

計算機網路

Project 1

B1029006 唐芳婷

# 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

- 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
- 後顯示狀態（URL 是否有效）
- 統計資訊（檔案數量、下載容量、下載時間）

進階功能：

- 下載過程顯示各物件的下載狀況（物件資訊、進度、速度）。

```
./hscclustocluster/index.html new1
URL is valid.
Output directory created: ./new1

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

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

Downloaded image: ./img/httpshscclustoimagesimagejpg.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
 httpscclustoimag...	2023/11/26 下午 08:32	图片文件(jpg)	314 KB	
 httpscclustoimag...	2023/11/26 下午 08:32	图片文件(jpg)	217 KB	
 httpscclustoimag...	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)

← → ↻ 127.0.0.1:5500/test\_code/demo/httpscclustoindexhtml.html

### Test Website for Computer Networks

Image 1



Image 2



```
test_code > demo > httpscclustoindexhtml.html > html > body > img.s1

1  <!DOCTYPE html>
2  <html>
3
4  <head>
5      <title>Computer Networks (Term Project: Part 1)</title>
6      <style>
7          body {
8              font-family: Tahoma, Verdana, Arial, sans-serif;
9          }
10
11         img.s1 {
12             height: 25%;
13             width: 25%;
14         }
15     </style>
16 </head>
17
18 <body>
19     <h1>Test Website for Computer Networks</h1>
20
21     <h3>Image 1</h3>
22     
23
24     <h3>Image 2 </h3>
25     
26
27     <h3>Image 3 </h3>
28     
29
30     <p><em>Images on this site were downloaded from Pixabay.</em></p>
31 </body>
32
33 </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。

```
1  #include <string>
2  #include <iostream>
3  #include <fstream>
4  #include <vector>
5  #include "WinSock2.h"
6  #include <time.h>
7  #include <queue>
8  #include <direct.h>
9  #include <unordered_set>
10 #include <regex>
11 #include <unordered_map>
12 #include <sys/stat.h>
13 #include <time.h>
14 #include <chrono>
15
16 using namespace std;
17 using namespace std::chrono;
18
19 #pragma comment(lib, "ws2_32.lib")
20
21 #define DEFAULT_PAGE_BUF_SIZE 1048576
22 queue<string> hrefUrl;
23 unordered_set<string> visitedUrl;
24 unordered_set<string> visitedImg;
25 unordered_map<string, string> originalImgUrls;
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417 int main(int argc, char *argv[])
418 {
419     WSADATA wsaData;
420     if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0)
421     {
422         return 1;
423     }
```

### 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) main 函式

```
427     if (argc >= 3)
428     {
429         string userURL = argv[1];
430         string outputDir = argv[2];
431         if (IsValidURL(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
453     else if (argc == 1)
454     {
455         string userURL, outputDir;
456
457         cout << "Enter the URL: ";
458         getline(cin, userURL);
459
460         cout << "Enter the output directory: ";
461         getline(cin, outputDir);
462
463         if (IsValidURL(userURL))
464         {
465             cout << "URL is valid." << endl;
466         }
467         else
468         {
469             cout << "Invalid URL format." << endl;
470         }
471     }
```

## (2) IsValidURL 函式

```
24 bool IsValidURL(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         }
519     }
```

#### (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。

```

57 bool GetHttpResponse(const string &url, char *&response, int &bytesRead)
58 {
59     string fullUrl;
60     string host, resource;
61     if (url.find("http://") == string::npos)
62     {
63         // 如果是相對 URL
64         fullUrl = "http://" + host + "/" + url;
65     }
66     else
67     {
68         fullUrl = url;
69     }
70
71     // 解析 URL
72     if (!ParseURL(fullUrl, host, resource))
73     {
74         cout << "Can not parse the url" << endl;
75         return false;
76     }
77
78     // 建立 socket
79     struct hostent *hp = gethostbyname(host.c_str());
80     if (hp == NULL)
81     {
82         cout << "Can not find host address" << endl;
83         return false;
84     }
85
86     SOCKET sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
87     if (sock == -1 || sock == -2)
88     {
89         cout << "Can not create sock." << endl;
90         return false;
91     }

```

```

93     // 設置 Socket 連接的地址
94     SOCKADDR_IN sa;
95     sa.sin_family = AF_INET;
96     sa.sin_port = htons(80);
97     memcpy(&sa.sin_addr, hp->h_addr, 4);
98
99     // 通過 connect 函數與主機建立連接
100    if (0 != connect(sock, (SOCKADDR *)&sa, sizeof(sa)))
101    {
102        cout << "Can not connect: " << fullUrl << endl;
103        closesocket(sock);
104        return false;
105    };
106
107    // 構建 HTTP GET 請求並通過 send 函數發送 HTTP GET請求
108    string request = "GET " + resource + " HTTP/1.1\r\nHost:" + host + "\r\nConnection:Close\r\n\r\n";
109
110    if (SOCKET_ERROR == send(sock, request.c_str(), request.size(), 0))
111    {
112        cout << "send error" << endl;
113        closesocket(sock);
114        return false;
115    }

```

```

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 memory" << endl;
136         m_nContentLength *= 2;
137         pageBuf = (char *)realloc(pageBuf, m_nContentLength); // 重新分配内存
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      {
44          pos = url.c_str();
45      }
46      else
47      {
48          pos += strlen("http://");
49          if (strstr(pos, "/") == NULL)
50          {
51              return false;
52          }
53          char pHost[100];
54          char pResource[2000];
55          sscanf(pos, "%[^/]%s", pHost, pResource);
56          host = pHost;
57          resource = pResource;
58          return true;
59      }
60  }

```

### 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                     // cout << 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         // cout << 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。

```

149 vector<string> HTMLParse(string &htmlResponse, vector<string> &imgurls, vector<string> &pageUrls, const string &host, vector<string> &originurls)
150 {
151     const char *p = htmlResponse.c_str();
152     const char *tag = "href=\"";
153     const char *pos = strstr(p, tag);
154
155     tag = "<img ";
156     const char *att1 = "src=\"";
157     const char *att2 = "lazy-src=\"";
158     const char *pos0 = strstr(p, tag);
159     while (pos0)
160     {
161         pos0 += strlen(tag);
162         const char *pos2 = strstr(pos0, att2);
163         if (!pos2 || pos2 > strstr(pos0, ">"))
164         {
165             pos = strstr(pos0, att1);
166             if (!pos)
167             {
168                 pos0 = strstr(att1, tag);
169                 continue;
170             }
171             else
172             {
173                 pos = pos + strlen(att1);
174             }
175         }
176         else
177         {
178             pos = pos2 + strlen(att2);
179         }
180     }

```

```

181     const char *nextQ = strstr(pos, "\"");
182     if (nextQ)
183     {
184         char *url = new char[nextQ - pos + 1];
185         sscanf(pos, "%[^\"]", url);
186         originurls.push_back(url);
187         string imgUrl = url;
188
189         // Check if lazy-src exists, if yes, use it as the image URL
190         const char *lazySrc = strstr(pos0, att2);
191         if (lazySrc && lazySrc < nextQ)
192         {
193             sscanf(lazySrc + strlen(att2), "%[^\"]", url);
194             imgUrl = url;
195         }
196
197         // Construct full URL if imgUrl is a relative path
198         if (imgUrl.find("http") != 0)
199         {
200             // Check if imgUrl starts with "/"
201             if (imgUrl.find("/") == 0)
202             {
203                 imgUrl = "http://" + host + imgUrl;
204             }
205             else
206             {
207                 // If it doesn't start with "/", it's a relative path; append it to the current page URL
208                 imgUrl = "http://" + host + "/" + imgUrl;
209             }
210         }
211         if (visitedImg.find(imgUrl) == visitedImg.end())
212         {
213             visitedImg.insert(imgUrl);
214             imgurls.push_back(imgUrl);
215             originurls.push_back(url);
216         }
217     }

```

### (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。

```

323 bool DownloadImage(const string &imageUrl, const string &outputFileName, unordered_map<string, string> &originalImgUrls, const string &filename)
324 {
325     char *response;
326     int bytes;
327
328     // 用 GET 請求獲取圖片
329     if (!GetHttpResponse(imageUrl, response, bytes))
330     {
331         cout << "Failed to get HTTP response for image: " << imageUrl << endl;
332         return false;
333     }
334
335     // 檢查 Content-Type
336     const string contentTypeHeader = "Content-Type:";
337     const char *contentTypePos = strstr(response, contentTypeHeader.c_str());
338     if (contentTypePos != nullptr)
339     {
340         contentTypePos += contentTypeHeader.length();
341         const char *endOfLine = strstr(contentTypePos, "\r\n");
342         if (endOfLine != nullptr)
343         {
344             string contentType(contentTypePos, endOfLine - contentTypePos);
345             // 檢查 Content-Type 是否為圖片格式
346             if (contentType.find("image/") != string::npos)
347             {
348                 // 寫入圖片文件
349                 ofstream imageFile(outputFileName, ios::binary);
350                 if (imageFile.is_open())
351                 {
352                     // 找到圖片數據的開始位置
353                     const char *imageDataStart = strstr(response, "\r\n\r\n");
354                     if (imageDataStart != nullptr)
355                     {
356                         imageDataStart += strlen("\r\n\r\n");
357                         int imageBytes = bytes - (imageDataStart - response);
358                         imageFile.write(imageDataStart, imageBytes);
359                         imageFile.close();
360                         cout << "Downloaded image: " << outputFileName << endl;
361                         // originalImgUrls[imageUrl] = outputFileName;
362                         free(response);
363                         auto endTime = steady_clock::now();
364                         auto elapsedTime = duration_cast<milliseconds>(endTime - startTime).count() / 1000.0; // 換算成秒
365
366                         // 計算速度
367                         double speed = bytes / elapsedTime / 1024.0; // 每秒的 KB 數
368
369                         // 顯示速度和預計剩餘完成時間
370                         cout << "Download speed: " << speed << " KB/s" << endl;
371                         cout << "Download size: " << imageBytes << " bytes" << endl;
372
373                         return true;
374                     }
375                     else
376                     {
377                         cout << "Failed to find image data in HTTP response." << endl;
378                     }
379                 }
380                 else
381                 {
382                     cout << "Failed to create image file: " << outputFileName << endl;
383                 }
384             }
385         }
386     }
387 }

```

#### (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。

```

256 void ModifyHTMLImagesWithLocalPaths(const string &filePath, const unordered_map<string, string> &originalImgUrls)
257 {
258
259     ifstream inputFile(filePath);
260     if (!inputFile.is_open())
261     {
262         cerr << "Failed to open input file: " << filePath << endl;
263         return;
264     }
265
266     // 讀取整個文件內容
267     string fileContent((istreambuf_iterator<char>(inputFile), istreambuf_iterator<char>()));
268     inputFile.close();
269
270     for (const auto &pair : originalImgUrls)
271     {
272
273         const string &originurls = pair.first;
274         const string &localPath = pair.second;
275
276         // debug
277         // cout << "originurls: " << pair.first << ", localpath: " << pair.second << endl;
278
279         string pattern = "(src\\s*=\\s*\"")(\" + originurls + "\")\"";
280         std::regex regexPattern(pattern);
281         std::smatch match;
282
283         if (std::regex_search(fileContent, match, regexPattern))
284         {
285             std::string srcMatch = match[0].str();
286             std::string replacedSrc = match[1].str() + "." + localPath + match[3].str();
287             fileContent.replace(match.position(), srcMatch.length(), replacedSrc);
288         }
289     }
290

```

```

292     ofstream outputFile(filePath);
293     if (!outputFile.is_open())
294     {
295         cerr << "Failed to open output file: " << filePath << endl;
296         return;
297     }
298
299     outputFile << fileContent;
300     outputFile.close();
301
302     cout << "HTML file modified successfully." << endl;
303 }
304

```

#### 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](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](https://blog.csdn.net/NKU_Yang/article/details/109489127)