

Readme

HW3

網路程式設計 第七組 組員：

許晉偉 610415145

傅智偉 611415163

彭璟文 612430028

劉冠宏 612430055

程式檔案：

server:

include:

```
8  #include <sys/types.h>
9  #include <sys/socket.h>
10 #include <netinet/in.h>
11 #include <arpa/inet.h>
12 #include <netdb.h>
13 #include <stdio.h>
14 #include <stdlib.h>
15 #include <string.h>
16 #include <unistd.h>
```

define:

```
18 #define BUFFER_SIZE 1024 /*字串最大長度*/
```

variable:

```

20 int main(int argc, char **argv)
21 {
22     int sock, length, fromlen, retval;
23     struct sockaddr_in srvaddr, fromaddr ;
24
25     char fromip[40] ;
26     int fromport ;
27
28     char buffer[BUFFER_SIZE];
29     char echobuf[BUFFER_SIZE] ;
30

```

建立socket:

```

36     sock = socket(PF_INET, SOCK_DGRAM, 0); // 創建一個UDP Socket

```

bind:

```

1 // 如果綁定Socket失敗，輸出錯誤訊息並結束程式
if (bind(sock, (struct sockaddr *)&srvaddr, length) < 0) {
    fprintf(stderr, "Socket Bind 錯誤,無法連結 Port : %d", atoi(argv[1]));
    exit(1);
}

```

recvfrom:

```

retval = recvfrom(sock, buffer, BUFFER_SIZE, 0,
    (struct sockaddr *)&fromaddr, &fromlen); // 接收UDP 訊息
if (retval < 0) {
    fprintf(stderr, "recvfrom function 發生錯誤,無法接收UDP 訊息!!\n"); // 如果接收訊息失敗，輸出錯誤訊息並結束程式
    exit(1);
}

```

sendto:

```

64     bzero(&fromip, 20); // 清空來源IP字串
65     strcpy(fromip, inet_ntoa(fromaddr.sin_addr)); // 獲取來源IP位址
66     fromport = ntohs(fromaddr.sin_port); // 獲取來源通訊埠
67     fprintf(stdout, "接收 UDP 訊息: %s [來至 %s:%d]\n", buffer, fromip, fromport); // 輸出接收到的UDP訊息及來源IP和通訊埠
68
69     bzero(&echobuf, BUFFER_SIZE); // 清空回應緩衝區
70     strcpy(echobuf, buffer); // 複製接收到的訊息到 echobuf
71     retval = sendto(sock, echobuf, BUFFER_SIZE,
72         0, (struct sockaddr *)&fromaddr, fromlen); // 傳送回應訊息
73     if (retval < 0) {
74         fprintf(stderr, "Sendto function 發生錯誤,無法傳送UDP 訊息!!\n"); // 如果傳送回應訊息失敗，輸出錯誤訊息並結束程式
75         exit(1);
76     }
77     fprintf(stdout, "回應 UDP 訊息: %s [送至 %s:%d]\n", echobuf, fromip, fromport); // 輸出已回應的UDP訊息及目的IP和通訊埠
78
79 }
80 close(sock); // 關閉Socket
81

```

client:

include:

```
8  #include <sys/types.h>
9  #include <sys/socket.h>
10 #include <netinet/in.h>
11 #include <arpa/inet.h>
12 #include <netdb.h>
13 #include <stdio.h>
14 #include <stdlib.h>
15 #include <string.h>
16 #include <unistd.h>
```

define:

```
18 #define BUFFER_SIZE 1024 /*字串最大長度*/
```

variable:

```
21 int main(int argc, char **argv)
22 {
23     int sockfd, length, retval;
24     struct sockaddr_in srvaddr, peeraddr;
25
26     char buffer[BUFFER_SIZE], echobuffer[BUFFER_SIZE];
27
28     char peerip[20];
29     int peerport;
30 }
```

建立socket:

```
36 sockfd = socket(PF_INET, SOCK_DGRAM, 0); // 創建一個UDP Socket
```

寄送和接收資料:

```

53 for(;;) { // 無限迴圈
54     fprintf(stdout, "請輸入UDP 訊息 (Enter鍵結束!) (輸入exit終止程式!); "); // 提示用戶輸入UDP 訊息
55     bzero(buffer, BUFFER_SIZE);
56     fgets(buffer, BUFFER_SIZE, stdin);
57     buffer[strlen(buffer) - 1] = '\0'; // 移除換行符
58     if (strcmp(buffer, "exit") == 0)
59         break; // 如果輸入 "exit" 則結束程式
60
61     retval = sendto(sockfd, buffer, strlen(buffer), 0,
62         (struct sockaddr *)&servaddr, length); // 傳送UDP 訊息到伺服器
63     if (retval < 0) {
64         fprintf(stderr, "sendto function 發生錯誤,無法送出 UDP 訊息!!\n"); // 如果傳送訊息失敗,輸出錯誤訊息並結束程式
65         exit(1);
66     }
67     fprintf(stdout, "送出 UDP 訊息: %s [送至 %s:%d]\n", buffer, argv[1], atoi(argv[2])); // 輸出已傳送的UDP 訊息及目的IP和通訊埠
68
69     bzero(&echobuffer, BUFFER_SIZE); // 清空回應緩衝區
70     retval = recvfrom(sockfd, echobuffer, BUFFER_SIZE, 0,
71         (struct sockaddr *)&peeraddr, &length); // 接收從伺服器回傳的UDP 訊息
72     if (retval < 0) {
73         fprintf(stderr, "recvfrom function 發生錯誤,無發接收 UDP 訊息!!\n"); // 如果接收訊息失敗,輸出錯誤訊息並結束程式
74         exit(1);
75     }
76
77     bzero(&peerip, 20);
78     strcpy(peerip, inet_ntoa(peeraddr.sin_addr)); // 獲取來源IP位址
79     peerport = ntohs(peeraddr.sin_port);
80     fprintf(stdout, "收到 UDP 訊息: %s [來至 %s:%d]\n", echobuffer, peerip, peerport); // 輸出已接收的UDP 訊息及來源IP和通訊埠
81 }
82 close(sockfd); // 關閉Socket
83 exit(0); // 正常結束程式

```

程式執行：

server:

```

cody@cody-VirtualBox: ~/桌面
cody@cody-VirtualBox:~/桌面$ ./Group7_UDPServer 10000
wait UDP message...
receive UDP msg: hi [from 127.0.0.1:52660]
reply UDP msg: hi [send to 127.0.0.1:52660]
wait UDP message...

```

client:

```

cody@cody-VirtualBox: ~/桌面
cody@cody-VirtualBox:~/桌面$ ./Group7_UDPClient 127.0.0.1 10000
enter UDP msg (Enter -> done!): hi
Send UDP msg: hi [to 127.0.0.1:10000]
Receive UDP msg: hi [from 127.0.0.1:10000]
cody@cody-VirtualBox:~/桌面$

```

利用Wireshark抓到UDP的封包

The screenshot displays a network capture in Wireshark and two terminal windows. The Wireshark window at the top shows a filter 'udp.port == 10000' and a list of two captured UDP packets. The first packet (No. 192) is from 127.0.0.1 to 127.0.0.1 on port 52660, with a length of 46 bytes. The second packet (No. 193) is from 127.0.0.1 to 127.0.0.1 on port 10000, with a length of 1068 bytes. Below the Wireshark window are two terminal windows. The left terminal window shows the execution of './Group7_UDPServer 10000', which waits for a message, receives 'hi' from 127.0.0.1:52660, and replies 'hi' to 127.0.0.1:52660. The right terminal window shows the execution of './Group7_UDPClient 127.0.0.1 10000', which enters 'hi', sends it to 127.0.0.1:10000, and receives 'hi' from 127.0.0.1:10000.

No.	Time	Source	Destination	Protocol	Length	Info
192	31.799053736	127.0.0.1	127.0.0.1	UDP	46	52660 → 10000 Len=2
193	31.799092467	127.0.0.1	127.0.0.1	UDP	1068	10000 → 52660 Len=1024

```
cody@cody-VirtualBox: ~/桌面
cody@cody-VirtualBox:~/桌面$ ./Group7_UDPServer 10000
wait UDP message...
receive UDP msg: hi [from 127.0.0.1:52660]
reply UDP msg: hi [send to 127.0.0.1:52660]
wait UDP message...
cody@cody-VirtualBox:~/桌面$

cody@cody-VirtualBox:~/桌面$ ./Group7_UDPClient 127.0.0.1 10000
enter UDP msg (Enter -> done!): hi
Send UDP msg: hi [to 127.0.0.1:10000]
Receive UDP msg: hi [from 127.0.0.1:10000]
cody@cody-VirtualBox:~/桌面$
```