// Initialize global strings

LoadStringW(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

LoadStringW(hInstance, IDC\_ASYNCSELECTEXAMPLE, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// Perform application initialization:

if (!InitInstance (hInstance, nCmdShow))

{

return FALSE;

}

WSADATA DATA;

WSAStartup(MAKEWORD(2, 2), &DATA);

s = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

SOCKADDR\_IN saddr;

saddr.sin\_family = AF\_INET;

saddr.sin\_port = htons(21);

saddr.sin\_addr.S\_un.S\_addr = 0;

bind(s, (sockaddr\*)&saddr, sizeof(saddr));

listen(s, 10);

WSAAsyncSelect(s, hWnd, WM\_ACCEPT, FD\_ACCEPT); //Sau lenh nay thi moi ket noi den ma can accept se lam phat sinh mot su kien vao cua so voi ma 1234

HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_ASYNCSELECTEXAMPLE));

MSG msg;

// Main message loop:

while (GetMessage(&msg, nullptr, 0, 0))

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int) msg.wParam;

}

//

// FUNCTION: MyRegisterClass()

//

// PURPOSE: Registers the window class.

//

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEXW wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = MyWndProc; //Dia chi cua ham se duoc goi ra de xu ly cac su kien phat sinh tren cua so

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_ASYNCSELECTEXAMPLE));

wcex.hCursor = LoadCursor(nullptr, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);

wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_ASYNCSELECTEXAMPLE);

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

return RegisterClassExW(&wcex);

}

//

// FUNCTION: InitInstance(HINSTANCE, int)

//

// PURPOSE: Saves instance handle and creates main window

//

// COMMENTS:

//

// In this function, we save the instance handle in a global variable and

// create and display the main program window.

//

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

hInst = hInstance; // Store instance handle in our global variable

hWnd = CreateWindowW(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInstance, nullptr);

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

return TRUE;

}

LONGLONG FileTime\_to\_POSIX(FILETIME ft)

{

// takes the last modified date

LARGE\_INTEGER date, adjust;

date.HighPart = ft.dwHighDateTime;

date.LowPart = ft.dwLowDateTime;

// 100-nanoseconds = milliseconds \* 10000

adjust.QuadPart = 11644473600000 \* 10000;

// removes the diff between 1970 and 1601

date.QuadPart -= adjust.QuadPart;

// converts back from 100-nanoseconds to seconds

return date.QuadPart / 10000000;

}

//

// FUNCTION: WndProc(HWND, UINT, WPARAM, LPARAM)

//

// PURPOSE: Processes messages for the main window.

//

// WM\_COMMAND - process the application menu

// WM\_PAINT - Paint the main window

// WM\_DESTROY - post a quit message and return

//

//

LRESULT CALLBACK MyWndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

SOCKET s, c;

SOCKADDR\_IN caddr;

int clen = sizeof(caddr);

char buffer[1024];

static SOCKADDR\_IN data\_addr;

static char\* currentDir = (char\*)calloc(1024, 1);

static char\* content = (char\*)calloc(1024 \* 1024, 1);

static char\* WorkingDir = (char\*)calloc(1024, 1);

switch (message)

{

case WM\_READ\_CLOSE:

c = (SOCKET)wParam;

if (LOWORD(lParam) & FD\_READ)

{

memset(buffer, 0, sizeof(buffer));

recv(c, buffer, sizeof(buffer), 0);

if (strstr(buffer, "USER"))

{

char\* tmp = (char\*)"331 Can password\r\n";

send(c, tmp, strlen(tmp), 0);

}

if (strstr(buffer, "PASS"))

{

char\* tmp = (char\*)"230 Login ngon roi\r\n";

send(c, tmp, strlen(tmp), 0);

strcpy(currentDir, "257 \"/\" is current directory.\r\n");

strcpy(content, "type=dir;modify=20200421102857; .NET\r\ntype=file;modify=20200421102857; Sample.doc\r\n");

strcpy(WorkingDir, "C:/Temp");

}

if (strstr(buffer, "SYST"))

{

char\* tmp = (char\*)"215 FTP CUA VINH\r\n";

send(c, tmp, strlen(tmp), 0);

}

if (strstr(buffer, "FEAT"))

{

char\* tmp = (char\*)"211-Features:\r\n MDTM\r\n REST STREAM\r\n SIZE\r\n MLST type\*; size\*; modify\*;\r\n MLSD\r\n UTF8\r\n CLNT\r\n MFMT\r\n EPSV\r\n EPRT\r\n211 End\r\n";

send(c, tmp, strlen(tmp), 0);

}

if (strstr(buffer, "CLNT"))

{

char\* tmp = (char\*)"200 Khong can biet cai nay lam gi\r\n";

send(c, tmp, strlen(tmp), 0);

}

if (strstr(buffer, "OPTS"))

{

char\* tmp = (char\*)"202 OK chap nhan truyen du lieu kieu nay\r\n";

send(c, tmp, strlen(tmp), 0);

}

if (strstr(buffer, "PWD"))

{

char tmp[1024];

memset(tmp, 0, sizeof(tmp));

if (strcmp(WorkingDir, "C:/Temp") == 0)

{

sprintf(tmp, "257 \"/\" is current directory\r\n");

}else

sprintf(tmp, "257 \"%s\" is current directory\r\n", WorkingDir + strlen("C:/Temp"));

send(c, tmp, strlen(tmp), 0);

}

if (strstr(buffer, "TYPE"))

{

char\* tmp = (char\*)"200 OK\r\n";

send(c, tmp, strlen(tmp), 0);

}

if (strstr(buffer, "PORT"))

{

char\* tmp = (char\*)"200 OK\r\n";

send(c, tmp, strlen(tmp), 0);

for (int i = 0; i < strlen(buffer); i++)

{

if (buffer[i] == ',') buffer[i] = ' ';

}

char port[1024];

char ip1[1024], ip2[1024], ip3[1024], ip4[1024], p1[1024], p2[1024];

sscanf(buffer, "%s%s%s%s%s%s%s", port, ip1, ip2, ip3, ip4, p1, p2);

unsigned short uport = atoi(p1) \* 256 + atoi(p2);

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

data\_addr.sin\_port = htons(uport);

data\_addr.sin\_addr.S\_un.S\_un\_b.s\_b1 = atoi(ip1);

data\_addr.sin\_addr.S\_un.S\_un\_b.s\_b2 = atoi(ip2);

data\_addr.sin\_addr.S\_un.S\_un\_b.s\_b3 = atoi(ip3);

data\_addr.sin\_addr.S\_un.S\_un\_b.s\_b4 = atoi(ip4);

}

if (strstr(buffer, "MLSD"))

{

//Quet thu muc hien thoi WorkingDir va tao xau content

char findPath[1024];

WIN32\_FIND\_DATAA fData;

memset(findPath, 0, sizeof(findPath));

sprintf(findPath, "%s/\*.\*", WorkingDir);

HANDLE hFind = FindFirstFileA(findPath, &fData);

//type=dir;modify=20200421102857; .NET\r\n

if (hFind != INVALID\_HANDLE\_VALUE)

{

strcpy(content, "");

char\* type = (char\*)(fData.dwFileAttributes & FILE\_ATTRIBUTE\_DIRECTORY ? "dir" : "file");

sprintf(content, "type=%s;modify=%ld; %s\r\n", type, (long)FileTime\_to\_POSIX(fData.ftLastWriteTime), fData.cFileName);

while (FindNextFileA(hFind, &fData) == TRUE)

{

char\* type = (char\*)(fData.dwFileAttributes & FILE\_ATTRIBUTE\_DIRECTORY ? "dir" : "file");

sprintf(content + strlen(content), "type=%s;modify=%ld; %s\r\n", type, (long)FileTime\_to\_POSIX(fData.ftLastWriteTime), fData.cFileName);

}

}

char\* tmp = (char\*)"150 \r\n";

send(c, tmp, strlen(tmp), 0);

data = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

connect(data, (sockaddr\*)&data\_addr, sizeof(data\_addr));

send(data, content, strlen(content), 0);

closesocket(data);

char\* tmp2 = (char\*)"226 Successfully transferred \"/\"\r\n";

send(c, tmp2, strlen(tmp2), 0);

}

if (strstr(buffer, "CWD"))

{

char dir[1024];

memset(dir, 0, 1024);

strncpy(dir, buffer + 4, strlen(buffer) - 6);

if (strcmp(dir, "/") == 0) //Thu muc goc cua FTP Client

{

strcpy(WorkingDir, "C:/Temp");

}

else

sprintf(WorkingDir + strlen(WorkingDir), "/%s", dir);

char\* tmp = (char\*)"250 CWD Successful\r\n";

send(c, tmp, strlen(tmp), 0);

}

if (strstr(buffer, "CDUP"))

{

int i = 0;

for (i = strlen(WorkingDir) - 1; i > 0; i--)

{

if (WorkingDir[i] != '/')

WorkingDir[i] = 0;

else

break;

}

WorkingDir[i] = 0;

char\* tmp = (char\*)"200 CDUP successful. \"/\" is the current directory\r\n";

send(c, tmp, strlen(tmp), 0);

}

}

if (LOWORD(lParam) & FD\_CLOSE)

{

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

{

if (clients[i] == c)

{

status[i] = 0; //Kenh nay da disconnect

break;

}

}

char\* msg = (char\*)"Da co mot client roi room\n";

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

{

if (clients[i] != c && status[i] == 1) //Chi gui cho cac client dang ket noi

{

send(clients[i], msg, strlen(msg), 0);

}

}

}

break;

case WM\_ACCEPT:

s = (SOCKET)wParam;

c = accept(s, (sockaddr\*)&caddr, &clen);

{

char\* tmp = (char\*)"220 READY\r\n";

send(c, tmp, strlen(tmp), 0);

clients[count] = c;

status[count] = 1;

count += 1;

WSAAsyncSelect(c, hWnd, WM\_READ\_CLOSE, FD\_READ | FD\_CLOSE); //Sau lenh nay neu co du lieu tren socket c thi cua so se nhan duoc thong diep WM\_READ

}

break;

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

// Parse the menu selections:

switch (wmId)

{

case IDM\_ABOUT:

DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

}

break;

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

// TODO: Add any drawing code that uses hdc here...

EndPaint(hWnd, &ps);

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

// Message handler for about box.

INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_COMMAND:

if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)

{

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

}

break;

}

return (INT\_PTR)FALSE;

}