

Computer Network Lab2

111062122 李彥呈

Simulation Results:

下圖(圖 1)為我的程式的執行結果，如圖所示，我的程式能用 gcc 進行編譯，同時程式會印出和 spec 要求相對應的輸出。

```
kevinli@LAPTOP-ABRSF7T7:/mnt/c/Users/famil/Desktop/計網概/lab2$ gcc lab.c -o lab && ./lab
Please enter the URL:
can.cs.nthu.edu.tw
===== Socket =====
Sending HTTP request
Receiving the response
===== Hyperlinks =====
index.php
members.php
http://www.cs.nthu.edu.tw/~jungchuk/home.html
LAB/
gallery.php
contact.php
http://web.cs.nthu.edu.tw/files/14-1015-143485_r109-1.php?Lang=zh-tw
http://www.nthu.edu.tw
http://web.cs.nthu.edu.tw/bin/home.php
http://www.com.nthu.edu.tw/
http://www.highimpact-seo.co.uk/
=====
We have found 11 hyperlinks
kevinli@LAPTOP-ABRSF7T7:/mnt/c/Users/famil/Desktop/計網概/lab2$
```

▲圖 1—執行結果的截圖

How I handle the response:

首先，避免在做 string concatenation 的時候出現不可預期的錯誤，我先把 **response** 字串初始化，接著就可以開始接收 response。實作這個 feature 的話可以使用迴圈 **recv()**抑或是使用特殊的 **flag(recv())**的最後一個參數來讀取 response。我是選擇前者，迴圈內的動作就是不斷地接字串，同時要確保 **buffer[packet_len]**要是 **'\0'** 避免產生不可預期的錯誤，程式碼請參考下圖 2。

```
1 void receive_response(void) {
2     printf("Receiving the response\n");
3     memset(response, 0, sizeof(response)); // Initialize response to an empty string
4     while((packet_len = recv(client_socket, buffer, BUFFER_SIZE - 1, 0)) > 0) { // Leave room for null terminator
5         buffer[packet_len] = '\0'; // Null-terminate the received data
6         strncat(response, buffer, packet_len);
7         tot_len += packet_len;
8     }
9     if(packet_len == -1) {
10         perror("receive_response()");
11         exit(EXIT_FAILURE);
12     }
13 }
```

▲圖 2—Screenshot of **receive_response()**

How I extract the hyperlinks:

首先，我將存取 **hyperlink** 的陣列宣告為 1000*1000 大小(最後會透過 **free_memory()**把配置的記憶體釋放掉)，由於<a 和 href 之間可以有其他字元，

因此我先找到<a，有找到的話再繼續往後找 href，如果找到的話那我們就把 **quote_start** 指到目標 hyperlink 的開頭、**quote_end** 指到目標 hyperlink 的結尾，最後透過 **strcpy()**把 extracted hyperlink 貼到 **hyperlink** 陣列裡面，請參考圖 3 程式碼。

A screenshot of a code editor with a dark background and light-colored text. The code is written in C and defines a function named getlink(). The function starts by allocating an array named 'hyperlink' of 1000 elements, each a char*. It then iterates through the array, processing HTML response data. The code uses various string functions like strstr, strlen, strchr, and strcpy to parse HTML tags and extract URLs. Comments in the code explain the purpose of certain steps, such as moving the cursor to the start of the URL and ensuring null termination. The function ends with a closing brace for the void getlink() scope.

```
1 void getlink(void) {
2     hyperlink = (char**)malloc(1000 * sizeof(char*));
3     for(int i = 0; i < 1000; i++) hyperlink[i] = (char*)malloc(1000 * sizeof(char));
4     char *cursor = response;
5     int i = 0;
6     char *a_tag = NULL;
7     char *href_tag = NULL;
8     char *quote_start = NULL;
9     char *quote_end = NULL;
10
11     a_tag = strstr(cursor, "<a");
12     while(a_tag != NULL) {
13         href_tag = strstr(a_tag, "href=\"");
14         if(href_tag != NULL) {
15             quote_start = href_tag + strlen("href=\""); // Move the cursor to the start of the URL
16             quote_end = strchr(quote_start, '\"'); // Find the end of the URL
17             if(quote_end == NULL) {
18                 perror("getlink()");
19                 exit(EXIT_FAILURE);
20             }
21             strcpy(hyperlink[i], quote_start, quote_end - quote_start);
22             hyperlink[i][quote_end - quote_start] = '\0'; // Ensure null termination
23             cursor = quote_end;
24             i++, ans++;
25             a_tag = strstr(cursor, "<a");
26         }
27         else a_tag = strstr(a_tag + 2, "<a");
28     }
29 }
```

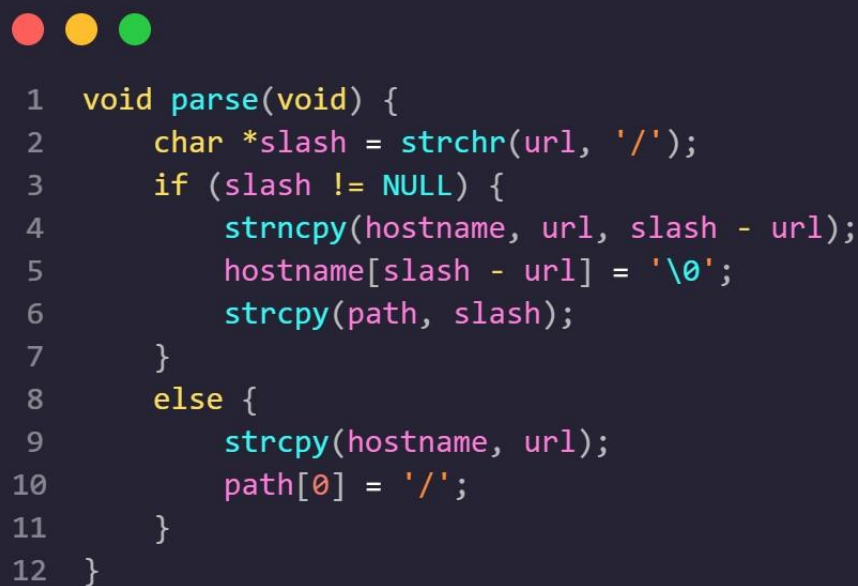
▲圖 3—Screenshot of **getlink()**

Functions that I use:

這次 lab 我把所有重要步驟 function 化，起初是為了好看，後來慶幸自己有這樣做，替我 debug 省了不少時間。

parse()

此 function 是把 URL 拆分成 hostname 跟 path，目的是等等要把 hostname 轉成 ip 位址，另一個目的是 path 會在 request 的指令裡派上用場。我的實作方式是透過 **strchr()**找到 '/' 以前的 URL，這個片段就是 hostname，剩下的就是 path，但是如果沒有找到 '/'，則要注意把 **path[0]**設為 '/'，請參考下圖(圖 4)的程式碼。



```

1 void parse(void) {
2     char *slash = strchr(url, '/');
3     if (slash != NULL) {
4         strncpy(hostname, url, slash - url);
5         hostname[slash - url] = '\0';
6         strcpy(path, slash);
7     }
8     else {
9         strcpy(hostname, url);
10        path[0] = '/';
11    }
12 }

```

▲圖 4—Screenshot of *parse()*

create_socket()

這個 function 基本上是參照投影片的 example 進行實作，值得注意的是 AF_INET 和 PF_INET 的數值雖然相同，混用也沒有關係，但其實二者還是有差別(參考網址在下圖(圖 5)的 comment)，故實作時我是遵照 PF_INET 是給創造 socket 用的規定。



```

1 void create_socket(void) {
2     // difference between PF_INET and AF_INET,
3     // https://spyker729.blogspot.com/2010/07/afinetpfinet.html
4     server_addr.sin_family = AF_INET;
5     server_addr.sin_addr.s_addr = inet_addr(server_ip);
6     server_addr.sin_port = htons(PORT);
7     if((client_socket = socket(PF_INET, SOCK_STREAM, 0)) == -1) {
8         perror("create_socket()");
9         exit(EXIT_FAILURE);
10    }
11 }

```

▲圖 5—Screenshot of *create_socket()*

connect_server()

這個 function 主要是透過 **connect()** 連接到 server，程式碼在下圖(圖 6)



▲圖 6—Screenshot of *connect_server()*

send_request()

這個 function 就是把連接好的 request message 傳出，下圖(圖 7,8)為 request 的相關程式碼。



▲圖 7—How I form my request message



```

1 void send_request(void) {
2     printf("Sending HTTP request\n");
3     if(send(client_socket, request, REQUEST_LEN, 0) == -1) {
4         perror("send_request()");
5         exit(EXIT_FAILURE);
6     }
7 }

```

▲圖 8—Screenshot of *send_request()*

receive_response()、getlink()

前面提到過，故略。

hostname_to_ip()

這個 function 能把 hostname 轉為 ip 位址(char 型式)，這段 code 是參考 <https://www.binarytides.com/hostname-to-ip-address-c-sockets-linux/>的(見 Reference)，程式碼見圖 9。



```

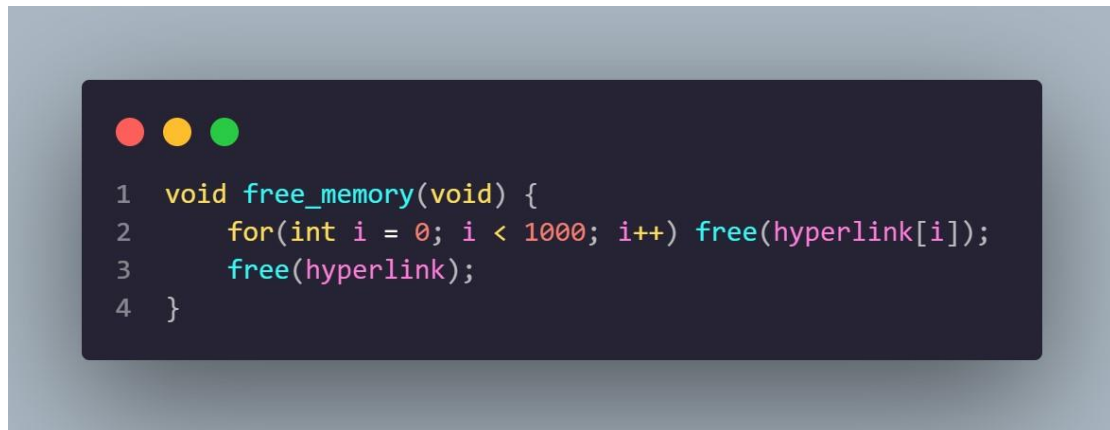
1 void hostname_to_ip(void) {
2     // Code reference https://www.binarytides.com/hostname-to-ip-address-c-sockets-linux/
3     struct hostent *he;
4     struct in_addr **addr_list;
5     if ((he = gethostbyname(hostname)) == NULL) {
6         perror("gethostbyname");
7         exit(EXIT_FAILURE);
8     }
9     addr_list = (struct in_addr **)he->h_addr_list;
10    strcpy(server_ip, inet_ntoa(*addr_list[0]));
11 }

```

▲圖 9—Screenshot of *hostname_to_ip()*

free_memory()

這個 function 即是把先前分配給 hyperlink 的記憶體釋放，以避免記憶體溢出，程式碼參考下圖 10。



▲圖 10—Screenshot of *free_memory()*

What I learned from this lab:

這次的 lab 很好玩，讓我除了知道理論外，也知道如何去實作。在寫 code 的過程中，我原本沒注意到和 href 之間可以有其他字元，因此我原本實作方式是在 response 裡面找 target 字串(target[] = "<a href=")，導致前面幾次執行都只有取到 10 條 hyperlink。除此之外，我學到一些 debug 小技巧，例如可以自己寫 error message 或是在執行到下一個 function 之前印出暫時的結果等等。

Reference

程式碼 line105~115

參考 <https://www.binarytides.com/hostname-to-ip-address-c-sockets-linux/>

create_socket()、connect_server()、sned_request()

參考 投影片