#define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

#include <vector>

#include <fstream>

#include <hash\_set>

#include <queue>

#include <WinSock2.h>

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

using namespace std;

queue<string> URL;

hash\_set<string> visitedurl;

hash\_set<string> visitedimg;

int g\_ImgCnt = 1;

#define DEFAULT\_PAGE\_BUF\_SIZE 1048576;//默认页面大小

// 1. http:// 2. jandan.net/ 3.ooxx

// http://jandan.net/ooxx/page-15#comments

bool ParseURL(const string &url, string &host, string &resource , string &document , string &mia)

{

size\_t found = url.find("http://"); //1

if (found == string::npos)

return false;

found += strlen("http://");

size\_t found1 = url.find\_first\_of('/', found);

if (found1 == string::npos)

return false;

host = url.substr(found, found1 - found); //2

size\_t found2 = url.find\_first\_of('/',found1);

if (found2 == string::npos)

return false;

resource = url.substr(found1,found2 - found1); //3

size\_t found3 = url.find\_first\_of('#', found2);

if (found2 == string::npos)

return false;

document = url.substr(found2, found3 - found2); //3

mia = url.substr(found3,url.size()-found3);

return true;

}

bool gethttpresponse(const string &host, const string &resource, string &response, int &bytes)

{

struct hostent \*hp = gethostbyname(host.c\_str());

if (hp == NULL) {

cout << "不能解析出主机地址！" << endl;

return false;

}

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

int nNetTimeout = 1000;

setsockopt(sock, SOL\_SOCKET, SO\_RCVTIMEO, (char \*)&nNetTimeout, sizeof(int));

if (sock == -1 || sock == -2) {

cout << "不能创建socket！" << endl;

return false;

}

//建立服务器地址

SOCKADDR\_IN sa;

sa.sin\_family = AF\_INET;

sa.sin\_port = htons(80);

memcpy(&sa.sin\_addr, hp->h\_addr, 4);

//连接服务器

if (connect(sock, (SOCKADDR\*)&sa, sizeof(sa)) != 0) {

cout << "不能连接服务器！" << endl;

return false;

}

//准备发送数据

string request = "GET " + resource + " HTTP/1.1\r\nHost:" + host + "\r\nConnection:Close\r\n\r\n";

if (SOCKET\_ERROR == send(sock, request.c\_str(), request.size(), 0)) {

cout << "发送数据错误！" << endl;

return false;

}

//接收数据

int m\_page\_bufsize = DEFAULT\_PAGE\_BUF\_SIZE;

char \* buf = new char[m\_page\_bufsize];

memset(buf, 0, m\_page\_bufsize);

int bytesread = 0;

int ret = 1;

cout << "读取：";

while (ret > 0) {

ret = recv(sock, buf + bytesread, m\_page\_bufsize - bytesread, 0);

if (ret > 0) {

bytesread += ret;

}

if (m\_page\_bufsize - bytesread < 100) {

cout << endl << "重新分配空间！" << endl;

char \* mbuf = new char[2 \* m\_page\_bufsize];

strcpy(mbuf, buf);

delete[] buf;

buf = mbuf;

}

cout << ret << " ";

}

cout << endl;

buf[bytesread] = '\0';

response.assign(buf, bytesread);

delete[] buf;

bytes = bytesread;

closesocket(sock);

return true;

}

bool ParseHtml(const string& response, vector<string> &imgurls )

{

string resource;

string url;

string http = "href=\"http://";

size\_t found = response.find(http);

ofstream ofile("url.txt", ios::app);

while (found != string::npos) {

found += strlen("href=\"");

SIZE\_T found1 = response.find('"', found + 1);

string tmpurl = response.substr(found, found1 - found);

if (visitedurl.find(tmpurl) == visitedurl.end()) {

visitedurl.insert(visitedurl.end(), tmpurl);

if (visitedurl.size() > 100000)

visitedurl.clear();

ofile << tmpurl << endl;

URL.push(tmpurl);

}

found = response.find(http, found1);

}

ofile.close();

string img = "http://";

found = response.find(img);

while (found != string::npos) {

SIZE\_T found1 = response.find('"', found + 1);

if (found1 == string::npos)

return true;

// imgurl http://ww3.sinaimg.cn/mw600/0073ob6Pgy1g59qgmtp8hj30rs0kuacs.jpg

string imgurl = response.substr(found, found1 - found);

size\_t found0 = imgurl.find("http://"); //1

if (found0 == string::npos)

return false;

found0 += strlen("http://");

size\_t found1 = url.find\_first\_of('/', found0);

if (found1 == string::npos)

return false;

size\_t found2 = url.find\_first\_of('/', found1);

if (found2 == string::npos)

return false;

resource = url.substr(found1, found2 - found1);

if (resource == "mw600" || resource == "thumb180") {

found = response.find(img, found1 + imgurl.size());

continue;

}

if (visitedimg.find(imgurl) == visitedimg.end()) {

visitedimg.insert(visitedimg.end(), imgurl);

if (visitedimg.size() > 100000)

visitedimg.clear();

imgurls.push\_back(imgurl);

}

found = response.find(img, found1 + imgurl.size());

}

cout << "结束解析这个网页" << endl;

return true;

}

//去除特殊符号

bool Tofilename(const string url, string &filename)

{

int size = url.size();

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

if (url[i] != '\*' && url[i] != '\\'&& url[i] != '/'

&& url[i] != ':'&& url[i] != '?'&& url[i] != '<'

&& url[i] != '>'&& url[i] != '|'&& url[i] != '"'

&& url[i] != '.' && url[i] != '-' && url[i] != ' ') {

filename += url[i];

}

}

filename += ".txt";

return true;

}

void Downloads(const vector<string> &imgurls, const string &url)

{

int size = imgurls.size();

string filename;

if (Tofilename(url, filename) == 0) {

cout << "转换名字错误" << endl;

return;

}

filename = "./img";

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

string str = imgurls[i];

SIZE\_T found = str.find\_last\_of('.');

string ext = imgurls[i].substr(found + 1, str.size() - found - 1);

if (ext.compare("jpg") && ext.compare("jpeg") && ext.compare("png") && ext.compare("gif") && ext.compare("bmp"))

continue;

string host;

string resource;

if (!ParseURL(imgurls[i], host, resource)) {

cout << "网址错误!" << endl;

return;

}

string image;

int bytes = 0;

if (gethttpresponse(host, resource, image, bytes)) {

if (image.size() == 0) {

cout << "传回数据错误！" << endl;

continue;

}

size\_t found = image.find("\r\n\r\n");

if (found == string::npos) {

cout << "传回数据错误！" << endl;

continue;

}

found += strlen("\r\n\r\n");

if (found == bytes) {

cout << "传回数据错误！" << endl;

continue;

}

int index = imgurls[i].find\_last\_of("/");

if (index != string::npos) {

string imgname = imgurls[i].substr(index, imgurls[i].size() - index);

ofstream ofile(filename + imgname, ios::binary);

if (!ofile.is\_open())

continue;

cout << g\_ImgCnt++ << filename + imgname << endl;

ofile.write(&image[found], bytes - found - 1);

ofile.close();

}

}

}

}

//广度优先

void BFS(string url)

{

string host;

string resource;

string document;

string mia;

if (!ParseURL(url, host, resource,document, mia)) {

cout << "网址错误!" << endl;

return;

}

string response = "";

int bytes = 0;

if (!gethttpresponse(host, resource, response, bytes)) {

cout << "没有得到网页响应！" << endl;

return;

}

//解析网页中所有的URL地址

if (response.size() == 0) {

cout << "服务器返回数据错误！" << endl;

return;

}

//存储网页返回信息

string filename;

if (Tofilename(url, filename) == 0) {

cout << "转换名字错误!" << endl;

return;

}

ofstream ofile("./html/" + filename);

if (ofile.is\_open()) {

ofile << response << endl;

ofile.close();

}

else {

cout << "打开文件错误！" << endl;

return;

}

vector<string> imgurls;

if (ParseHtml(response, imgurls) == 0) {

cout << "解析网页错误" << endl;

return;

}

Downloads(imgurls, url);

}

int main()

{

WSADATA wsadata;

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

cout << "加载winsockt动态库失败" << endl;

return 1;

}

CreateDirectoryA("./img", 0);

CreateDirectoryA("./html", 0);

//读取种子文件URL

string urlstart;

int i;

for (i = 1;i <= 15;i++)

{

urlstart = "http://jandan.net/ooxx/page-"+to\_string(i)+"#comments";

/\*fstream ifile("HrefURL.txt");

if (!ifile.is\_open())

{

cout << "打开文件错误！" << endl;

return 1;

} \*/

while (getline(ifile, urlstart))

{

URL.push(urlstart);

visitedurl.insert(urlstart);

}

ifile.close();

while (URL.size() > 0)

{

string str = URL.front();

cout << str << endl;

//广度优先，寻找种子文件里面所有url

BFS(str);

URL.pop();

cout << "URL数目：" << URL.size() << endl;

if (g\_ImgCnt > 200)

break;

}

}

WSACleanup();

return 0;

}