實驗環境：於 Linux 系統中，並使用 127.0.0.1 於本機利用 c 語言進行 socket 單機互傳。

1. Client / Srever 程式(1 個 Client 對 1 個 Server)(30%)

(1) 程式碼

a. Server 端

```
1.  #include <netinet/in.h>
2.  #include <sys/types.h>
3.  #include <sys/socket.h>
4.  #include <stdio.h>
5.  #include <stdlib.h>
6.  #include <string.h>
7.  #include <unistd.h>
8.  #include <arpa/inet.h>
9.
10. #define PORT 7777
11. #define BUFFER_SIZE 1024
12.
13. int main(int argc, char **argv)
14. {
15.     // 宣告一個 server_addr 來設定 server socket 的資訊
16.     struct sockaddr_in server_addr;
17.     // 把一段記憶體區的內容先全部設置為 0
18.     bzero(&server_addr,sizeof(server_addr));
19.     // 設置成 IPv4
20.     server_addr.sin_family = AF_INET;
21.     // 地址設置成 0.0.0.0，即任意的地址
22.     server_addr.sin_addr.s_addr = htons(INADDR_ANY);
23.     // 設置 port number 為 7777
24.     server_addr.sin_port = htons(PORT);
25.
26.     // 創建一個 socket，使用 IPv4、TCP
27.     int server_socket = socket(AF_INET,SOCK_STREAM,0);
28.     // 檢查 socket 是否有創建失敗
29.     if( server_socket < 0)
30.     {
31.         printf("[-]Create socket failed!");
32.         exit(1);
33.     }
34.
```

```
35.         // 將設定好的 address 及 port number 設定置創建好的 socket，錯誤回傳-1
36.         if( bind(server_socket,(struct sockaddr*)&server_addr,sizeof(server_addr)))
37.         {
38.             printf("[-]Server bind port : %d failed!", PORT);
39.             exit(1);
40.         }
41.         // 監聽 server socket 有無連線請求，最多 5 個 client 能連線，錯誤回傳-1
42.         if ( listen(server_socket, 5) )
43.         {
44.             printf("[-]Server listen failed!");
45.             exit(1);
46.         }
47.         // 無限迴圈，使 server socket 能一直運作
48.         while (1)
49.         {
50.             // 宣告一個 client_addr，來存放 client 端的資訊
51.             struct sockaddr_in client_addr;
52.             socklen_t length = sizeof(client_addr);
53.             // 接收 client 端的連線，並產出新的 socket，專與此 client 溝通
54.             int new_server_socket = accept(server_socket,(struct
    sockaddr*)&client_addr,&length);
55.             // 若 accept 返回-1 即代表錯誤
56.             if ( new_server_socket < 0)
57.             {
58.                 printf("[-]Server accept failed!\n");
59.                 break;
60.             }
61.             printf("[+]Client connect successfully!");
62.             // 創建一個 1024 格空間的 buffer array，並將其清成 0
63.             char buffer[1024];
64.             bzero(buffer, 1024);
65.             // server socket 將收到的 data 放進 buffer 裡面，並回傳收到的位元組數
66.             length = recv(new_server_socket,buffer,BUFFER_SIZE,0);
67.             // 若回傳的位元組數小於 0 則錯誤
68.             if (length < 0)
69.             {
70.                 printf("Server recieve data failed!\n");
71.                 break;
```

```
72.             }
73.             // 創建一個 1024 大小的 array，用來存放 client 想要下載的檔案名稱
74.             char file_name[1024];
75.             // 將 array 清空
76.             bzero(file_name, 1024);
77.             // 將剛剛收到後放在 buffer 的檔案名稱複製進入 file_name array
78.             strncpy(file_name, buffer, 1024);
79.             // 創建指向要讓 client 端下載的檔案指標
80.             FILE * fp = fopen(file_name,"r");
81.             // 若該指標為 NULL，即代表該檔案不存在
82.             if(fp == NULL)
83.             {
84.                 printf("File: %s not found\n", file_name);
85.             }
86.             else
87.             {
88.                 // 若檔案存在，就將 buffer 清空，準備傳送給 client 端
89.                 bzero(buffer, BUFFER_SIZE);
90.                 // 宣告一個整數型態的變數，用來接收 fread 回傳值
91.                 int file_block_length = 0;
92.                 // fread 的回傳值為成功有效的讀取 fp 指向的檔案的元素個數
93.                 while( (file_block_length =
    fread(buffer,sizeof(char),BUFFER_SIZE,fp))>0)
94.                 {
95.                     // 若元素個數至少有一個即開始傳送
96.                     // 發送 buffer 中的字串至 new_server_socket
97.                     if(send(new_server_socket,buffer,file_block_length,0)<0)
98.                     {
99.                         printf("Send file: %s failed\n", file_name);
100.                        break;
101.                    }
102.                    // 將 buffer 清空
103.                    bzero(buffer, BUFFER_SIZE);
104.                }
105.                // 關閉指向要傳輸檔案的指標
106.                fclose(fp);
107.                printf("File: %s transfer finished\n",file_name);
108.            }
```

```
109.            // 關閉與 client 端 socket 的連接
110.            close(new_server_socket);
111.        }
112.        // 關閉 server 端監聽所使用的 socket
113.        close(server_socket);
114.        return 0;
115. }
```

b. Client 端

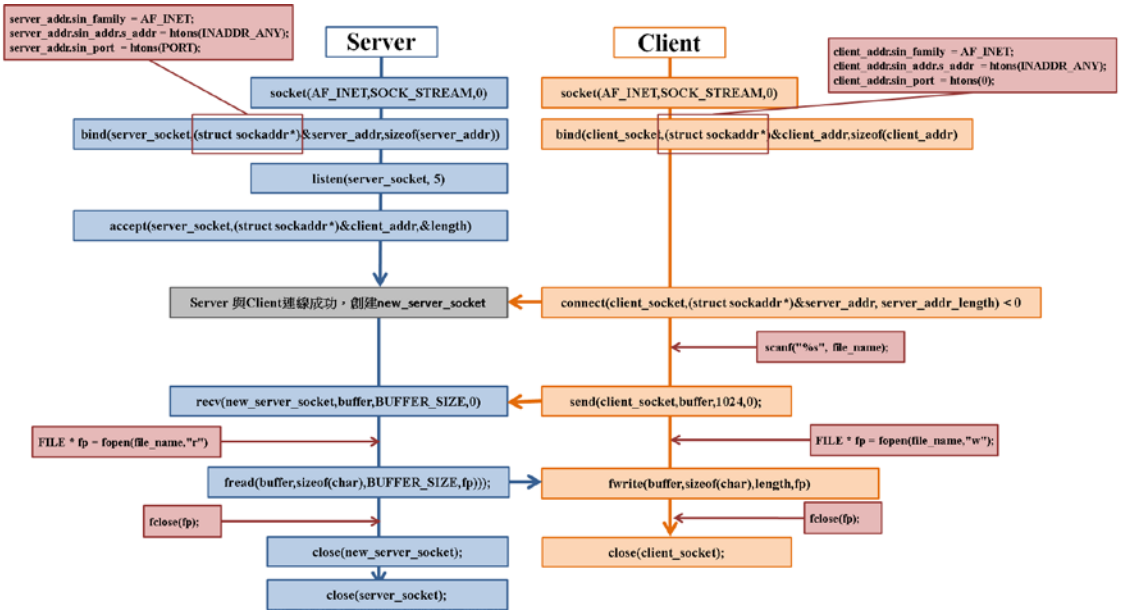
```
1.  #include <netinet/in.h>
2.  #include <sys/types.h>
3.  #include <sys/socket.h>
4.  #include <stdio.h>
5.  #include <stdlib.h>
6.  #include <string.h>
7.  #include <unistd.h>
8.  #include <arpa/inet.h>
9.
10. #define PORT 7777
11. #define BUFFER_SIZE 1024
12.
13. int main(int argc, char **argv)
14. {
15.     // 宣告一個 client_addr 來設定 client socket 的資訊
16.     struct sockaddr_in client_addr;
17.     // 把一段記憶體區的內容先全部設置為 0
18.     bzero(&client_addr,sizeof(client_addr));
19.     // 設置成 IPv4
20.     client_addr.sin_family = AF_INET;
21.     // 將地址設置為 0.0.0.0，即任意位置
22.     client_addr.sin_addr.s_addr = htons(INADDR_ANY);
23.     // 0 表示讓系統自動分配一個 port number
24.     client_addr.sin_port = htons(0);
25.     // 創建一個 socket，使用 IPv4、TCP
26.     int client_socket = socket(AF_INET,SOCK_STREAM,0);
27.     // 檢查 socket 是否有創建失敗
28.     if( client_socket < 0)
29.     {
```

```
30.             printf("[-]Create Socket Failed!\n");
31.             exit(1);
32.         }
33.         // 將設定好的 address 及 port number 設定置創建好的 socket，錯誤回傳-1
34.         if( bind(client_socket,(struct sockaddr*)&client_addr,sizeof(client_addr)))
35.         {
36.             printf("[-]Client Bind Port Failed!\n");
37.             exit(1);
38.         }
39.         // 宣告一個 server_addr 來設定 server socket 的資訊
40.         struct sockaddr_in server_addr;
41.         // 把一段記憶體區的內容先全部設置為 0
42.         bzero(&server_addr,sizeof(server_addr));
43.         // 設置成 IPv4
44.         server_addr.sin_family = AF_INET;
45.         // 將輸入在終端機的字串轉成數字後，將之設為伺服器的 address
46.         if(inet_aton(argv[1],&server_addr.sin_addr) == 0)
47.         {
48.             printf("[-]Server IP Address Error!\n");
49.             exit(1);
50.         }
51.         // 將 server port number 設置成 7777
52.         server_addr.sin_port = htons(PORT);
53.         // 取得 server_addr 的大小
54.         socklen_t server_addr_length = sizeof(server_addr);
55.         //向 server 發起連接請求，若錯誤則回傳-1
56.         if(connect(client_socket,(struct sockaddr*)&server_addr,
                        server_addr_length) < 0)
57.         {
58.             printf("[-]Can Not Connect To %s!\n",argv[1]);
59.             exit(1);
60.         }
61.         // 創建放置檔案名稱的 array
62.         char file_name[1024];
63.         bzero(file_name, 1024);
64.         // 要求 client 端輸入要從 server 端要載的檔案名稱
65.         printf("Please input a file name which is you want to download from server:
                        ");
```

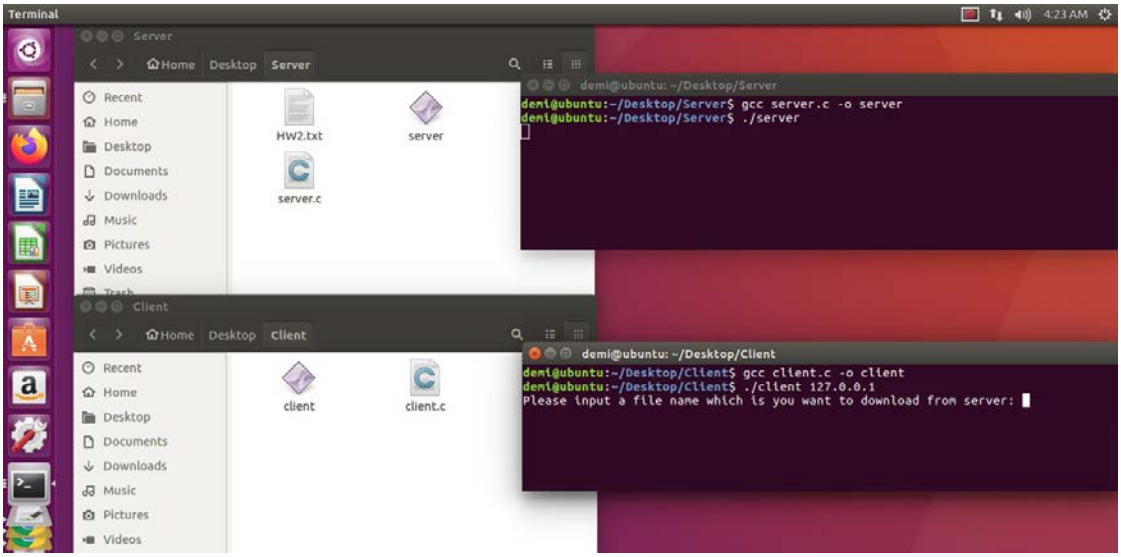
```
66.     scanf("%s", file_name);
67.     // 創建一個大小為 1024 的 client 端 buffer，並將其清空
68.     char buffer[1024];
69.     bzero(buffer,1024);
70.     // 將 client 端輸入的 file_name 放進 buffer 中
71.     strncpy(buffer, file_name, 1024);
72.     //將 buffer 中的資料送出去給 server
73.     send(client_socket,buffer,1024,0);
74.     // 創建一個指向下載的檔案名稱的指標，並創建此檔案
75.     FILE * fp = fopen(file_name,"w");
76.     // 若該指標為 NULL，則無法打開此檔寫入，即錯誤
77.     if(fp == NULL)
78.     {
79.         printf("File: %s Can Not Open To Write\n", file_name);
80.         exit(1);
81.     }
82.     // 清空 buffer
83.     bzero(buffer,BUFFER_SIZE);
84.     int length = 0;
85.     // client socket 將收到的 data 放進 buffer 裡面，並回傳收到的位元組數
86.     while( length = recv(client_socket,buffer,BUFFER_SIZE,0))
87.     {
88.         // 若回傳的位元數組小於 0，則代表錯誤
89.         if(length < 0)
90.         {
91.             printf("Recieve Data From Server %s Failed!\n", argv[1]);
92.             break;
93.         }
94.         // fwrite 的回傳值為成功有效的讀取 fp 指向的檔案的元素個數
95.         int write_length = fwrite(buffer,sizeof(char),length,fp);
96.         // 若 fwrite 取得的回傳的元素個數小於 recv 得到的即錯誤
97.         if (write_length<length)
98.         {
99.             printf("File:\t%s Write Failed\n", file_name);
100.            break;
101.        }
102.        // 清空 buffer
103.        bzero(buffer,BUFFER_SIZE);
```

```
104.        }
105.        printf("Recieve file: %s from server[%s] finished\n",file_name, argv[1]);
106.        // 完成將從 server 端下載的資料寫入檔案中後，關閉指向檔案的指標
107.        fclose(fp);
108.        //關閉 client 端 socket
109.        close(client_socket);
110.        return 0;
111.    }
```

(2) 架構流程圖

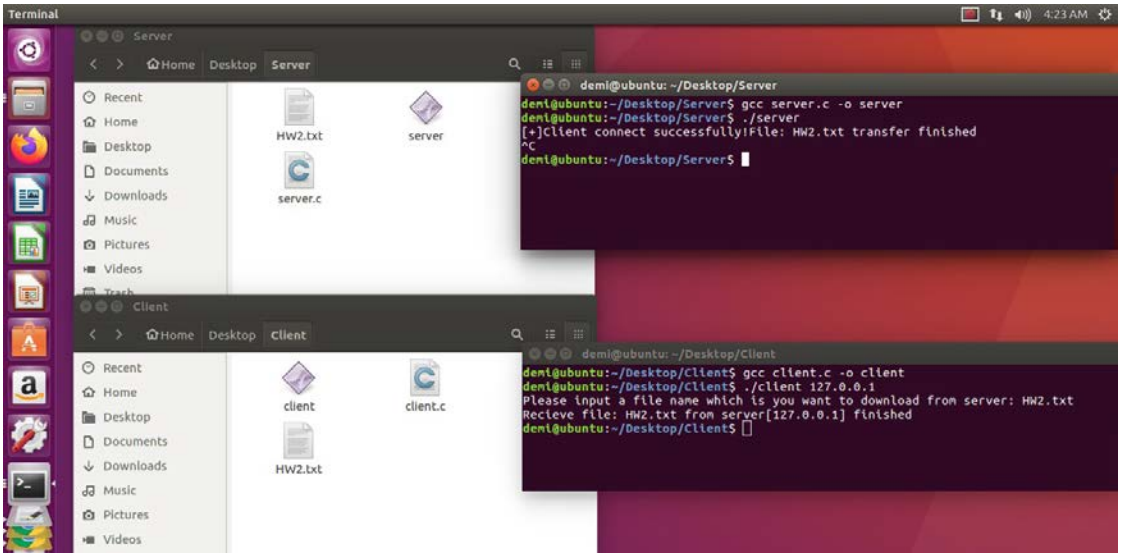


(3) 實驗過程、內容及結果

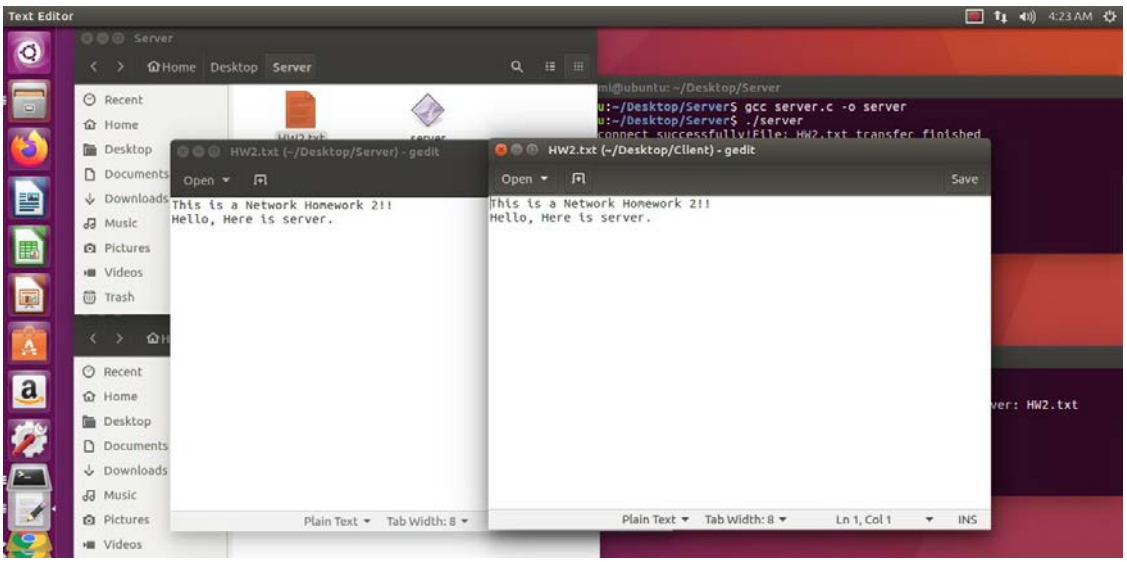


圖(1)：我將 server 及 client 分至於兩個資料夾，如同兩台電腦般。並開啟兩個終端機各自編譯後執行。另外在 server 的資料夾中有準備讓 client 端下載的檔案

HW2.txt，且在 client 端還未下載的時候其並未有此檔。



圖(2)：client 端輸入要下載的檔案名稱 HW2.txt，即在 client 的資料夾中，多出了一個從 server 端下載的 HW2.txt 檔案，並在 server 端顯示傳輸成功，在 client 端顯示接收成功。接收完檔案之後 client 端即關閉 socket，server 端也輸入 Ctrl+C 關閉。



圖(3)：打開 server 端和 client 端各自的 HW2.txt 檔案驗證，傳輸內容相同。下載成功！

2. Client/Srever 程式(多個 Client 對 1 個 Server)(30%)

(1) 程式碼

a. Server 端

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <string.h>
4.  #include <unistd.h>
5.  #include <sys/socket.h>
6.  #include <sys/types.h>
7.  #include <netinet/in.h>
8.  #include <arpa/inet.h>
9.
10. #define PORT 4444
11. #define HELLO_WORLD_SERVER_PORT 6666
12. #define LENGTH_OF_LISTEN_QUEUE 20
13. #define BUFFER_SIZE 1024
14. #define FILE_NAME_MAX_SIZE 512
15.
16. int main()
17. {
18.         int sockfd, ret;
19.         // 宣告一個 serverAddr 來設定 server socket 的資訊
20.         struct sockaddr_in serverAddr;
21.         int newSocket;
22.         struct sockaddr_in newAddr;
23.         socklen_t addr_size;
24.
25.         char buffer[1024];
26.         pid_t childpid;
27.         // 創建一個 socket，使用 IPv4、TCP
28.         sockfd = socket(AF_INET, SOCK_STREAM, 0);
29.         // 若回傳值小於 0，即創建錯誤
30.         if(sockfd < 0){
31.                 printf("[-]Error in connection.\n");
32.                 exit(1);
33.         }
34.         printf("[+]Server Socket is created.\n");
35.         // 先將 serverAddr 清空
```

```
36.         memset(&serverAddr, '\0', sizeof(serverAddr));
37.         // 設定成 IPv4
38.         serverAddr.sin_family = AF_INET;
39.         // 將 server port number 設為 4444
40.         serverAddr.sin_port = htons(PORT);
41.         // 將 server address 設置成 loopback address 127.0.0.1
42.         serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
43.         // 將設定好的 server address 和 port number 等資訊綁在 server socket 上
44.         ret = bind(sockfd, (struct sockaddr*)&serverAddr, sizeof(serverAddr));
45.         // 若 bind 回傳值小於 0 的時候即代表發生錯誤
46.         if(ret < 0){
47.             printf("[-]Error in binding.\n");
48.             exit(1);
49.         }
50.         printf("[+]Bind to port %d\n", 4444);
51.         // server socket 監聽是否有連線請求，且規定最多有五個 client 連至 server
52.         if(listen(sockfd, 5) == 0)
53.         {
54.             printf("[+]Listening....\n");
55.         }
56.         else
57.         {
58.             printf("[-]Error in binding.\n");
59.         }
60.         // 無限迴圈，使 server socket 能一直運作
61.         int num = 1;
62.         while(1){
63.             // 接收 client 端的連線，並產出新的 socket，專與此 client 溝通
64.             newSocket = accept(sockfd, (struct sockaddr*)&newAddr, &addr_size);
65.             // 若 accept 回傳的數值小於 0，即 server socket 接收連線請求發生錯
    誤
66.             if(newSocket < 0){
67.                 exit(1);
68.             }
69.             printf("Client %d connection accepted from %s:%d\n", num++,
    inet_ntoa(newAddr.sin_addr), ntohs(newAddr.sin_port));
70.             // fork 將當前的 process 分支出另一個 process，並回傳子程序的 pid 0
71.             if((childpid = fork()) == 0){
```

```
72.                 close(sockfd);
73.                 while(1){
74.                     // 從 newSocket 中取得 client 端輸入的字串並放置於 buffer
75.                     recv(newSocket, buffer, 1024, 0);
76.                     // 比較 buffer 和字串 exit;，若回傳值為 0，則兩者字串相同
77.                     if(strcmp(buffer, "exit;") == 0){
78.                     // 若 client 輸入的是 exit;，則 break 結束與該 client 的連線
79.                         printf("Disconnected from %s:%d\n",
    inet_ntoa(newAddr.sin_addr), ntohs(newAddr.sin_port));
80.                         break;
81.                     }
82.                     // 比較 buffer 和字串 download;，若回傳值為 0 則兩者字串相
    同
83.                     else if(strcmp(buffer, "download;") == 0){
84.                         // 創建一個指向要下載檔案的指標
85.                         FILE * fp = fopen("HW2.txt","r");
86.                         // 若 fp 回傳為等於 NULL，即為找不到此檔
87.                         if(fp == NULL)
88.                         {
89.                             printf("File:HW2.txt Not Found\n");
90.                         }
91.                         else
92.                         {
93.                             // 將 buffer 清空
94.                             bzero(buffer, 1024);
95.                             // 宣告一個整數型態的變數，用來接收 fread 回傳值
96.                             int file_block_length = 0;
97.                             // fread 的回傳值為成功有效的讀取 fp 指向的檔案的元
    素個數
98.                             while( (file_block_length =
    fread(buffer,sizeof(char),1024,fp))>0)
99.                             {
100.
    if(send(newSocket,buffer,file_block_length,0)<0)
101.                                {
102.                                    // 若元素個數至少有一個即開始傳送
103.                                    // 發送 buffer 中的字串至 newSocket
104.                                    // 若將 buffer 的東西傳送出去回傳的數值小
```

```
    於 0，即發生錯誤
105.                                    printf("Send File:HW2.txt Failed\n");
106.                                    break;
107.                                }
108.                                bzero(buffer, 1024);
109.                            }
110.                            // 關閉指向要寫入的檔案的檔案指標
111.                            fclose(fp);
112.                            printf("File:HW2 Transfer Finished\n");
113.                            break;
114.                        }
115.                    }
116.                }
117.            }
118.        }
119.        // 關閉用來與該 client 溝通及處理的 socket
120.        close(newSocket);
121.        return 0;
122.        }
```

b. Client 端

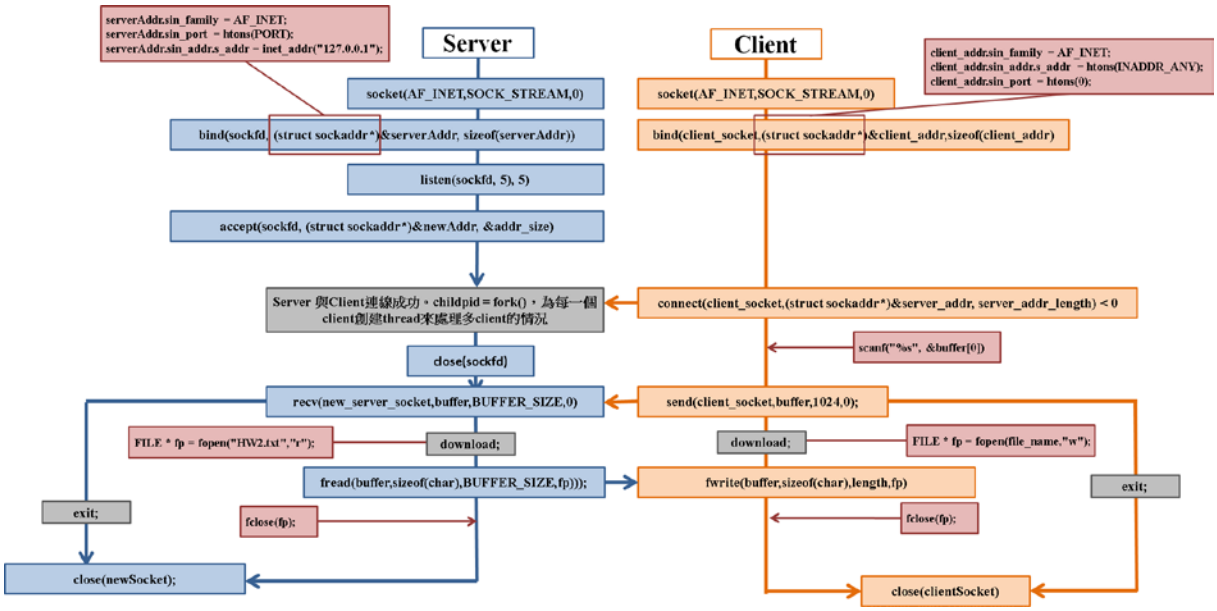
```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <string.h>
4.  #include <unistd.h>
5.  #include <sys/socket.h>
6.  #include <sys/types.h>
7.  #include <netinet/in.h>
8.  #include <arpa/inet.h>
9.
10. #define PORT 4444
11. #define HELLO_WORLD_SERVER_PORT       6666
12. #define BUFFER_SIZE 1024
13. #define FILE_NAME_MAX_SIZE 512
14.
15. int main(){
16.
17.         int clientSocket, ret;
```

```
18.         // 宣告一個 serverAddr 來設定 serversocket 的資訊
19.         struct sockaddr_in serverAddr;
20.         // 宣告一個大小為 1024 的陣列 buffer
21.         char buffer[1024];
22.         // Client 端創建一個 socket
23.         clientSocket = socket(AF_INET, SOCK_STREAM, 0);
24.         // 若創建的回傳值小於 0 即發生錯誤
25.         if(clientSocket < 0){
26.             printf("[-]Error in connection.\n");
27.             exit(1);
28.         }
29.         printf("[+]Client Socket is created.\n");
30.         // 清空 serverAddr
31.         memset(&serverAddr, '\0', sizeof(serverAddr));
32.         // 設定成 IPv4
33.         serverAddr.sin_family = AF_INET;
34.         // 將要連線的 server port number 設定成 4444
35.         serverAddr.sin_port = htons(PORT);
36.         // 將要連線的 server IP address 設定成 loopback address 127.0.0.1
37.         serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
38.         // Client 端發起連線至 Server 端
39.         ret = connect(clientSocket, (struct sockaddr*)&serverAddr,
    sizeof(serverAddr));
40.         // 若 connect 的回傳值小於 0 即發生錯誤
41.         if(ret < 0){
42.             printf("[-]Error in connection.\n");
43.             exit(1);
44.         }
45.         printf("[+]Connected to Server.\n");
46.         printf("Client: \t");
47.         // Client 端輸入要執行的動作，此輸入放進 buffer 中
48.         scanf("%s", &buffer[0]);
49.         // 將 buffer 的內容透過 clientSocket 送出去
50.         send(clientSocket, buffer, strlen(buffer), 0);
51.         // 若使用者輸入的是 exit 的話就直接斷掉與 server 的連線
52.         if(strcmp(buffer, "exit;") == 0){
53.             // 關閉 client 端的 socket
54.             close(clientSocket);
```

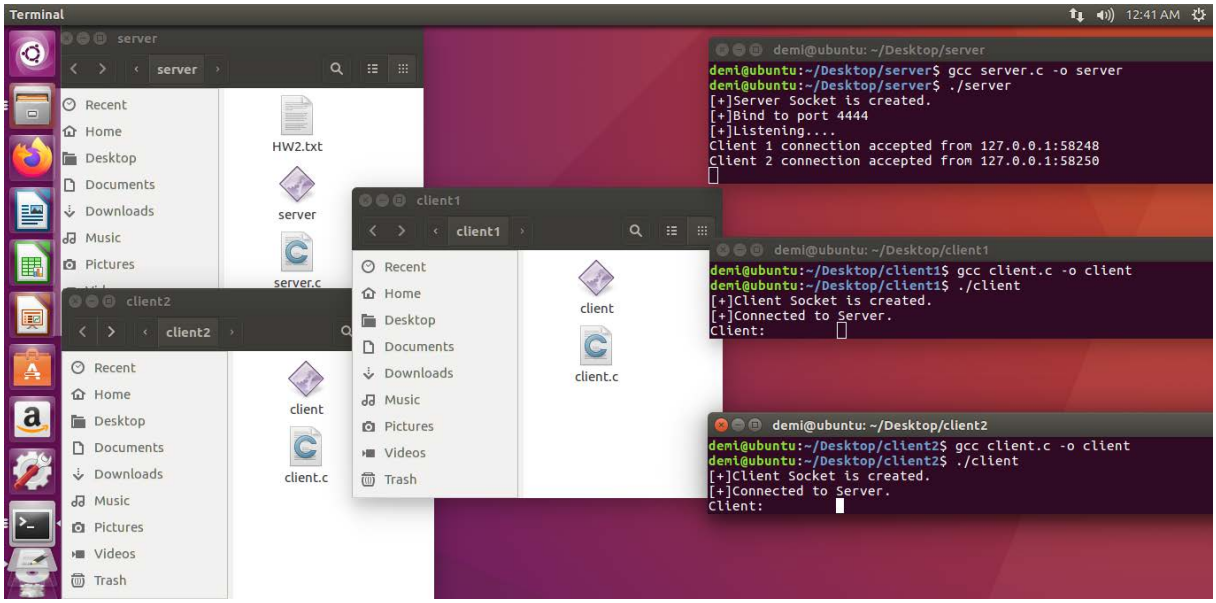
```
55.             printf("[-]Disconnected from server.\n");
56.             // 結束應用程式
57.             exit(1);
58.         }
59.         // 若使用者輸入的是 download;，則開始下載檔案
60.         if(strcmp(buffer, "download;") == 0){
61.             printf("Downloading.....\n");
62.             // 創建指向要寫入的檔案的指標，並創建此檔案
63.             FILE * fp = fopen("HW2.txt","w");
64.             // 若此 fp 為 NULL，即發生錯誤，無法打開檔案寫入
65.             if(fp == NULL)
66.             {
67.                 printf("File:HW2.txt Can Not Open To Write\n");
68.                 exit(1);
69.             }
70.             // 將 buffer 清空
71.             bzero(buffer,1024);
72.             // 創建存放 recv 回傳值的變數
73.             int length = 0;
74.             // 透過 clientSocket 從獲取資料放進 buffer 中，並回傳收到的位元組數
75.             while( length = recv(clientSocket,buffer,1024,0))
76.             {
77.                 // 若收到的位元組數小於 0 即發生錯誤
78.                 if(length < 0)
79.                 {
80.                     printf("Recieve Data From Server Failed!\n");
81.                     break;
82.                 }
83.                 // fwrite 的回傳值為成功有效的讀取 fp 指向的檔案的元素個數
84.                 int write_length = fwrite(buffer,sizeof(char),length,fp);
85.                 // 若 fwrite 回傳值小於從 socket 取得的資料數量即發生錯誤
86.                 if (write_length<length)
87.                 {
88.                     printf("File:HW2.txt Write Failed\n");
89.                     break;
90.                 }
91.                 // 清空 buffer
92.                 bzero(buffer,1024);
```

```
93.             }
94.             printf("Recieve File:HW2.txt From Server Finished\n");
95.             // 關閉指向要寫入的檔案的指標
96.             fclose(fp);
97.             printf("[-]Disconnected from server.\n");
98.             // 關閉 client 端 socket
99.             close(clientSocket);
100.            // 結束應用程式
101.            exit(1);
102.        }
103.        return 0;
104. }
```

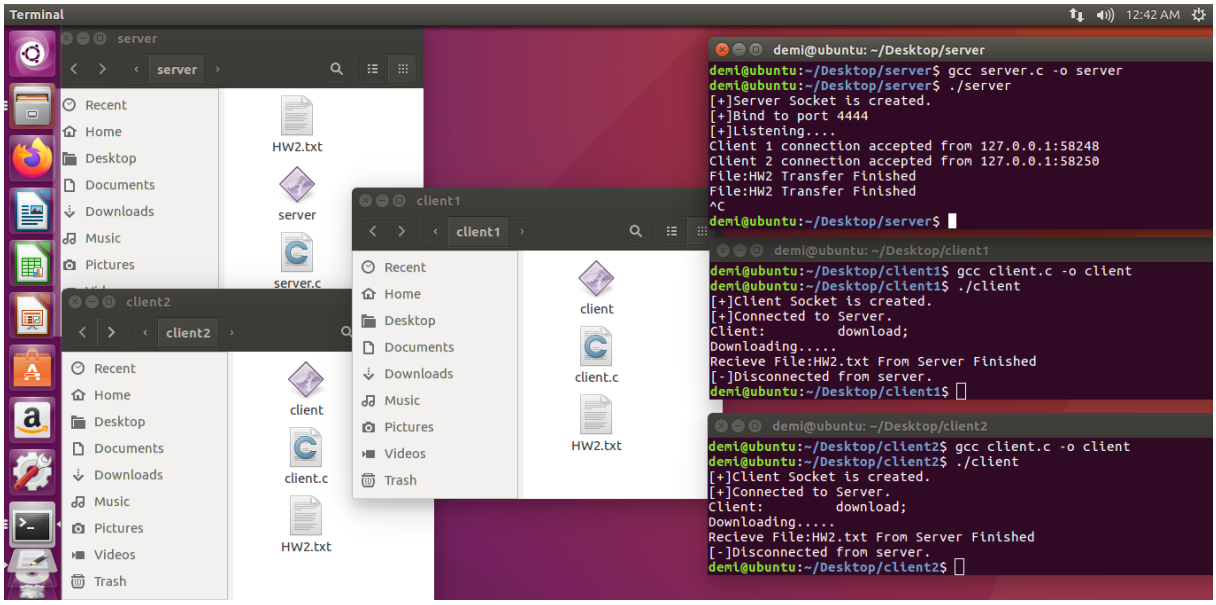
(2) 架構流程圖



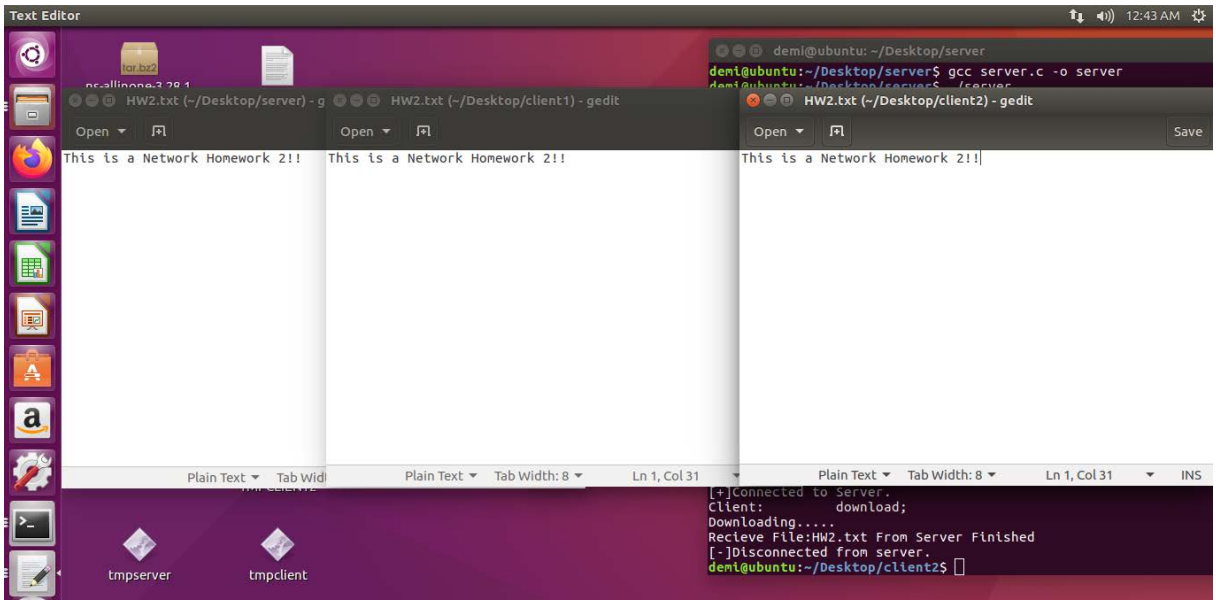
(3) 實驗過程、內容及結果



圖(1)：我將 server 及 client1、client2 分至於三個資料夾，模擬一個 server 和多個 client。並開啟三個終端機各自編譯後執行。另外在 server 的資料夾中有準備讓 client 端下載的檔案 HW2.txt，且在 client 端還未下載的時候其並未有此檔。



圖(2)：clinet1 及 client2 連至 server，此 server 利用不同的 port number 可以辨識不同的 client 端。而 client1 和 client2 分別輸入 download;來下載檔案，將 server 按 ctrl+c 關閉之後，兩個 client 端成功下載檔案。下載完畢後可以發現 client1 和 client2 的資料夾內各自多了一個下載下來的 HW2.txt



圖(3)：分別打開 server、client1、client2 的 HW2.txt 檔案，檔案內容相同，下載成功！