第七章　Socket 通訊

此文件分成(1)客戶端傳送伺服端接收(2) 客戶端接收伺服端傳送兩部份

(1)客戶端傳送伺服端接收

首先將 socketsend-dirfiles 資料夾拷貝至/root 目錄。在 socketsend-dirfiles 資料夾內有 clientsenddir 及 serversenddir 兩個目錄。在 clientsenddir 目錄的程式架構下，客戶端將指定資料夾內所有檔案，透過網路線傳至伺服端。伺服端接收到檔案後，存至指定資料夾。clientsenddir 目錄的程式如圖 7-1 所示，客戶端原始程式是 client-txdir.c，以及 numbers.txt。伺服端原始程式包括 server-rxdir.c、socket.h。(伺服端，客戶端)皆可在個人電腦或嵌入式系統執行。伺服端程式可接受多重客戶端的連線要求。

圖 7-1. 客戶端傳送伺服端接收的程式資料夾

在編譯可在嵌入式系統端執行前，需有正確的編譯器路徑的設定:

~:#PATH=$PATH:/usr/local/arm/3.4.1/bin

或者將此指令寫至/etc/profile 檔案，然後執行

~#source /etc/profile

接下來須決定個人電腦端及嵌入式系統端如何執行(伺服端，客戶端)程式。有兩種選擇

1. 個人電腦當伺服端，嵌入式系統當客戶端:

切換至**/root/socketsend-dirfiles/clientsenddir** 目錄，編譯個人電腦所執行的伺服端程式:

~/socketsend-dirfiles/clientsenddir#**gcc –o server-rxdir server-rxdir.c**

編譯成功所得的執行檔是 server-rxdir。記得規劃電腦端的 IP:

~/socketsend-dirfiles/clientsenddir#**ifconfig eth0 192.168.1.20**

然後執行伺服端程式:

~/socketsend-dirfiles/clientsenddir**#./server-rxdir**

編譯嵌入式系統所執行的客戶端程式:

~:#arm-linux-gcc –o client-txdir client-txdir.c

編譯成功所得的執行檔是 client-txdir。將 number.txt 文字檔、dhcpall 草稿檔，以及 client-txdir 拷貝至隨身碟的 programs 資料夾內。270l 開機後自動執行的 dhcpall 草稿檔如圖 7-2 所示。在 dhcpall 草稿檔規劃 270l 的 IP 為 192.168.1.30，然後自動執行 client-txdir 程式。執行 client-txdir 程式時，會自動讀取 number.txt 檔內的字串。由圖 3 內容可知，指定伺服端的 IP 為 192.168.1.20，埠號為 9990。按照目前 client-txdir 的規劃，客戶端將隨身碟內 pictures 目錄下的所有檔案，透過網路線傳至伺服端。伺服端接收這些檔案後，存至/tmp 目錄。

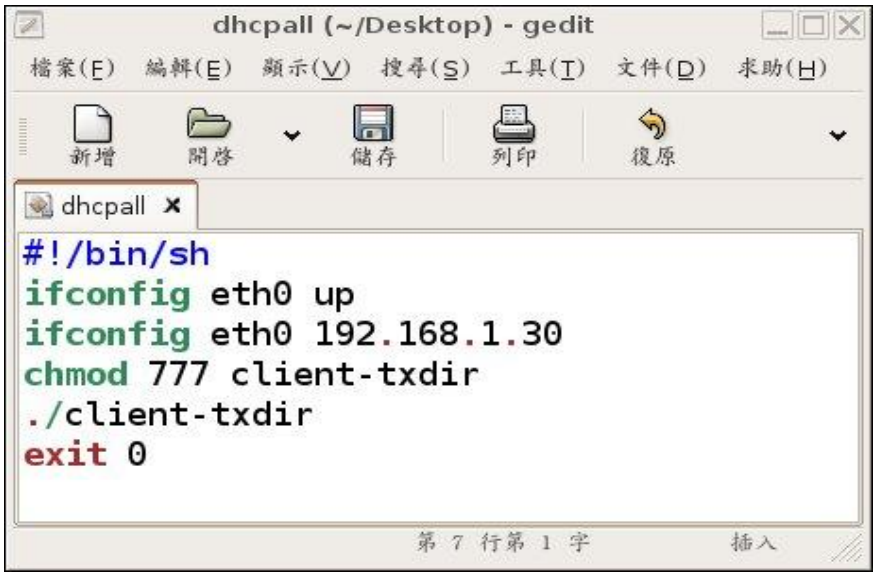


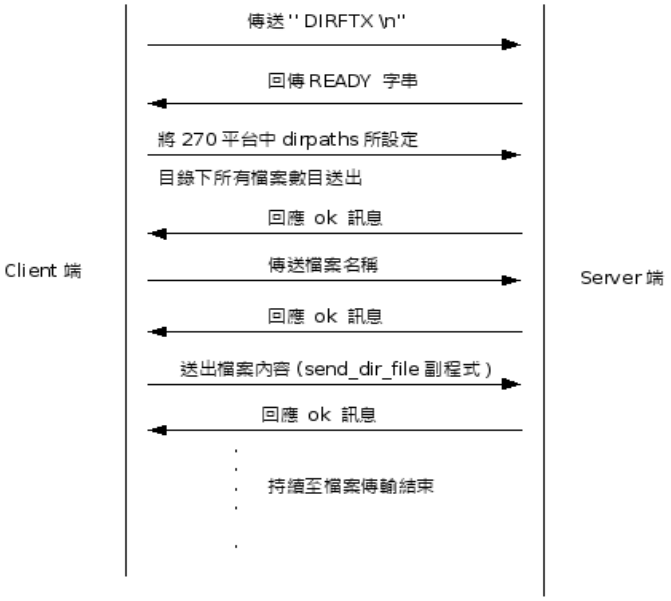
圖 7-2. dhcpall 草稿檔的內容

圖 7-3. number.txt 檔的內容

寫Socket程式的重要觀念

Socket 用檔案描述元(file descriptor) 做 Internet 上的資料傳送與接收。為了要在 Internet 上傳送資料，因此 socket 必須要遵守 TCP/IP 網路通訊協定，而在寫程式時所有與硬體有關的部分都是交給 API (應用程式介面)。 API 可以簡單看成就是一組組的函式，只要照著函式的格式加參數及寫變數，就可以達到與硬體溝通。Socket 依照作業系統有 windows 下的 winsock API 與 unix-like 的 sock API，兩者基本的概念與流程都一樣，差別在語法的不同。

Ethernet 是 TCP/IP 通訊協定的其中一種。 欲透過 Ethernet 作資料的對傳，分成 Clent 端及 Server 端兩種情況。就 Client 端而言，相關動作為
1. 建立 socket 功能 。2. 填寫連線遠端 server 的資料，包含連線協定(ipv4，ipv6，TCP ，UDP)及 server ip，server port 。3. 提出連線要求。4. 傳送資料或接收資料。5. 結束 socket 連線。就 Server 端而言，相關動作為 1. 建立 socket 功能。2. 確定自己的資料 (ip，port)。3. 聆聽 等待連接。4. 接受連接。5. 傳送資料或接收資料 6. 結束 socket 連線。接下來分別針對這些動作說明。傳輸動作如
下圖



Client 端的程式

```
file_fd = open("numbers.txt",O_RDONLY);
memset(bufd,0,sizeof(bufd));
nread = read(file_fd,bufd,sizeof(bufd));
close(file_fd);
pd=strtok(bufd, "\n");
```

```
strcpy(S_ip, pd);    // Read assigned server IP
printf("Server IP are: %s \n", S_ip);
pd = strtok(NULL,"\n");
strcpy(S_port, pd); // Read assigned server port number
        PORT= atoi(S_port);
        printf("Server PORT are: %d \n", PORT);
        strcpy(comstr,"DIRFTX\n");
        printf(" command string are : %s", comstr);
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if(sockfd == -1){
         perror("socket ");
         exit(1);
    }
    bzero(&addr,sizeof(addr));
    addr.sin_family = AF_INET;
    addr.sin_addr.s_addr = inet_addr(S_ip);
    addr.sin_port = htons(PORT);
    nread = connect(sockfd, (void *)&addr, sizeof(addr));
    if(nread == -1){
         perror("connect ");
         exit(1);
    }
      signal(SIGALRM, handler);      // handler is alarm service routine
      alarm(0);
    write(sockfd,comstr,7); // send command
      nread=read_str5(sockfd,5); //wait two seconds to receive
    send_dir_files(sockfd);
```

說明如下:

1.依據 number.txt 讀取設定 SERVER IP 與 Port number

先將 number.txt 資料讀取出，將資料設定為 IP 與 PORT。

A. 將 number.txt 設為讀取型態。

B. 將 S_ip 設定為 number.txt 中所設定之 IP。

C. 將 S_port 設定為 number.txt 中所設定之 PORT。

範例：

```
file_fd = open("numbers.txt",O_RDONLY);
// 將 nubmer.txt 開啟，並設為唯讀型態
        memset(bufd,0,sizeof(bufd));
//   將 bufd 陣列中所有資料設定為 null (清空陣列）
        nread = read(file_fd,bufd,sizeof(bufd));
    /*
```

```
   nread  為實際讀取字元數
            read 讀取檔案描述子當中資料
             file_fd  使用者自訂檔案描述子之名稱，代表以開啟檔案或硬體功能
            bufd  預存放陣列
            sizeof(bufd) bufd 陣列當中的位元數*/
          close(file_fd);
        // 關閉 file_df  檔案
          pd=strtok(bufd, "\n");
     // 讀取 \n (換行符號）前之字串 （讀取到 number.txt 當中第一行）存至 pd
          strcpy(S_ip, pd);   // 將讀取到資訊存入 S_ip
          printf("Server IP are: %s \n", S_ip);
 //印出所設定之 IP
       pd = strtok(NULL,"\n");
 //再次讀取 \n 前之字串，此時會讀取至第二行存至 pd
          strcpy(S_port, pd); // 將讀取到資訊存入 S_port
          PORT= atoi(S_port);
// 將字串轉為整數型態 (ASCII → Integer）
      printf("Server PORT are: %d \n", PORT);
// 印出所設定之 PORT
          strcpy(comstr,"DIRFTX\n");
   //將 DIRFTX 字串存入 comstr 變數中
       printf(" command string are : %s", comstr); // 印出 comstr 資訊
```

2.建立 Socket 功能

函式 int socket(int domain , int type , int protocol)

A. 參數 domain：要使用的規範 IPV4 的規範 是 AF_INET 或者是 PF_INET 兩者皆表示同一規範

B. 參數 type：傳輸資料所使用的協定 TCP ，所以使用 SOCK_STREAM 此參數。

C. 參數 protocol：網路位元組順序的 32 位元整數，大部分連線型態都只支援 protocol=0。

若回傳值為-1 表示 socket 建立失敗，其他則表示成功。

範例：

```
sockfd = socket(AF_INET, SOCK_STREAM, 0);
      if(sockfd == -1)
      {
      perror("socket ");   /*socket 建立失敗，在終端機畫面顯示 Socket 相關的錯誤的訊息*/
      exit(1);
      }
```

3. 填寫遠端 Server 資料

建立 Socket 連線後，需填寫的遠端 Server 資料，包括：

宣告儲存 Server 資料的結構： struct sockaddr_in dest。dest 是資料結構變數的型態，包括

1. 填寫 Server 使用的規範：dest.sin_family = AF_INET。
2. 將 16 位元的 2 進位碼轉換成從大到小的排序：dest.sin_port = htons(PORT)
3. 將使用句點分格式(xxx.xxx.xxx.xxx)的位址轉換成網路位元組順序的 2 進位碼：
   inet_aton(SERVER_ADDR, &dest.sin_addr.s_addr)

相關程式範例：

……

```
#define PORT          9999              /*要連接到 server  的  port */
#define S_ip          "10.15.1.94"      /*server  的 ip address   */
/*  填寫連線遠端  server  的資料     */
bzero(&addr,sizeof(addr));   /*將 Struct 記憶體內的值初始化為零*/
addr.sin_family = AF_INET;
/*將使用句點分格式的位址 S_ip  轉換成網路位元組順序的 2 進位碼，然後存於
addr.sin_addr.s_addr。*/
addr.sin_addr.s_addr = inet_addr(S_ip);
/*  將 PORT 號轉換成二進制碼，存放於 addr.sin_port */
addr.sin_port = htons(PORT);
```

4. 要求連線

填寫遠端 Server 資料後，Client 端便可透過 connect 指令對 Server 端提出連線要求，指令格式說明如下：

int connect(int sockfd , const struct sockaddr * serv_addr ,socklen_t addrlen);

A. 參數 sockfd：建立 socket 時的檔案描述元

B. 參數 serv_addr：填寫 Server 資料結構的變數

C.參數 addrlen：Server 資料結構的大小，

範例：

```
nread = connect(sockfd, (void *)&addr, sizeof(addr));
    if(nread == -1)
    {
    perror("connect "); /*若連線失敗，在終端機畫面顯示 Connect 相關的錯誤的訊息*/
    exit(1);
    }
```

5. 資料收發與結束連線

連線成功後，即可用 read 及 write 傳送資料，範例如下:

```
write(sockfd,comstr,7); //  傳送字串，字元數為 7。
nread=read(file_fd,sbuf,sizeof(sbuf))   ; //接收字元數為 sizeof(sbuf)的字串，若接受到換行符號
```

則強迫脫離 read 的執行。
當傳輸資料或接收資料完畢後，便可結束 socket 連線指令格式說明如下：
int close(int sd)

參數 sd：建立 socket 時的檔案描述元

send_dir_files(sockfd) // 將 sockfd 送至 send_dir_files 模組等待傳送

/* 結束 SOCKET 連線*/
close(sockfd);

5. send_dir_files 模組說明

```
strcpy(dirpaths,"/mnt/udisk/pictures/");
// 將/mnt/udisk/pictures 存至 dirpaths 變數中，此為預設傳送資料夾路徑
total=scandir(dirpaths,&namelist,NULL,0);
//將 dirpaths 變數目錄中檔案數目回傳
printf("total file counts are %d \n",total-2);
//
印出資料夾中檔案數目，需去掉 . 與 .. 兩目錄數
dir=opendir(dirpaths);
// 將 dirpaths 路徑資料夾開啟，存入 dir 變數中
sprintf(xbuf,"%d",total-2);
// 將 total-2 之數值轉換為字串存至 xbuf 中
strcat(xbuf, "\n");   // 將 xbuf 陣列內的字串後串接一換行符號(“\n”)
printf("file count string is ..%s",xbuf);
// 將 xbuf 內容印出
write(client_sockfd,xbuf,strnlen(xbuf));
sread =read_str2(client_sockfd,2); //wait two seconds to receive echo ok
/* fetch files in the specified directory s2 */
if(dir != NULL)
{
        while((ptr=readdir(dir)) != NULL)
{
            if((strncmp(ptr->d_name, ".", 1)) != 0)
{
                strcpy(s3,ptr->d_name);
                dflag=0;
                strcat(s3, "\n");       //send filename
                write(client_sockfd,s3,strnlen(s3));
                sread =read_str2(client_sockfd,2); //wait two seconds to receive echo ok
                printf("file name are %s\n",s3);
                //sread=read_str2(client_sockfd,2); //wait two seconds to receive echo
```

```
                    strcpy(s1,dirpaths);
                    strcat(s1,ptr->d_name);        /* cascade filename */
                    printf("file are %s\n",s1);
                    send_filey(s1,client_sockfd); /* send file */
                        printf("file received ..\n");
                }    /* end if */
            } /* end read dir while */
        closedir(dir);
            } //end dir null
        printf("transmit dir over \n");
```

Server 端的程式

範例程式碼 :

```
int server_len,client_len;
        int server_sockfd, client_sockfd;
        struct sockaddr_in server_address;
        struct sockaddr_in client_address;
    int fd,nread,i;
    int result; //select()
    server_sockfd=socket(AF_INET,SOCK_STREAM,0);
    server_address.sin_family=AF_INET;
    server_address.sin_addr.s_addr=htonl(INADDR_ANY);
    server_address.sin_port=htons(9990);
    server_len=sizeof(server_address);
  if( bind(server_sockfd, (struct sockaddr *)&server_address, server_len) == -1){
            perror("bind :");
            exit(1);
    }
    printf("bind OK\n");
       if( (listen(server_sockfd,50)) == -1){
            perror("Listen :");
            exit(1);
    }
```

1. 建立 Socket 連線

函式 int socket(int domain , int type , int protocol)

A. 參數 domain：要使用的規範 IPV4 的規範 是 AF_INET 或者是 PF_INET 兩者皆表示同一規範

B. 參數 type：傳輸資料所使用的協定 TCP ，所以使用 SOCK_STREAM 此參數。

C. 參數 protocol：網路位元組順序的 32 位元整數，大部分連線型態都只支援 protocol=0。若回傳值為-1 表示 socket 建立失敗，其他則表示成功。範例：

```
server_sockfd=socket(AF_INET,SOCK_STREAM,0);
if( (server_sockfd == -1)
{
    perror("socket :");
    exit(1);
}
```

2.socket 位址結構

大多數的 socket 函式需要 socket 位址結構的指標作為參數，而支援的每種協定組也都定義了它們自己的 socket 位址結構；這些結構的名稱是以 sockaddr_開頭，而且每個協定組都有個自唯一的結尾。IPv4 的 SOCKET 位址結構 sockaddr_in，通常被稱為「網際網路 socket 位址結構」，並且定義在<netinet/in.h>標頭檔中。

```
server_address.sin_family=AF_INET;              //IPv4 協定
server_address.sin_addr.s_addr=htonl(INADDR_ANY); //允許來自任何 IP 位址的連結
server_address.sin_port=htons(9990);     //將 port 號轉為 16 位元
```

3.Client 與 Server 端的連繫

為了讓 socket 可以被其它處理程序使用，一個伺服器程式需要給予 socket 一個名稱。因此，AF_UNIX socket 結合一個檔案系統名稱，AF_INET socket 則結合一個 IP port 數值

```
server_len = sizeof(ser_addr);
if( bind(server_sockfd, (struct sockaddr *)&ser_addr, server_len) == -1)
{
    perror("bind :");
    exit(1);
}
```

bind()系統呼叫，必須在 address 中指定位址，給予未命名的 socket 檔案描述子定義一個名稱。address_len 則要傳入位止結構長度。bind 成功之後，它會回傳 0。如果失敗，它會回傳 -1。

4.產生一個 socket 佇列

在一個 socket 接收連結請求之前，伺服器程式必須產生一個佇列，這個佇列儲存懸而未決的請求。要做這樣的動作，我們必須使用 listen()系統呼叫：

```
if( (listen(server_sockfd,1)) == -1)       //一次祇允許一個 client 端連線
{
    perror("Listen :");
    exit(1);
```

}

一個 Linux 系統可以限制在佇列中的數量，也就是懸而未決定的連結請求量。為了符合最大量，必須在 listen()中設定佇列長度 backlog。在佇列還有空間的狀況下，socket 還可以接收傳送的連結請求，如果超過佇列長度，後續的客戶端連結請求就會失敗。當伺服器程式在處理先前的客戶時，listen 提供一個機制，讓伺服器在忙碌中，也能接受連結請求。

listen()函數在成功後會回覆 0。失敗後就回傳-1。

5.接受連結

伺服器程式產生和命名 socket 之後，透過 accept()系統呼叫，它會接受這個 socket 的連結請求。

```
client_sockfd = accept(server_sockfd,(struct sockaddr *)&cli_addr, &client_len);
if(client_sockfd == -1)
{
    perror("accept error");
    exit(1);
}
```

當一個客戶端程式試圖連結這個 socket，accept()系統呼叫就會回覆。若客戶端是在 socket 佇列中第一個未進的連結請求。accept()函數產生一個新的 socket 來與這個客戶端溝通，所以會回覆這個 socket 描述子。新的 socket 的型態如同伺服器放在 listen()的 socket 型態一般。

6. 資料收發與結束連線

Server 端接受一個連線要求成功後，即可用 read 及 write 傳送資料，範例如下:

write(sockfd,comstr,7); // 傳送字串，字元數為 7。

nread=read(file_fd,sbuf,sizeof(sbuf)) ; //接收字元數為 sizeof(sbuf)的字串，若接受到換行符號則強迫脫離 read 的執行。

當傳輸資料或接收資料完畢後，便可結束 socket 連線指令格式說明如下：

int close(int sd)

參數 sd：建立 socket 時的檔案描述元

```
/* 結束 SOCKET 連線*/
    close(sockfd);
```

(2)伺服端傳送客戶端接收

首先將 socketsend-dirfiles 資料夾拷貝至/root 目錄。在 socketsend-dirfiles 資料夾內有 clientsenddir 及 serversenddir 兩個目錄。在 serversenddir 目錄的程式架構下，伺服端將指定資料夾內所有檔案，透過網路線傳至客戶端。客戶端接收到檔案後，存至指定資料夾。serversenddir 目錄的程式如圖 7-4 所示，客戶端原始程式是 client-rxdir.c，以及 numbers.txt。伺服端原始程式包括 server-senddir.c、socket.h。(伺服端，客戶端)皆可在個人電腦或嵌入式系統執行。伺服端程式可接受多重客戶端的連線要求。

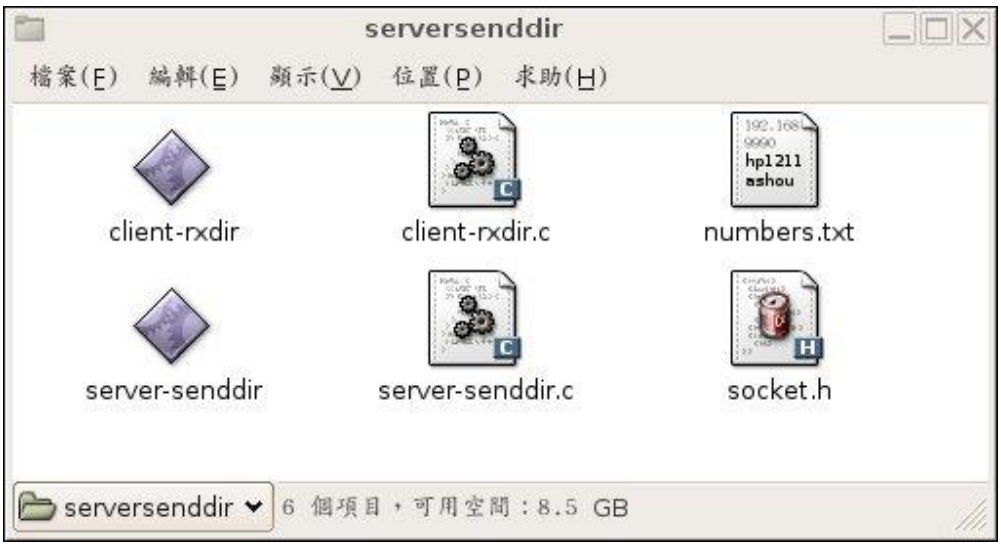


圖 7-4. 伺服端傳送客戶端接收的程式資料夾

在編譯可在嵌入式系統端執行前，需有正確的編譯器路徑的設定:

~:#PATH=$PATH:/usr/local/arm/3.4.1/bin

或者將此指令寫至/etc/profile 檔案，然後執行

~#source /etc/profile

接下來須決定個人電腦端及嵌入式系統端如何執行(伺服端，客戶端)程式。有兩種選擇

1. 個人電腦當伺服端，嵌入式系統當客戶端:

切換至**/root/socketsend-dirfiles/serversenddir** 目錄，編譯個人電腦所執行的伺服端程式:

~/socketsend-dirfiles/serversenddir#**gcc –o server-senddir server-senddir.c**

編譯成功所得的執行檔是 server-senddir。記得規劃電腦端的 IP:

~/socketsend-dirfiles/serversenddir#**ifconfig eth0 192.168.1.20**

然後先執行伺服端程式:

~/socketsend-dirfiles/serversenddir**#./server-senddir**

編譯嵌入式系統所執行的客戶端程式:

~/socketsend-dirfiles/serversenddir**#arm-linux-gcc –o client-rxdir client-rxdir.c**

編譯成功所得的執行檔是 client-rxdir。將 number.txt 文字檔、dhcpall 草稿檔，以及 client-rxdir 拷貝至隨身碟的 programs 資料夾內。270l 開機後自動執行的 dhcpall 草稿檔如圖 7-5 所示。在 dhcpall 草稿檔規劃 270l 的 IP 為 192.168.1.30，然後自動執行 client-rxdir 程式。執行 client-rxdir 程式時，會自動讀取 number.txt 檔內的字串。由圖 7-6 內容可知，指定伺服端的 IP 為 192.168.1.20，埠號為 9990。按照目前 client-txdir 的規劃，伺服端將/root/pictures 目錄下的所有檔案，透過網路線傳至客戶端。客戶端接收這些檔案後，存至/tmp 目錄。

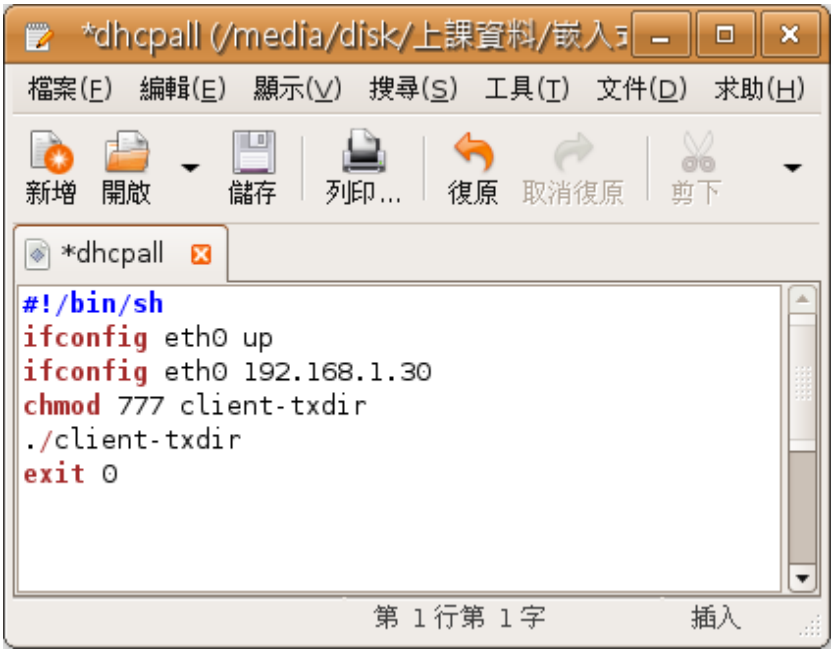


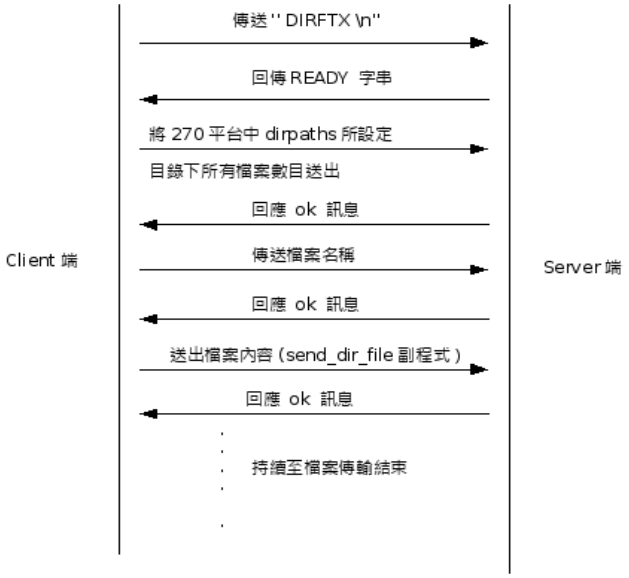
圖 7-5. dhcpall 草稿檔的內容

圖 7-6. number.txt 檔的內容

寫Socket程式的重要觀念

Socket 用檔案描述元(file descriptor) 做 Internet 上的資料傳送與接收。為了要在 Internet 上傳送資料，因此 socket 必須要遵守 TCP/IP 網路通訊協定，而在寫程式時所有與硬體有關的部分都是交給 API (應用程式介面)。 API 可以簡單看成就是一組組的函式，只要照著函式的格式加參數及寫變數，就可以達到與硬體溝通。Socket 依照作業系統有 windows 下的 winsock API 與 unix-like 的 sock API，兩者基本的概念與流程都一樣，差別在語法的不同。

Ethernet 是 TCP/IP 通訊協定的其中一種。 欲透過 Ethernet 作資料的對傳，分成 Clent 端及 Server 端兩種情況。就 Client 端而言，相關動作為
1. 建立 socket 功能 。2. 填寫連線遠端 server 的資料，包含連線協定(ipv4，ipv6，TCP ，UDP)及 server ip，server port 。3. 提出連線要求。4. 傳送資料或接收資料。5. 結束 socket 連線。就 Server 端而言，相關動作為 1. 建立 socket 功能。2. 確定自己的資料 (ip，port)。3. 聆聽 等待連接。4. 接受連接。5. 傳送資料或接收資料 6. 結束 socket 連線。接下來分別針對這些動作說明。傳輸動作如下圖



Client 端的程式

```
file_fd = open("numbers.txt",O_RDONLY);

memset(bufd,0,sizeof(bufd));

nread = read(file_fd,bufd,sizeof(bufd));
```

```
close(file_fd);
pd=strtok(bufd, "\n");
strcpy(S_ip, pd);    // Read assigned server IP
printf("Server IP are: %s \n", S_ip);
pd = strtok(NULL,"\n");
strcpy(S_port, pd); // Read assigned server port number
        PORT= atoi(S_port);
        printf("Server PORT are: %d \n", PORT);
        strcpy(comstr,"DIRFTX\n");
        printf(" command string are : %s", comstr);
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if(sockfd == -1){
          perror("socket ");
          exit(1);
    }
    bzero(&addr,sizeof(addr));
    addr.sin_family = AF_INET;
    addr.sin_addr.s_addr = inet_addr(S_ip);
    addr.sin_port = htons(PORT);
    nread = connect(sockfd, (void *)&addr, sizeof(addr));
    if(nread == -1){
          perror("connect ");
          exit(1);
    }
      signal(SIGALRM, handler);      // handler is alarm service routine
      alarm(0);
    write(sockfd,comstr,7); // send command
      nread=read_str5(sockfd,5); //wait two seconds to receive
    send_dir_files(sockfd);
```

說明如下:

1.依據 number.txt 讀取設定 SERVER IP 與 Port number

先將 number.txt 資料讀取出，將資料設定為 IP 與 PORT。

A. 將 number.txt 設為讀取型態。

B. 將 S_ip 設定為 number.txt 中所設定之 IP。

C. 將 S_port 設定為 number.txt 中所設定之 PORT。

範例：

```
 file_fd = open("numbers.txt",O_RDONLY);
// 將 nubmer.txt 開啟，並設為唯讀型態
      memset(bufd,0,sizeof(bufd));
//   將 bufd 陣列中所有資料設定為 null (清空陣列）
```

```
        nread = read(file_fd,bufd,sizeof(bufd));
    /*
  nread  為實際讀取字元數
           read 讀取檔案描述子當中資料
            file_fd  使用者自訂檔案描述子之名稱，代表以開啟檔案或硬體功能
           bufd  預存放陣列
           sizeof(bufd) bufd 陣列當中的位元數*/
        close(file_fd);
       // 關閉 file_df  檔案
        pd=strtok(bufd, "\n");

    // 讀取 \n (換行符號）前之字串 （讀取到 number.txt 當中第一行）存至 pd
        strcpy(S_ip, pd);   // 將讀取到資訊存入 S_ip
        printf("Server IP are: %s \n", S_ip);
 //印出所設定之 IP
      pd = strtok(NULL,"\n");
 //再次讀取 \n 前之字串，此時會讀取至第二行存至 pd
        strcpy(S_port, pd); // 將讀取到資訊存入 S_port
        PORT= atoi(S_port);
// 將字串轉為整數型態 (ASCII → Integer）
     printf("Server PORT are: %d \n", PORT);
// 印出所設定之 PORT
        strcpy(comstr,"DIRFTX\n");
  //將 DIRFTX 字串存入 comstr 變數中
      printf(" command string are : %s", comstr); // 印出 comstr 資訊
```

2.建立 Socket 功能

函式 int socket(int domain , int type , int protocol)

A. 參數 domain：要使用的規範 IPV4 的規範 是 AF_INET 或者是 PF_INET 兩者皆表示同一規範

B. 參數 type：傳輸資料所使用的協定 TCP ，所以使用 SOCK_STREAM 此參數。其他參數型態請參考文獻[6]。

C. 參數 protocol：網路位元組順序的 32 位元整數 [2, p9, p13]，大部分連線型態都只支援 protocol=0。

若回傳值為-1 表示 socket 建立失敗，其他則表示成功。

範例：

```
sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if(sockfd == -1)
    {
```

```
perror("socket ");   /*socket 建立失敗，在終端機畫面顯示 Socket 相關的錯誤的訊息*/
exit(1);
}
```

3. 填寫遠端 Server 資料

建立 Socket 連線後，需填寫的遠端 Server 資料，包括：

宣告儲存 Server 資料的結構： struct sockaddr_in dest。dest 是資料結構變數的型態，包括

4. 填寫 Server 使用的規範：dest.sin_family = AF_INET。
5. 將 16 位元的 2 進位碼轉換成從大到小的排序：dest.sin_port = htons(PORT)
6. 將使用句點分格式(xxx.xxx.xxx.xxx)的位址轉換成網路位元組順序的 2 進位碼：
   inet_aton(SERVER_ADDR, &dest.sin_addr.s_addr)

相關程式範例：

```
……
#define PORT          9999              /*要連接到 server 的 port */
#define S_ip          "10.15.1.94"      /*server 的 ip address   */
/*  填寫連線遠端 server 的資料     */
bzero(&addr,sizeof(addr));   /*將 Struct 記憶體內的值初始化為零*/
addr.sin_family = AF_INET;
/*將使用句點分格式的位址 S_ip 轉換成網路位元組順序的 2 進位碼，然後存於
addr.sin_addr.s_addr。*/
addr.sin_addr.s_addr = inet_addr(S_ip);
/*  將 PORT 號轉換成二進制碼，存放於 addr.sin_port */
addr.sin_port = htons(PORT);
```

4. 要求連線

填寫遠端 Server 資料後，Client 端便可透過 connect 指令對 Server 端提出連線要求，指令格式說明如下：

int connect(int sockfd , const struct sockaddr * serv_addr ,socklen_t addrlen);

A. 參數 sockfd：建立 socket 時的檔案描述元

B. 參數 serv_addr：填寫 Server 資料結構的變數

C.參數 addrlen：Server 資料結構的大小，

範例：

```
nread = connect(sockfd, (void *)&addr, sizeof(addr));
    if(nread == -1)
    {
    perror("connect "); /*若連線失敗，在終端機畫面顯示 Connect 相關的錯誤的訊息*/
```

```
        exit(1);
        }
```

5. 資料收發與結束連線

連線成功後，即可用 read 及 write 傳送資料，範例如下:

write(sockfd,comstr,7); // 傳送字串，字元數為 7。

nread=read(file_fd,sbuf,sizeof(sbuf)) ; //接收字元數為 sizeof(sbuf)的字串，若接受到換行符號則強迫脫離 read 的執行。

當傳輸資料或接收資料完畢後，便可結束 socket 連線指令格式說明如下：

int close(int sd)

參數 sd：建立 socket 時的檔案描述元

send_dir_files(sockfd) // 將 sockfd 送至 send_dir_files 模組等待傳送

```
/* 結束 SOCKET 連線*/
            close(sockfd);
```

5. send_dir_files 模組說明

```
        strcpy(dirpaths,"/mnt/udisk/pictures/");

        // 將/mnt/udisk/pictures 存至 dirpaths 變數中，此為預設傳送資料夾路徑
        total=scandir(dirpaths,&namelist,NULL,0);
 //將 dirpaths 變數目錄中檔案數目回傳
        printf("total file counts are %d \n",total-2);
        //
印出資料夾中檔案數目，需去掉 . 與 .. 兩目錄數
        dir=opendir(dirpaths);
   // 將 dirpaths 路徑資料夾開啟，存入 dir 變數中
        sprintf(xbuf,"%d",total-2);
 // 將 total-2 之數值轉換為字串存至 xbuf 中
        strcat(xbuf, "\n");   // 將 xbuf 陣列內的字串後串接一換行符號(“\n”)
        printf("file count string is ..%s",xbuf);
 // 將 xbuf 內容印出
        write(client_sockfd,xbuf,strnlen(xbuf));
        sread =read_str2(client_sockfd,2); //wait two seconds to receive echo ok
        /* fetch files in the specified directory s2 */
        if(dir != NULL)
{
              while((ptr=readdir(dir)) != NULL){
                    if((strncmp(ptr->d_name, ".", 1)) != 0){
```

```
                    strcpy(s3,ptr->d_name);
                    dflag=0;
                    strcat(s3, "\n");       //send filename
                    write(client_sockfd,s3,strnlen(s3));
                    sread =read_str2(client_sockfd,2); //wait two seconds to receive echo ok
                    printf("file name are %s\n",s3);
                    //sread=read_str2(client_sockfd,2); //wait two seconds to receive echo
                    strcpy(s1,dirpaths);
                    strcat(s1,ptr->d_name);       /* cascade filename */
                    printf("file are %s\n",s1);
                    send_filey(s1,client_sockfd); /* send file */
                        printf("file received ..\n");
                }    /* end if */
            } /* end read dir while */
        closedir(dir);
        } //end dir null
        printf("transmit dir over \n");
```

Server 端的程式

範例程式碼 :

```
int server_len,client_len;
        int server_sockfd, client_sockfd;
        struct sockaddr_in server_address;
        struct sockaddr_in client_address;
    int fd,nread,i;
    int result; //select()
    server_sockfd=socket(AF_INET,SOCK_STREAM,0);
    server_address.sin_family=AF_INET;
    server_address.sin_addr.s_addr=htonl(INADDR_ANY);
    server_address.sin_port=htons(9990);
    server_len=sizeof(server_address);
  if( bind(server_sockfd, (struct sockaddr *)&server_address, server_len) == -1){
            perror("bind :");
            exit(1);
    }
    printf("bind OK\n");
       if( (listen(server_sockfd,50)) == -1){
```

```
        perror("Listen :");
        exit(1);
    }
```

1. 建立 Socket 連線

函式 int socket(int domain , int type , int protocol)

A. 參數 domain：要使用的規範 IPV4 的規範 是 AF_INET 或者是 PF_INET 兩者皆表示同一規範

B. 參數 type：傳輸資料所使用的協定 TCP ，所以使用 SOCK_STREAM 此參數。其他參數型態請參考文獻[6]。

C. 參數 protocol：網路位元組順序的 32 位元整數 [2, p9, p13]，大部分連線型態都只支援 protocol=0。

若回傳值為-1 表示 socket 建立失敗，其他則表示成功。範例：

```
server_sockfd=socket(AF_INET,SOCK_STREAM,0);
if( (server_sockfd == -1)
{
    perror("socket :");
    exit(1);
}
```

2.socket 位址結構

大多數的 socket 函式需要 socket 位址結構的指標作為參數，而支援的每種協定組也都定義了它們自己的 socket 位址結構；這些結構的名稱是以 sockaddr_開頭，而且每個協定組都有個自唯一的結尾。IPv4 的 SOCKET 位址結構 sockaddr_in，通常被稱為「網際網路 socket 位址結構」，並且定義在<netinet/in.h>標頭檔中。

```
server_address.sin_family=AF_INET;              //IPv4 協定
server_address.sin_addr.s_addr=htonl(INADDR_ANY); //允許來自任何 IP 位址的連結
server_address.sin_port=htons(9990);     //將 port 號轉為 16 位元
```

3.Client 與 Server 端的連繫

為了讓 socket 可以被其它處理程序使用，一個伺服器程式需要給予 socket 一個名稱。因此，AF_UNIX socket 結合一個檔案系統名稱，AF_INET socket 則結合一個 IP port 數值

```
server_len = sizeof(ser_addr);
if( bind(server_sockfd, (struct sockaddr *)&ser_addr, server_len) == -1)
{
    perror("bind :");
    exit(1);
}
```

bind()系統呼叫，必須在 address 中指定位址，給予未命名的 socket 檔案描述子定義一個名

稱。address_len 則要傳入位止結構長度。bind 成功之後，它會回傳 0。如果失敗，它會回傳 -1。

4.產生一個 socket 佇列

在一個 socket 接收連結請求之前，伺服器程式必須產生一個佇列，這個佇列儲存懸而未決的請求。要做這樣的動作，我們必須使用 listen()系統呼叫：

```
if( (listen(server_sockfd,1)) == -1)       //一次祇允許一個 client 端連線
{
    perror("Listen :");
    exit(1);
}
```

一個 Linux 系統可以限制在佇列中的數量，也就是懸而未決定的連結請求量。為了符合最大量，必須在 listen()中設定佇列長度 backlog。在佇列還有空間的狀況下，socket 還可以接收傳送的連結請求，如果超過佇列長度，後續的客戶端連結請求就會失敗。當伺服器程式在處理先前的客戶時，listen 提供一個機制，讓伺服器在忙碌中，也能接受連結請求。
listen()函數在成功後會回覆 0。失敗後就回傳-1。

5.接受連結

伺服器程式產生和命名 socket 之後，透過 accept()系統呼叫，它會接受這個 socket 的連結請求。

```
client_sockfd = accept(server_sockfd,(struct sockaddr *)&cli_addr, &client_len);
if(client_sockfd == -1)
{
    perror("accept error");
    exit(1);
}
```

當一個客戶端程式試圖連結這個 socket，accept()系統呼叫就會回覆。若客戶端是在 socket 佇列中第一個未進的連結請求。accept()函數產生一個新的 socket 來與這個客戶端溝通，所以會回覆這個 socket 描述子。新的 socket 的型態如同伺服器放在 listen()的 socket 型態一般。

6. 資料收發與結束連線

Server 端接受一個連線要求成功後，即可用 read 及 write 傳送資料，範例如下:

```
write(sockfd,comstr,7); //  傳送字串，字元數為 7。
nread=read(file_fd,sbuf,sizeof(sbuf))  ; //接收字元數為 sizeof(sbuf)的字串，若接受到換行符號則強迫脫離 read 的執行。
```

當傳輸資料或接收資料完畢後，便可結束 socket 連線指令格式說明如下：

int close(int sd)

參數 sd：建立 socket 時的檔案描述元

```
/*  結束  SOCKET  連線*/
      close(sockfd);
```