#include "myftp.h"

int getIFname( int socketfd, char \*device ) {

//Function: To get the device name

//Hint: Use ioctl() with SIOCGIFCONF as an arguement to get the interface list

struct ifconf ifconf;

struct ifreq ifr[10];

ifconf.ifc\_len = sizeof(ifr);

ifconf.ifc\_buf = (char \*) ifr;

if(ioctl(socketfd, SIOCGIFCONF, &ifconf) < 0)

errCTL("ioctl SIOCGIFCONF error");

int interfacelen= ifconf.ifc\_len / sizeof(struct ifreq);

int i;

for(i = 0; i < interfacelen; i++) {

if( strcmp(ifr[i].ifr\_name,"lo" )!=0 )

strcpy( device,ifr[i].ifr\_name );

}

printf ("network interface = %s\n", device);

return 0;

}

int initServAddr( int socketfd, int port, const char \*device, struct sockaddr\_in \*addr ) {

//Function: Bind device with socketfd

// Set sever address(struct sockaddr\_in), and bind with socketfd

//Hint: Use setsockopt to bind the device

// Use bind to bind the server address(struct sockaddr\_in)

if (setsockopt(socketfd, SOL\_SOCKET, SO\_BINDTODEVICE,(char \*)device, sizeof(device)) < 0){

close(socketfd);

errCTL("setsockopt SO\_BINDTODEVICE failed");

}

struct sockaddr\_in server\_addr;

bzero( &server\_addr, sizeof(server\_addr) );

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_port = htons(port);

server\_addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

if( bind(socketfd, (struct sockaddr \*)&server\_addr, sizeof(struct sockaddr\_in)) < 0)

errCTL("bind");

struct ifreq interface;

bzero( &interface, sizeof(struct ifreq) );

strcpy( interface.ifr\_name, device );

if( ioctl(socketfd, SIOCGIFADDR, &interface)<0 )

errCTL("ioctl SIOCGIFADDR error!");

memcpy(addr, (struct sockaddr\_in\*)&interface.ifr\_addr, sizeof(interface.ifr\_addr));

printf("network port = %d\n",port);

return 0;

}

int initCliAddr( int socketfd, int port, char \*sendClient, struct sockaddr\_in \*addr ) {

//Function: Set socketfd with broadcast option and the broadcast address(struct sockaddr\_in)

//Hint: Use setsockopt to set broadcast option

int optval = 1;//true

if (setsockopt(socketfd, SOL\_SOCKET, SO\_BROADCAST, &optval, sizeof(optval)) < 0){

perror(" SO\_BINDTODEVICE ");

close(socketfd);

exit(1);

}

addr->sin\_family = AF\_INET;

addr->sin\_port = htons(port);

if( inet\_aton(sendClient, &(addr->sin\_addr) ) == 0 ){//10.255.255.255

perror("iner\_aton");

exit(1);

}//inet\_aton是一個改進的方法來將一個字符串IP地址轉換為一個32位的網絡序列IP地址

return 0;

}

int findServerAddr( int socketfd, char \*filename, const struct sockaddr\_in \*broadaddr, struct sockaddr\_in \*servaddr ) {

//Function: Send broadcast message to find server

// Set timeout to wait for server replay

//Hint: Use struct bootServerInfo as boradcast message

// Use setsockopt to set timeout

int addr\_len = sizeof(struct sockaddr);

struct bootServerInfo packet;

//send to server

bzero(&packet, sizeof(struct bootServerInfo));

strcpy(packet.filename, filename);

if( sendto(socketfd, &packet, sizeof(struct bootServerInfo), 0, (struct sockaddr \*)broadaddr, sizeof(struct sockaddr\_in))<0 )

errCTL("sendto error");

//set timeout to wait for server replay

struct timeval timeout;

timeout.tv\_sec = 20;

timeout.tv\_usec = 0;

if ( setsockopt (socketfd, SOL\_SOCKET, SO\_RCVTIMEO, (char \*)&timeout, sizeof(timeout)) < 0 )

errCTL("setsockopt SO\_RCVTIMEO failed");

//receive from server

bzero(servaddr, sizeof(struct sockaddr\_in));

//MSG\_WAITALL；在阻塞模式下，接收到符合大小的資料為止，或是超時也會停止

if( recvfrom(socketfd, &packet, sizeof(struct bootServerInfo), MSG\_WAITALL, (struct sockaddr\*)servaddr, &addr\_len) <0 ){

printf("No server answer!!\n");

exit(1);

}

if(packet.filename[0] == '\0'){

printf("No filename \"%s\" in the myftpServer\n",filename);

exit(1);

}else{

printf("[Receive Reply]\n");

printf("\t\tGet MyftpServer servAddr : %s\n", inet\_ntoa(servaddr->sin\_addr));

printf("\t\tMyftp connentPort : %d\n", packet.connectPort);

servaddr->sin\_port = htons(packet.connectPort);

}

return 0;

}

int listenClient(int socketfd, int port, int tempPort, char \*filename, struct sockaddr\_in \*clientaddr, struct sockaddr\_in \*servaddr){

//packet is from client

//packet2 is ready to send to client

struct bootServerInfo packet;

int addr\_len = sizeof(struct sockaddr);

bzero(&packet, sizeof(struct bootServerInfo));

//recvfrom broadcast message from client

if( recvfrom(socketfd, &packet, sizeof(packet), 0L, (struct sockaddr \*)clientaddr, &addr\_len ) < 0 )

errCTL("recvfrom error");

char \*clientip = inet\_ntoa(clientaddr->sin\_addr);

//if receive broadcast message, check file exist, then send back

struct stat buf;

struct bootServerInfo packet2;

bzero(&packet2, sizeof(struct bootServerInfo));

if( lstat(packet.filename, &buf) < 0 ){

packet2.filename[0] = '\0';

}else{

strcpy(packet2.filename, packet.filename);

}

packet2.connectPort = tempPort;

strcpy(packet2.servAddr, inet\_ntoa(servaddr->sin\_addr));

if( (sendto(socketfd, &packet2, sizeof(packet2), 0, (struct sockaddr \*)clientaddr, sizeof(struct sockaddr\_in)))<0 )

errCTL("sendto error");

if( lstat(packet.filename, &buf) < 0 ) {

printf("client request a non-exist file : %s\n", packet.filename);

return 0;

}

else{

strcpy(filename, packet.filename);

return 1;

}

}

int startMyftpServer( int tempPort, struct sockaddr\_in \*clientaddr, const char \*filename ) {

int socketfd;

int addr\_len = sizeof(struct sockaddr);

if( (socketfd= socket(AF\_INET, SOCK\_DGRAM, 0))<0 )

errCTL("socket error");

struct sockaddr\_in client;

bzero(&client, sizeof(client));

client.sin\_family = AF\_INET;

client.sin\_port = htons(tempPort);

client.sin\_addr.s\_addr = htonl(INADDR\_ANY);

if( bind(socketfd, (struct sockaddr \*)(&client), sizeof(struct sockaddr\_in)) < 0 )

errCTL("bind error");

struct timeval timeout;

timeout.tv\_sec = 20;

timeout.tv\_usec = 0;

if( setsockopt(socketfd, SOL\_SOCKET, SO\_RCVTIMEO, (char \*)&timeout, sizeof(timeout)) < 0 )

errCTL("setsockopt SO\_RCVTIMEO failed");

//initialize every packet: FRQ, ACK, ERROR, DATA

struct myFtphdr \*FRQ, \*ACK, \*ERROR, \*DATA;

//FRQ

int FRQsize = strlen(filename) + 1 + 4 ;// filename + '\0' + opcode(2) + checksum(2)

FRQ = (struct myFtphdr \*)malloc( FRQsize );

bzero(FRQ, FRQsize);

//ACK

int ACKsize = 6, ERRORsize = 6;// opcode(2) + checksum(2) + block(2)

ACK = (struct myFtphdr \*)malloc(ACKsize);

bzero(ACK, ACKsize);

//ERROR

ERROR = (struct myFtphdr \*)malloc(ERRORsize);

bzero(ERROR, ERRORsize);

//DATA

int DATAsize = MFMAXDATA + 6;//data + opcode(2) + checksum(2) + block(2)

DATA = (struct myFtphdr \*)malloc(DATAsize);

//printf("wait for client send FRQ\n");

while(1){

//receive FRQ from client

if( recvfrom(socketfd, FRQ, FRQsize, MSG\_WAITALL, (struct sockaddr\*)clientaddr, &addr\_len)<0 ){

printf("time out waiting FRQ\n");

}

else{

if( in\_cksum((unsigned short \*)FRQ, FRQsize) != 0 )

printf("receive FRQ packet but checksum error\n");

else{

printf("\t[Request]\n");

printf("\t\tfile : %s\n",FRQ->mf\_filename);

printf("\t\tfrom : %s\n",inet\_ntoa(clientaddr->sin\_addr));

printf("\t[Reply]\n");

printf("\t\tfile : %s\n",filename);

printf("\t\tconnectPort : %d\n",tempPort);

break;

}

}

}

int block = 1;

FILE \*fin = fopen(filename, "rb");

printf("wait client!\n");

printf("\t[file transmission - start]\n");

printf("\t\tsend file : <%s> to %s\n", FRQ->mf\_filename, inet\_ntoa(clientaddr->sin\_addr));

int readlength;

int finish = 0;

int totalsendbytes = 0;

struct myFtphdr \*finalDATA;

while(1){

if( finish == 1 ) break;

//用fread將檔案讀入DATA封包

bzero(DATA ,DATAsize);

readlength = fread(DATA->mf\_data, 1, MFMAXDATA, fin);

totalsendbytes += readlength;

//檔案讀到最後一筆，因DATAsize可能不足512，需調整

if( feof(fin) && readlength<MFMAXDATA ){

DATAsize = readlength + 6;

finalDATA = (struct myFtphdr \*)malloc(DATAsize);

memcpy( finalDATA->mf\_data, DATA->mf\_data, readlength );

free(DATA);

DATA = finalDATA;

}

//short int 的最大值:2的16次方，超過會溢位

if(block == 65536) block = 1;

DATA->mf\_opcode = htons( DATAop );

DATA->mf\_block = block;

DATA->mf\_cksum = 0;

DATA->mf\_cksum = in\_cksum((unsigned short \*)DATA, DATAsize);

if( sendto(socketfd, DATA, DATAsize , 0, (struct sockaddr \*)clientaddr, sizeof(struct sockaddr\_in)) <0 ){

errCTL("sendto");

}

//wait ACK packet, may receive previous FRQ

while(1){

if( recvfrom(socketfd, ACK, ACKsize, MSG\_WAITALL, (struct sockaddr\*)clientaddr,&addr\_len )<0 ){

printf("time out waiting ACK,send data again\n");

//可能DATA遺失或ACK遺失，DATA遺失代表client沒收到也不會回送ack，ack遺失代表需要client再送一次

//兩者情況對server而言都需要重送DATA

//DATA->mf\_block = htons(block);

DATA->mf\_block = block;

DATA->mf\_cksum = 0;

DATA->mf\_cksum = in\_cksum((unsigned short \*)DATA, DATAsize);

if( sendto(socketfd, DATA, DATAsize , 0, (struct sockaddr \*)clientaddr, sizeof(struct sockaddr\_in)) <0 )

errCTL("sendto");

}else{

if( in\_cksum((unsigned short \*)ACK, ACKsize) !=0 ){

//checksum出錯，必須重送

printf("checksum error,wait next ACK\n");

DATA->mf\_block = block;

DATA->mf\_cksum = 0;

DATA->mf\_cksum = in\_cksum((unsigned short \*)DATA, DATAsize);

if( sendto(socketfd, DATA, DATAsize , 0, (struct sockaddr \*)clientaddr, sizeof(struct sockaddr\_in)) <0 )

errCTL("sendto");

}else{

//checksum正確，區分封包種類 by opcode

if( ntohs(ACK->mf\_opcode) == ACKop ){

//if(ntohs(ACK->mf\_block) == 0){

if(ACK->mf\_block == 0){

//block為0，代表client已經把server送出去的最後一個DATA收到了

//printf("receive block 0,file transmission stop with client IP = %s\n",inet\_ntoa(clientaddr->sin\_addr));

finish =1;

break;

}

if(ACK->mf\_block == block){

block++;

break;

}else{

//this is old packet due to time out,discard it,wait for expected packet

//printf("receive old packet\n");

}

}

else if( ntohs(ACK->mf\_opcode) == ERRORop ){

//如果前一個封包checksum出錯，就會收到一個client發的ERROR

//DATA->mf\_block = htons(block);

DATA->mf\_block = block;

DATA->mf\_cksum = 0;

DATA->mf\_cksum = in\_cksum((unsigned short \*)DATA, DATAsize);

if( sendto(socketfd, DATA, DATAsize , 0, (struct sockaddr \*)clientaddr, sizeof(struct sockaddr\_in)) <0 )

errCTL("sendto");

}

}

}

}

}

fclose(fin);

printf("\t[file transmission - finish]\n");

printf("\t\t%d bytes sent\n", totalsendbytes);

printf("wait client!\n");

return 0;

}

int startMyftpClient( int socketfd, struct sockaddr\_in \*servaddr, const char \*filename ) {

int addr\_len = sizeof(struct sockaddr);

struct timeval timeout;

timeout.tv\_sec = 20;

timeout.tv\_usec = 0;

if ( setsockopt (socketfd, SOL\_SOCKET, SO\_RCVTIMEO, (char \*)&timeout, sizeof(timeout)) < 0 )

errCTL("setsockopt SO\_RCVTIMEO failed");

//initialize every packet: FRQ, ACK, ERROR, DATA

struct myFtphdr \*FRQ, \*ACK, \*ERROR, \*DATA;

//FRQ

int FRQsize = strlen(filename) + 1 + 4 ;// filename + '\0' + opcode(2) + checksum(2)

FRQ = (struct myFtphdr \*)malloc( FRQsize );

bzero(FRQ, FRQsize);

strcpy( FRQ->mf\_filename, filename);

FRQ->mf\_opcode = htons( FRQop );

FRQ->mf\_cksum = 0;

FRQ->mf\_cksum = in\_cksum((unsigned short \*)FRQ, FRQsize );

if(sendto(socketfd, FRQ, FRQsize, 0, (struct sockaddr \*)servaddr, sizeof(struct sockaddr\_in)) == -1)

errCTL("send FRQ failure");

//ACK

int ACKsize = 6, ERRORsize = 6;// opcode(2) + checksum(2) + block(2)

ACK = (struct myFtphdr \*)malloc(ACKsize);

bzero(ACK, ACKsize);

ACK->mf\_opcode = htons(ACKop);

//ERROR

ERROR = (struct myFtphdr \*)malloc(ERRORsize);

bzero(ERROR, ERRORsize);

ERROR->mf\_opcode = htons(ERRORop);

//DATA

int DATAsize = MFMAXDATA + 6;//data + opcode(2) + checksum(2) + block(2)

//int DATAsize = MAXLINE;

DATA = (struct myFtphdr \*)malloc(DATAsize);

//重新命名從server傳過來的檔案

FILE \*fin;

char ftpfile[FNAMELEN];

sprintf(ftpfile, "client\_%s", filename);

fin=fopen(ftpfile, "wb");

printf("[file transmission - start]\n");

printf("\t\tdownload to file : %s\n", ftpfile);

printf("\t\tget file : <%s> from %s\n", filename, inet\_ntoa(servaddr->sin\_addr));

int block = 0 ;

int recvlength;

int totalrecvbytes = 0;

while(1){

bzero(DATA, DATAsize);

//接收時間限制20秒

if((recvlength = recvfrom(socketfd, DATA, DATAsize, MSG\_WAITALL, (struct sockaddr\*)servaddr, &addr\_len))<0){

printf("time out waiting data(block=%d),request server to resend\n",block+1);

ERROR->mf\_block = block;

ERROR->mf\_cksum = 0;

ERROR->mf\_cksum = in\_cksum((unsigned short \*)ERROR, ERRORsize );

if(sendto(socketfd, ERROR, ERRORsize, 0, (struct sockaddr \*)servaddr, sizeof(struct sockaddr\_in)) == -1)

errCTL("send ERROR failure");

}else if( in\_cksum((unsigned short \*)DATA, DATAsize )!=0){

//收到封包必須檢查checksum

printf("received data checksum error\n");

ERROR->mf\_block = block;

ERROR->mf\_cksum = 0;

ERROR->mf\_cksum = in\_cksum((unsigned short \*)ERROR, ERRORsize );

if(sendto(socketfd, ERROR, ERRORsize, 0, (struct sockaddr \*)servaddr, sizeof(struct sockaddr\_in)) == -1)

errCTL("send ERROR failure");

}

else if(ntohs(DATA->mf\_opcode) == DATAop && DATA->mf\_block == block+1){;

int writebytes;

writebytes = strlen(DATA->mf\_data);

totalrecvbytes += writebytes;

fwrite( DATA->mf\_data, 1, writebytes, fin);

if( writebytes<MFMAXDATA ){

//當writebytes<512代表是最後一次從server傳檔

block = 0;

//ACK->mf\_block = htons(block);//最後一次 將block設為0傳回給server

ACK->mf\_block = block;//最後一次 將block設為0傳回給server

ACK->mf\_cksum = 0;

ACK->mf\_cksum = in\_cksum((unsigned short \*)ACK, ACKsize );

if(sendto(socketfd, ACK, ACKsize, 0, (struct sockaddr \*)servaddr, sizeof(struct sockaddr\_in)) == -1)

errCTL("send ACK failure");

break;

}

else{

block = DATA->mf\_block;

ACK->mf\_block = block;

ACK->mf\_cksum = 0;

ACK->mf\_cksum = in\_cksum((unsigned short \*)ACK, ACKsize );

if(sendto(socketfd, ACK, ACKsize, 0, (struct sockaddr \*)servaddr, sizeof(struct sockaddr\_in)) == -1)

errCTL("send ACK failure");

if(block == 65535)//block為short型態，避免溢位

block = 0;

}

}

else if( ntohs(DATA->mf\_opcode) == DATAop && DATA->mf\_block != block+1){

//server沒收到上一個ack，所以又寄一次上一個DATA

int preblock = DATA->mf\_block;

ACK->mf\_block = preblock;

ACK->mf\_cksum = 0;

ACK->mf\_cksum = in\_cksum((unsigned short \*)ACK, ACKsize );

if(sendto(socketfd, ACK, ACKsize, 0, (struct sockaddr \*)servaddr, sizeof(struct sockaddr\_in)) == -1)

errCTL("send ACK failure");

}

}

printf("[file transmission - finish]\n");

printf("\t\t%d bytes received\n", totalrecvbytes);

fclose(fin);

return 0;

}

unsigned short in\_cksum( unsigned short \*addr, int len ) {

int nleft = len;

int sum = 0;

unsigned short \*w = addr;

unsigned short answer = 0;

while( nleft > 1 ) {

sum += \*w++;

nleft -= 2;

}

if( nleft == 1 ) {

\*(unsigned char \*) (&answer) = \*(unsigned char \*) w;

sum += answer;

}

sum = (sum >> 16) + (sum & 0xFFFF);

sum += (sum >> 16);

answer = ~sum;

return (answer);

}