網路程式設計 HW-簡易的網頁爬蟲程式

一、 程式說明

程式執行時，輸入要讀取的網頁 ip 位址(如: 127.0.0.1)、埠號(如: 80)、起始頁面(如: index.html)，即可取得其網頁資訊，並且一併擷取此網頁中所有連結到的頁面的內容。

二、 執行畫面

正常執行程式結果如下。

```
C:\data\Code\Cpp\network_programming\cmake-build-debug\hw3.exe
請輸入要連線的 ip 位址(如: 127.0.0.1):
127.0.0.1
請輸入要連線的 port number(如: 80):
8000
請輸入起始頁面(如: index.html):
index.html
[頁面] index.html
[回應內容]
<html>
<head>
        <title>index</title>
</head>
<body>
        <p>Hello, world!</p>
        <a href="1.html">link 1</a>
        <a href="2.html">link 2</a>
        <a href="3.html">link 3</a>
</body>
</html>
(185bytes)

[頁面] 1.html
[回應內容]
<html>
<head>
        <title>1</title>
</head>
<body>
        <p>1</p>
</body>
</html>
(79bytes)
```

```
[頁面] 2.html
[回應內容]
<html>
<head>
        <title>2</title>
</head>
<body>
        <p>2</p>
</body>
</html>
(79bytes)

[頁面] 3.html
[回應內容]
<html>
<head>
        <title>3</title>
</head>
<body>
        <p>3</p>
</body>
</html>
(79bytes)


Process finished with exit code 0
```

找不到網頁的執行結果如下。

```
C:\data\Code\Cpp\network_programming\cmake-build-debug\hw3.exe
請輸入要連線的 ip 位址(如: 127.0.0.1):
127.0.0.1
請輸入要連線的 port number(如: 80):
8000
請輸入起始頁面(如: index.html):
no_page.html
[錯誤] 此網頁找不到: no_page.html

Process finished with exit code 0
```

無法連線的執行結果如下。

```
C:\data\Code\Cpp\network_programming\cmake-build-debug\hw3.exe
請輸入要連線的 ip 位址(如: 127.0.0.1):
127.0.0.1
請輸入要連線的 port number(如: 80):
80
[錯誤] 無法連線, 錯誤訊息: 10061

Process finished with exit code 0
```

三、 程式碼

[hw3.cpp]
Github: https://github.com/linwebs/network_programming/blob/main/hw3.cpp

```
/*
 * NCYU 109 Network Programming HW3
 * Created by linwebs on 2021/3/22.
 */
#include <iostream>
#include <winsock.h>

// 定義回傳字元最大數量
#define MAXLINE 1024

using namespace std;

/*
 * 執行完畢結果
 * int status          => 執行狀態
 * int send_len        => 傳送的內容長度
 * int recv_len        => 接收到的內容長度
 * string send         => 傳送的內容
 * string recv         => 接收到的內容
 */
struct content {
    int status = -1;
    int send_len = 0;
    int recv_head_len = 0;
    int recv_content_len = 0;
```

```
    string page;
    string send;
    string recv_head;
    string recv_content;
};

/*
 * 連結結構
 * int next          => 下一個開始尋找的連結位置
 * string href       => 連結檔案
 */
struct href_struct {
    int next = -1;
    string href;
};

content *winsock_service(const char str[MAXLINE]);

content *get_page(const string &page);

void output(content *result);

bool check_http_status(content *result);

href_struct *find_href(const string &html, int start);

int find_href_tag(const string &html, int start);

int find_href_quot(const string &html, int start);

bool init_winsock_service(const char server_ip[16], u_short server_port);

void finish_winsock_service();

void start_search(const string &page);

SOCKET sd;
WSADATA wsadata;
```

```
struct sockaddr_in serv{};
string ip;              // ip address
u_short port;       // port number

int main() {
    string msg;
    cout << "請輸入要連線的 ip 位址(如: 127.0.0.1):" << endl;
    cin >> ip;
    //ip = "127.0.0.1";

    cout << "請輸入要連線的 port number(如: 80):" << endl;
    cin >> port;
    //port = 8000;

    // test connect
    if (!init_winsock_service(ip.c_str(), port)) {
        return 0;
    }

    finish_winsock_service();

    cout << "請輸入起始頁面(如: index.html):" << endl;
    // Input single line
    //getline(cin, msg);     // 第一次把 endl 吃掉
    //getline(cin, msg);

    // Input single string
    cin >> msg;
    //msg = "index.html";

    if (msg == "exit") {
        cout << "goodbye~" << endl;
    }

    start_search(msg);

    return 0;
}
```

```
/*
 * 開始搜尋頁面
 * @param page => 頁面
 */
void start_search(const string &page) {
    // result 結構 => 取得頁面
    content *result = get_page(page);
    if (result->status == 0) {
        if (check_http_status(result)) {
            output(result);
            int next = 0;

            // 尋找所有連結並輸出
            while (true) {
                href_struct *href = find_href(result->recv_content, next);
                if (href->next == -1) {
                    break;
                }
                next = href->next;

                content *sub_result = get_page(href->href);
                if (check_http_status(sub_result)) {
                    output(sub_result);
                } else {
                    cout << "[錯誤] 此網頁找不到: " << href->href << endl;
                }
            }
        } else {
            cout << "[錯誤] 此網頁找不到: " << page << endl;
        }
    }
}
```

```
/*
 * 取得單一頁面資訊
 * @param page => 頁面
 * @return     => 取得到的資訊
 */
content *get_page(const string &page) {
    string request = "GET /" + page + " HTTP/1.1\r\nHost: localhost\r\nConnection: close\r\n\r\n";
    auto *result = winsock_service(request.c_str());
    if (result->status == -1) {
        return result;
    }
    auto *result_page = new content;
    result_page->recv_head_len = result->recv_head_len;
    result_page->recv_head = result->recv_head;
    result_page->recv_content_len = result->recv_content_len;
    result_page->recv_content = result->recv_content;
    result_page->status = result->status;
    result_page->send = result->send;
    result_page->send_len = result->send_len;
    result_page->page = page;
    return result_page;
}

/*
 * 初始化 Windows socket 服務
 * @param server_ip   => 伺服器端 ip 位址 (ipv4 格式)
 * @param server_port => 伺服器端埠號
 * @return            => 初始化成功與否
 */
bool init_winsock_service(const char server_ip[16], u_short server_port) {
    // connect status
    int conn_status;

    // Call WSAStartup() to Register "WinSock DLL"
    if (WSAStartup(0x101, (LPWSADATA) &wsadata) == SOCKET_ERROR) {
        cout << "[錯誤] 無法啟動 Windows Sockets，錯誤訊息：" << WSAGetLastError() << endl;
```

```
        return false;
    }

    // Open a TCP socket
    sd = socket(AF_INET, SOCK_STREAM, 0);

    // Prepare for connect.
    // Include sockaddr_ing struct (serv)
    serv.sin_family = AF_INET;

    // server's ip address
    serv.sin_addr.s_addr = inet_addr(server_ip);

    // server's port number
    // htons: host to network
    serv.sin_port = htons(server_port);

    // connect to echo server
    conn_status = connect(sd, (LPSOCKADDR) &serv, sizeof(serv));

    if (conn_status == SOCKET_ERROR) {
        cout << "[錯誤] 無法連線，錯誤訊息: " << WSAGetLastError() << endl;
        closesocket(sd);
        WSACleanup();
        return false;
    } else {
        return true;
    }
}

/*
 * Socket 連線
 * @param str       => 傳送的內容
 * @return          => 取得到的資訊
 */
content *winsock_service(const char str[MAXLINE]) {
    // receive content
    auto *result = new content;
```

```
// init service
if (!init_winsock_service(ip.c_str(), port)) {
    result->status = -1;
    return result;
}

// send status
int send_status;

// receive string
char str_head[MAXLINE + 1] = "";
char str_content[MAXLINE + 1] = "";

// receive bytes
int recv_head_len, recv_content_len;

result->status = 0;

// send "str" to echo server
send_status = send(sd, str, int(strlen(str)) + 1, 0);

if (send_status == SOCKET_ERROR) {
    cout << "[錯誤] 無法傳送訊息，錯誤訊息: " << WSAGetLastError() << endl;

    result->status = -1;

    return result;
}

result->send = str;
result->send_len = int(strlen(str) + 1);

// receive http header
recv_head_len = recv(sd, str_head, MAXLINE, 0);

if (recv_head_len == SOCKET_ERROR) {
    cout << "[錯誤] 無法接收訊息，錯誤訊息: " << WSAGetLastError() << endl;
```

```
        result->status = -1;

        return result;
    }

    // receive http content
    recv_content_len = recv(sd, str_content, MAXLINE, 0);

    if (recv_content_len == SOCKET_ERROR) {
        cout << "[錯誤] 無法接收訊息，錯誤訊息: " << WSAGetLastError() << endl;

        result->status = -1;

        return result;
    }

    result->recv_head_len = recv_head_len;
    result->recv_head = str_head;

    result->recv_content_len = recv_content_len;
    result->recv_content = str_content;

    // close service
    finish_winsock_service();

    return result;
}

/*
 * 結束 Windows socket 服務
 */
void finish_winsock_service() {
    // close socket
    if (closesocket(sd) == SOCKET_ERROR) {
        cout << "[錯誤] 無法關閉 socket，錯誤訊息: " << WSAGetLastError() <<
endl;
    }
```

```
    // finish "WinSock DLL"
    if (WSACleanup() == SOCKET_ERROR) {
        cout << "[錯誤] 無法終止 Windows Sockets，錯誤訊息: " << WSAGetLastError() << endl;
    }
}

/*
 * 將結果輸出
 * @param result    => 執行結果指標
 */
void output(content *result) {
    //cout << "[傳送內容] " << result->send << "(" << result->send_len << "bytes)" << endl;
    cout << "[頁面] " << result->page << endl;
    //cout << "[回應標頭] " << endl << result->recv_head << "(" << result->recv_head_len << "bytes)" << endl;
    cout << "[回應內容] " << endl << result->recv_content << endl << "(" << result->recv_content_len << "bytes)" << endl << endl;
}

/*
 * 檢查頁面是否正常運作(HTTP 狀態碼是否為 200 OK)
 * @param result    => 執行結果
 * @return          => 正常與否
 */
bool check_http_status(content *result) {
    int pre = 9;
    size_t found = result->recv_head.find("\r\n");
    if (result->recv_head.substr(pre, found - pre) == "200 OK") {
        return true;
    }
    return false;
}
```

```
/*
 * 尋找 href 標籤文字
 * @param html      => 內容
 * @param start     => 開始尋找處
 * @return          => 尋找結果狀態
 */
int find_href_tag(const string &html, int start) {
    size_t found = html.find("href", start);
    if (found != std::string::npos) {
        return found;
    }
    return -1;
}

/*
 * 尋找 href 標籤的引號
 * @param html      => 內容
 * @param start     => 開始尋找處
 * @return          => 尋找結果狀態
 */
int find_href_quot(const string &html, int start) {
    size_t found = html.find("", start);
    if (found != std::string::npos) {
        return found;
    }
    return -1;
}

/*
 * 尋找單一 href 標籤
 * @param html      => 內容
 * @param start     => 開始尋找處
 * @return          => 尋找結果 href_struct 結構
 */
href_struct *find_href(const string &html, int start) {
    int href, head, foot;
    auto *result = new href_struct;
```

```
    // 尋找 href 標籤
    href = find_href_tag(html, start);

    if (href == -1) {
        result->href = "";
        result->next = -1;
    } else {
        // 尋找 href 後的第一個 "
        head = find_href_quot(html, href);

        // 尋找 href 後的第二個 "
        foot = find_href_quot(html, head + 1);

        // 取得兩個 " 之間的內容
        result->href = html.substr(head + 1, foot - head - 1);

        // 下一個開始尋找的位置
        result->next = foot + 1;
    }

    return result;
}
```

[index.html]
Github: https://github.com/linwebs/network_programming/blob/main/hw3/index.html

```
<html>
<head>
    <title>index</title>
</head>
<body>
    <p>Hello, world!</p>
    <a href="1.html">link 1</a>
    <a href="2.html">link 2</a>
    <a href="3.html">link 3</a>
</body>
</html>
```

[1.html]
Github: https://github.com/linwebs/network_programming/blob/main/hw3/1.html

```
<html>
<head>
    <title>1</title>
</head>
<body>
    <p>1</p>
</body>
</html>
```

[2.html]
Github: https://github.com/linwebs/network_programming/blob/main/hw3/2.html

```
<html>
<head>
    <title>2</title>
</head>
<body>
    <p>2</p>
</body>
</html>
```

[3.html]
Github: https://github.com/linwebs/network_programming/blob/main/hw3/3.html

```
<html>
<head>
    <title>3</title>
</head>
<body>
    <p>3</p>
</body>
</html>
```

四、 心得

這次的課程是寫出一個爬蟲的程式，以往聽到爬蟲都是使用 python 語言來撰寫，這次使用 C++語言來寫爬蟲，對我而言真的是一大的突破，沒有別人寫好的函式庫，分析的程式碼全都要自己寫，而且在進行 socket 連線時，每一行程式碼都有可能出錯，都需要撰寫例外的處理，為的就是程式的可執行性，這個作業真的是需要投注許多時間來撰寫，我盡力的將程式碼寫到高可讀性，為了就是將來有需要時可以再來回顧。