# 一、操作說明

## 1.執行程式，分兩種執行方法

### (1) 互動模式

```
PS C:\Users\翁芳婷\桌面\作業\大三上\新增資料夾\computer_network\project1\test_code> ./website_2
Enter the URL: http://hsccl.us.to/index.htm
Enter the output directory: demo
```

### (2) 參數模式

```
./website_2 http://hsccl.us.to/index.htm dic
```

## 2.執行結束，列出下載時間和總共下載檔案

包括基本題

a. 檢查使用者輸入的參數是否有誤，並且輸出適當的說明訊息

b. 後顯示狀態（URL 是否有效）

c. 統計資訊(檔案數量、下載容量、下載時間)

進階功能:

a. 下載過程顯示各物件的下載狀況（物件資訊、進度、速度）。

```
URL is valid.
Output directory created: ./new1


Downloaded image: ./img/httphscclustoimagesanimalscatsimagejpg.jpg
Download speed: 5848.84 KB/s
Download size: 221353  bytes
Downloading progress: 1 / 3


Downloaded image: ./img/httphscclustoimagejpg.jpg
Download speed: 8471.73 KB/s
Download size: 320729  bytes
Downloading progress: 2 / 3


Downloaded image: ./img/httphscclustoimagesimagejpg.jpg
Download speed: 8194.71 KB/s
Download size: 243102  bytes
Downloading progress: 3 / 3

Total execution time: 0.111 seconds

Total downloaded files: 3
HTML file modified successfully.
```

## 3.打開資料夾，有剛剛下載的三個檔案

| 名稱 | 日期 | 類型 | 大小 | 標籤 |
|---|---|---|---|---|
| httphscclustoimag... | 2023/11/26 下午 08:32 | 图片文件(.jpg) | 314 KB | |
| httphscclustoimag... | 2023/11/26 下午 08:32 | 图片文件(.jpg) | 217 KB | |
| httphscclustoimag... | 2023/11/26 下午 08:32 | 图片文件(.jpg) | 238 KB | |

4. 離線瀏覽網站(到剛剛輸入的目錄)


test_code > demo > httphscclustoindexhtm.html > html > body > img.s1

```
<!DOCTYPE html>
<html>

<head>
    <title>Computer Networks (Term Project: Part 1)</title>
    <style>
        body {
            font-family: Tahoma, Verdana, Arial, sans-serif;
        }

        img.s1 {
            height: 25%;
            width: 25%;
        }
    </style>
</head>

<body>
    <h1>Test Website for Computer Networks</h1>

    <h3>Image 1</h3>
    <img class="s1" src="../img/httphscclustoimagesimagejpg.jpg" alt="cat1">

    <h3>Image 2 </h3>
    <img class="s1" src="../img/httphscclustoimagejpg.jpg" alt="cat2">

    <h3>Image 3 </h3>
    <img class="s1" src="../img/httphscclustoimagesimagejpg.jpg" alt="cat3">

    <p><em>Images on this site were downloaded from Pixabay.</em></p>
</body>

</html>
```

# 二、程式碼說明

## 1.首先引入會使用到的函式庫

題目規定不能使用 http request/response 相關函式庫，所以這裡使用”WinSock2.h”，來做相關處理。首先初始化 Winsock。

```
#include <string>
#include <iostream>
#include <fstream>
#include <vector>
#include "WinSock2.h"
#include <time.h>
#include <queue>
#include <direct.h>
#include <unordered_set>
#include <regex>
#include <unordered_map>
#include <sys/stat.h>
#include <time.h>
#include <chrono>

using namespace std;
using namespace std::chrono;

#pragma comment(lib, "ws2_32.lib")

#define DEFAULT_PAGE_BUF_SIZE 1048576
queue<string> hrefUrl;
unordered_set<string> visitedUrl;
unordered_set<string> visitedImg;
unordered_map<string, string> originalImgUrls;
```

```
int main(int argc, char *argv[])
{
    WSADATA wsaData;
    if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0)
    {
        return 1;
    }
```

## 2.透過使用者執行的檔案的方式，做相對應處理。

假如輸入>=3，使用命令列模式;假設輸入==1，使用互動模式，在這裡會先用 IsURLValid 函式，判斷輸入的 url 是否符合格式，並且創建使用者輸入的目錄，如果無法建立，則輸出”Failed to create output directory.”。

(1) maim 函式

```
427     if (argc >= 3)
428     {
429         string userURL = argv[1];
430         string outputDir = argv[2];
431         if (IsURLValid(userURL))
432         {
433             cout << "URL is valid." << endl;
434         }
435         else
436         {
437             cout << "Invalid URL format." << endl;
438         }
439         _mkdir("./img");
440         userOutputDir = "./" + outputDir;
441         if (_mkdir(userOutputDir.c_str()) == 0)
442         {
443             cout << "Output directory created: " << userOutputDir << endl;
444         }
445         else
446         {
447             cout << "Failed to create output directory." << endl;
448         }
449         urlStart = userURL;
450         dirfile = userOutputDir;
451     }
```

```
452     else if (argc == 1)
453     {
454         string userURL, outputDir;
455 
456         cout << "Enter the URL: ";
457         getline(cin, userURL);
458 
459         cout << "Enter the output directory: ";
460         getline(cin, outputDir);
461 
462         if (IsURLValid(userURL))
463         {
464             cout << "URL is valid." << endl;
465         }
466         else
467         {
468             cout << "Invalid URL format." << endl;
469         }
```

(2) IsURLValid 函式

```
24  bool IsURLValid(const string &url)
25  {
26      // 定義 URL 的正則表達式
27      regex urlRegex("^https?://[\\w\\.-]+(:\\d+)?(/\\S*)?$");
28 
29      // 使用 regex_match 函數檢查是否匹配
30      return regex_match(url, urlRegex);
31  }
```

3.初始化參數後，對使用者輸入的 url 進行 http get

(1) main 函式

假設得到 HTTP GET，其中包含網站的 source code，將 response 寫入使用者輸入的目錄裡。

```
492     char *response;
493     int bytes;
494     int totalDownloadedFiles = 0;
495     int totalDownloadSize = 0;
496     string outputdir;
497     if (GetHttpResponse(urlStart, response, bytes))
498     {
499         string httpResponse = response;
500         // free(response);
501         const char *htmlContentStart = strstr(response, "\r\n\r\n");
502         if (htmlContentStart != nullptr)
503         {
504             htmlContentStart += strlen("\r\n\r\n");
505 
506             // Save only the HTML content to the file
507             string htmlContent = htmlContentStart;
508 
509             filename = ToFileName(urlStart);
510             outputdir = dirfile.c_str();
511             // cout << "outputdir" << outputdir << endl;
512             ofstream ofile(outputdir + "/" + filename);
513             if (ofile.is_open())
514             {
515                 ofile << htmlContent << endl;
516                 ofile.close();
517             }
518         }
```

(2) GetHttpResponse 函式

首先判斷 url 是否為相對 url，透過 ParseURL 函式解析 url，建立 socket 後設置連接地址，並與主機建立連接，發送 HTTP GET 請求，最後得到 HTTP response，而得到的 response 包括網頁 source code，在這裡只輸出 source code，html 標頭資訊不輸出，只記錄，這個是透過參考資料在重寫的函式庫。

```cpp
bool GetHttpResponse(const string &url, char *&response, int &bytesRead)
{
    string fullUrl;
    string host, resource;
    if (url.find("http://") == string::npos)
    {
        // 如果是相對 URL
        fullUrl = "http://" + host + "/" + url;
    }
    else
    {
        fullUrl = url;
    }

    // 解析 URL
    if (!ParseURL(fullUrl, host, resource))
    {
        cout << "Can not parse the url" << endl;
        return false;
    }

    // 建立 socket
    struct hostent *hp = gethostbyname(host.c_str());
    if (hp == NULL)
    {
        cout << "Can not find host address" << endl;
        return false;
    }

    SOCKET sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if (sock == -1 || sock == -2)
    {
        cout << "Can not create sock." << endl;
        return false;
    }
```

```cpp
    // 設置 Socket 連接的地址
    SOCKADDR_IN sa;
    sa.sin_family = AF_INET;
    sa.sin_port = htons(80);
    memcpy(&sa.sin_addr, hp->h_addr, 4);

    // 通過 connect 函數與主機建立連接
    if (0 != connect(sock, (SOCKADDR *)&sa, sizeof(sa)))
    {
        cout << "Can not connect: " << fullUrl << endl;
        closesocket(sock);
        return false;
    };

    // 構建 HTTP GET 請求並通過 send 函數發送 HTTP GET請求
    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 << "send error" << endl;
        closesocket(sock);
        return false;
    }
```

```
117     // 接收 HTTP response
118     int m_nContentLength = DEFAULT_PAGE_BUF_SIZE;
119     char *pageBuf = (char *)malloc(m_nContentLength);
120     memset(pageBuf, 0, m_nContentLength);
121 
122     bytesRead = 0;
123     int ret = 1;
124     while (ret > 0)
125     {
126         ret = recv(sock, pageBuf + bytesRead, m_nContentLength - bytesRead, 0);
127 
128         if (ret > 0)
129         {
130             bytesRead += ret;
131         }
132 
133         if (m_nContentLength - bytesRead < 100)
134         {
135             cout << "\nRealloc memorry" << endl;
136             m_nContentLength *= 2;
137             pageBuf = (char *)realloc(pageBuf, m_nContentLength); // 重新分配內存
138         }
139         }
140     cout << endl;
141 
142     pageBuf[bytesRead] = '\0';
143     response = pageBuf;
144     closesocket(sock);
145     return true;
146 }
```

## (3) ParseURL 函式

解析 url 並解析 url 的主機名和 resource，並儲存解析結果

```
34  bool ParseURL(const string &url, string &host, string &resource)
35  {
36      if (strlen(url.c_str()) > 2000)
37      {
38          return false;
39      }
40
41      const char *pos = strstr(url.c_str(), "http://");
42      if (pos == NULL)
43          pos = url.c_str();
44      else
45          pos += strlen("http://");
46      if (strstr(pos, "/") == NULL)
47          return false;
48      char pHost[100];
49      char pResource[2000];
50      sscanf(pos, "%[^/]%s", pHost, pResource);
51      host = pHost;
52      resource = pResource;
53      return true;
54  }
```

## 3. 解析 html 檔案裡面的物件 url，並下載

確定已將使用者輸入的 url 解析完，接下來對 html 內容的物件進行解析，在這裡直接將下載圖片放入/img，並且透過解析完的 html 的 imgurl，知道總共需要下載幾個檔案，並記錄下載的時間，確認下載檔案完，在做 download file 加總。

```
489         string host;
490         string resource;
491         vector<string> originurls;
492         if (ParseURL(urlStart, host, resource))
493         {
494             vector<string> imgurls;
495             vector<string> pageUrls;
496 
497             originurls = HTMLParse(httpResponse, imgurls, pageUrls, host);
498             // debug
499             //  for (int i = 0; i < originurls.size(); ++i)
500             //  {
501             //      cout << "originurls:" << originurls[i] << endl;
502             //  }
503             clock_t start = clock();
504             for (int i = 0; i < imgurls.size(); i++)
505             {
506                 string imgFileName = ToIMGFileName(imgurls[i]);
507                 string imgUrl = imgurls[i];
508                 string localPath = "./img/" + imgFileName;
509                 // debug
510                 //  cout << "localpath_inside:" << localPath << endl;
511                 string reoriginurl = originurls[i];
512                 originalImgUrls[reoriginurl] = localPath;
513                 if (DownloadImage(imgUrl, "./img/" + imgFileName, originalImgUrls, filename))
514                 {
515                     totalDownloadedFiles++;
516                     cout << "Downloading progress: " << (i + 1) << " / " << imgurls.size() << endl
517                         << endl;
518                     // totalDownloadedBytes += imageBytes;
519                 }
520             }
521             clock_t end = clock();
```

```
522             double elapsedTime = double(end - start) / CLOCKS_PER_SEC;
523             cout << "Total execution time: " << elapsedTime << " seconds" << endl
524                 << endl;
525             cout << "Total downloaded files: " << totalDownloadedFiles << endl;
526         }
527         else
528         {
529             cout << "Failed to parse URL: " << urlStart << endl;
530         }
531         ModifyHTMLImagesWithLocalPaths(outputdir + "/" + filename, originalImgUrls);
532     }
533     else
534     {
535         cout << "Failed to get HTTP response." << endl;
536     }
537 
538     WSACleanup();
539     return 0;
540 }
```

## (2)HTMLParse 函式

因為這次不用做遞迴下載，所以只解析圖檔的 html，在這裡重新定義了圖檔的 url，假設 html 的圖檔 url 不包括”http://”，就利用 host 來重新定義圖片的 url。

```
vector<string> HTMLParse(string &htmlResponse, vector<string> &imgurls, vector<string> &pageUrls, const string &host, vector<string> &originurls)
{
    const char *p = htmlResponse.c_str();
    const char *tag = "href=\"";
    const char *pos = strstr(p, tag);

    tag = "<img ";
    const char *att1 = "src=\"";
    const char *att2 = "lazy-src=\"";
    const char *pos0 = strstr(p, tag);
    while (pos0)
    {
        pos0 += strlen(tag);
        const char *pos2 = strstr(pos0, att2);
        if (!pos2 || pos2 > strstr(pos0, ">"))
        {
            pos = strstr(pos0, att1);
            if (!pos)
            {
                pos0 = strstr(att1, tag);
                continue;
            }
            else
            {
                pos = pos + strlen(att1);
            }
        }
        else
        {
            pos = pos2 + strlen(att2);
        }
```

```
            const char *nextQ = strstr(pos, "\"");
            if (nextQ)
            {
                char *url = new char[nextQ - pos + 1];
                sscanf(pos, "%[^\"]", url);
                originurls.push_back(url);
                string imgUrl = url;

                // Check if lazy-src exists, if yes, use it as the image URL
                const char *lazySrc = strstr(pos0, att2);
                if (lazySrc && lazySrc < nextQ)
                {
                    sscanf(lazySrc + strlen(att2), "%[^\"]", url);
                    imgUrl = url;
                }

                // Construct full URL if imgUrl is a relative path
                if (imgUrl.find("http") != 0)
                {
                    // Check if imgUrl starts with "/"
                    if (imgUrl.find("/") == 0)
                    {
                        imgUrl = "http://" + host + imgUrl;
                    }
                    else
                    {
                        // If it doesn't start with "/", it's a relative path; append it to the current page URL
                        imgUrl = "http://" + host + "/" + imgUrl;
                    }
                }
                if (visitedImg.find(imgUrl) == visitedImg.end())
                {
                    visitedImg.insert(imgUrl);
                    imgurls.push_back(imgUrl);
                    originurls.push_back(url);
                }
```

## (3)DownloadImage 函式

在一次透過 http request 下載圖片，這裡的 url 是透過 HTMLParse 函式解析出的圖檔 url，還加上計算下載檔案的大小、速度。

```cpp
bool DownloadImage(const string &imageUrl, const string &outputFileName, unordered_map<string, string> &originalImgUrls, const string &filename)
{
    char *response;
    int bytes;

    // 用 GET 請求獲取圖片
    if (!GetHttpResponse(imageUrl, response, bytes))
    {
        cout << "Failed to get HTTP response for image: " << imageUrl << endl;
        return false;
    }

    // 檢查 Content-Type
    const string contentTypeHeader = "Content-Type:";
    const char *contentTypePos = strstr(response, contentTypeHeader.c_str());
    if (contentTypePos != nullptr)
    {
        contentTypePos += contentTypeHeader.length();
        const char *endOfLine = strstr(contentTypePos, "\r\n");
        if (endOfLine != nullptr)
        {
            string contentType(contentTypePos, endOfLine - contentTypePos);
            // 檢查 Content-Type 是否為圖片格式
            if (contentType.find("image/") != string::npos)
            {
                // 寫入圖片文件
                ofstream imageFile(outputFileName, ios::binary);
                if (imageFile.is_open())
                {
                    // 找到圖片數據的開始位置
                    const char *imageDataStart = strstr(response, "\r\n\r\n");
                    if (imageDataStart != nullptr)
                    {
                        imageDataStart += strlen("\r\n\r\n");
                        int imageBytes = bytes - (imageDataStart - response);
                        imageFile.write(imageDataStart, imageBytes);
                        imageFile.close();
                        cout << "Downloaded image: " << outputFileName << endl;
                        // originalImgUrls[imageUrl] = outputFileName;
                        free(response);
                        auto endTime = steady_clock::now();
                        auto elapsedTime = duration_cast<milliseconds>(endTime - startTime).count() / 1000.0; // 換算成秒

                        // 計算速度
                        double speed = bytes / elapsedTime / 1024.0; // 每秒的 KB 數

                        // 顯示速度和預計剩餘完成時間
                        cout << "Download speed: " << speed << " KB/s" << endl;
                        cout << "Download size: " << imageBytes << "  bytes" << endl;

                        return true;
                    }
                    else
                    {
                        cout << "Failed to find image data in HTTP response." << endl;
                    }
                }
                else
                {
                    cout << "Failed to create image file: " << outputFileName << endl;
                }
            }
        }
    }
```

## (4) ModifyHTMLImagesWithLocalPaths 函式

這個函式是為了修改將 html 的圖檔路徑，以便離線瀏覽，首先透過 unordered_map 建立 html 的圖檔 url 會與哪個下載圖檔路徑對應，如果 url 有與 html 的圖檔 url 相同，就將其替換成下載本地路徑，這裡主要是透過正規表示式更換，這樣可以確保與圖檔 url 完全一樣，才可以換。

```cpp
void ModifyHTMLImagesWithLocalPaths(const string &filePath, const unordered_map<string, string> &originalImgUrls)
{

    ifstream inputFile(filePath);
    if (!inputFile.is_open())
    {
        cerr << "Failed to open input file: " << filePath << endl;
        return;
    }

    // 讀取整個文件內容
    string fileContent((istreambuf_iterator<char>(inputFile)), istreambuf_iterator<char>());
    inputFile.close();

    for (const auto &pair : originalImgUrls)
    {

        const string &originurls = pair.first;
        const string &localPath = pair.second;

        // debug
        //  cout << "originurls: " << pair.first << ", localpath: " << pair.second << endl;

        string pattern = "(src\\s*=\\s*\")(" + originurls + ")(\")";
        std::regex regexPattern(pattern);
        std::smatch match;

        if (std::regex_search(fileContent, match, regexPattern))
        {
            std::string srcMatch = match[0].str();
            std::string replacedSrc = match[1].str() + "." + localPath + match[3].str();
            fileContent.replace(match.position(), srcMatch.length(), replacedSrc);
        }
    }
```

```cpp
    ofstream outputFile(filePath);
    if (!outputFile.is_open())
    {
        cerr << "Failed to open output file: " << filePath << endl;
        return;
    }

    outputFile << fileContent;
    outputFile.close();

    cout << "HTML file modified successfully." << endl;
}
```

## 4. 整體架構

整體架構主要是透過建立 http request 和 response，得到該網址的 sourcecode，並解析出 html 裡面的所有物件 url，並在做一次 http request，得到允許下載，在解析物件 url 時，因為 html 的 source code 不一定完整的 url，所以要做相對變更，才可下載物件，為了允許線瀏覽，要把 html 裡 source code 的物件路徑位置更改成自己本地位置，利用比對的方式，作相對路徑更改即可離線瀏覽。

主要使用的函式

```
GetHttpResponse -> ParseURL -> HTMLParse -> DownloadImage ->
ModifyHTMLImagesWithLocalPaths
```

## 三、參考資料:


1. C++網路爬蟲的實現——WinSock 程式設計
   https://www.796t.com/content/1550065531.html
2. 手把手教你用 Winsock 创建 socket server 和 client 原创
   https://blog.csdn.net/weixin_38369492/article/details/111032276
3. C++编写爬虫脚本爬取网站图片原创
   https://blog.csdn.net/NKU_Yang/article/details/109489127