**PROGRAM 4**

**1. TCP CLIENT PROGRAM**

#include <sys/types.h>

#include <sys/time.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <netdb.h>

#include <stdio.h>

#include <unistd.h>

struct sockaddr\_in sock\_in, temp, from\_addr;

int

from\_len;

extern

int

errno;

char \* file\_read(char \*file\_name);

main(argc, argv)

int

argc;

char

\*\*argv;

{

struct timeval

timeout;

register int

n;

u\_short

len;

char

\*cp;

int

i, retry, resplen, done = 0;

int

dsmask, flags, sockFD;

char

buf[100],answer[4048];

char

hostname[100];

struct

hostent

\*h\_name;

struct

servent

\*s\_name;

int port\_no = 2000;

char pof;

int

numTimeOuts

= 0;

sockFD = -1;

strcpy(hostname, "spock.ee.iastate.edu");

opterr = 0;

while ((i = getopt(argc, argv, "hpft")) != -1)

{

switch (i)

{

case 'h':

strcpy(hostname, argv[optind]);

break;

case 't':

// this is a test flag to show how the flags work

// this will print out the parms

printf("%s\n", argv[optind]);

break;

case 'p':

// add code for the p flag set

pof = 'p';

printf("%s\n", argv[optind]);

printf("\nPort\_no = %d\n",port\_num);

break;

case 'f':

// add code for the f flag set

pof = 'f';

strcpy(buf,file\_read(argv[optind]));

break;

case '?':

default:

done = 1;

break;

}

if (done) break;

}

h\_name = gethostbyname(hostname);

sock\_in.sin\_family = AF\_INET;

sock\_in.sin\_port = htons(port\_no);//accepts port number as a flag

sock\_in.sin\_addr.s\_addr

= \*(u\_long \*)h\_name->h\_addr;

printf("port = %d -- %s\n",ntohs(sock\_in.sin\_port),inet\_ntoa(sock\_in.sin\_addr));

// Send request

sockFD = socket(AF\_INET, SOCK\_STREAM, 0);

if (connect(sockFD, &sock\_in, sizeof(sock\_in)) < 0) {

perror("connect request");

(void) close(sockFD);

exit(1);

}

if(pof == 'f')

{

if (send(sockFD, buf, strlen(buf),0) != strlen(buf)) {

perror("send request");

(void) close(sockFD);

exit(1);

}

cp = answer;

if ((n = recv(sockFD, cp, 100, 0)) < 0){

perror("SendRequest");

(void) close(sockFD); }

printf("===<%s>===\n",cp);

}

else

{

strcpy(buf,"hello there\n");

strcpy(buf,"from client");

if (send(sockFD, buf, strlen(buf),0) != strlen(buf)) {

perror("send request");

(void) close(sockFD);

exit(1);

}

cp = answer;

if ((n = recv(sockFD, cp, 100, 0)) < 0){

perror("SendRequest");

(void) close(sockFD);

}

printf("===<%s>===\n",cp);

}

cp[n] = 0;

printf("===<%s>===\n",cp);

(void) close(sockFD);

exit(1);

}

char \* file\_read(char \*file\_name)

{

int i = 0;

char buf[10000];

FILE \*file;

char c;

file = fopen(file\_name,"r");

if(file == NULL)

{

printf("Error!!!!\n");

}

else

{

while((c = fgetc(file)) != EOF)

{

buf[i] = putchar(c);

i++;

}

fclose(file);

}

return buf;

}

**2. TCP SERVER PROGRAM**

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <netdb.h>

#include <stdio.h>

#include <errno.h>

#include <unistd.h>

struct

sockaddr\_in nsaddr, rcvaddr;

struct

sockaddr\_in from\_addr;

/\* Source addr of last packet \*/

int

from\_len;

/\* Source addr size of last packet \*/

#define PORT

2132

extern int errno;

main(argc, argv, envp)

int argc;

char \*argv[], \*envp[];

{

int vs, ns, blen, done=0, i;

int port\_num = PORT;

char buf[4048];

opterr = 0;

while ((i = getopt(argc, argv, "pft")) != -1)

{

switch (i)

{

case 't':

// this is a test flag to show how the flags work

// this will print out the parms

printf("%s\n", argv[optind]);

break;

case 'p':

// add code for the p flag set

port\_num = atoi(argv[optind]);

break;

case 'f':

// add code for the f flag set

printf("We are here");

break;

case '?':

default:

done = 1;

break;

}

if (done) break;

}

nsaddr.sin\_family = AF\_INET;

nsaddr.sin\_addr.s\_addr = INADDR\_ANY;

nsaddr.sin\_port = htons(port\_num);

/\*

\*\* Open stream port.

\*/

if ((vs = socket(AF\_INET, SOCK\_STREAM, 0)) < 0) {

printf("socket(SOCK\_DGRAM): %d\n",errno);

exit(1); }

if (bind(vs, (struct sockaddr \*)&nsaddr, sizeof(nsaddr)) < 0) {

printf("bind(vs, %s[%d]) errno = %d\n "

,inet\_ntoa(nsaddr.sin\_addr), ntohs(nsaddr.sin\_port),errno);

perror("bind error");

exit(1);

}

fprintf(stderr,"SERVER: bind(vs, %s[%d]):\n ",

inet\_ntoa(nsaddr.sin\_addr), ntohs(nsaddr.sin\_port));

printf("SERVER: listen waiting\n");

if ((listen(vs,5)) < 0 ) {

perror("listen");

exit(1);

}

while (1)

{

printf("SERVER: waiting buf size = %d\n",sizeof(buf));

from\_len = sizeof(from\_addr);

if ((ns = accept(vs, (struct sockaddr \*) &from\_addr, &from\_len)) < 0)

perror("accept");

printf("SERVER: accepted call\n");

fprintf(stderr,"SERVER: from\_addr(ns, %s[%d]):\n ",

inet\_ntoa(from\_addr.sin\_addr), ntohs(from\_addr.sin\_port));

while((blen = recv(ns,buf,sizeof(buf), 0))!=0 )

{

buf[blen] = 0;

printf("SERVER: --<%s>--\n",buf);

}

strcpy(buf,"hello");

printf("SERVER: sending\n");

if (send(ns, buf, strlen(buf), 0) != strlen(buf)) {

perror("Sendto");

}

printf("\n");

}

shutdown(ns,2);

}

**3. UDP CLIENT PROGRAM**

#include <sys/types.h>

#include <sys/time.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <netdb.h>

#include <stdio.h>

#include <unistd.h>

#define

HOST

"spock.ee.iastate.edu"

#define PORT

2000

struct sockaddr\_in sock\_in, temp, from\_addr;

int

from\_len;

extern

int

errno;

main(argc, argv)

int

argc;

char

\*\*argv;

{

struct timeval

timeout;

register int

n;

u\_short

len;

char

\*cp;

int

i, retry, resplen, done = 0;

int

dsmask, flags, sockFD;

char

buf[100],answer[4048];

struct

hostent

\*h\_name;

struct

servent

\*s\_name;

char

hostname[100];

int

numTimeOuts

= 0;

sockFD = -1;

opterr = 0;

strcpy(hostname, HOST);

while ((i = getopt(argc, argv, "pft")) != -1)

{

switch (i)

{

case 't':

// this is a test flag to show how the flags work

// this will print out the parms

printf("%s\n", argv[optind]);

break;

case 'p':

// add code for the p flag set

break;

case 'h':

// copy parm to host name

strcpy(hostname, argv[optind]);

break;

case '?':

default:

done = 1;

break;

}

if (done) break;

}

strcpy(buf,"hello there\n");

strcpy(buf,inet\_ntoa(sock\_in.sin\_addr));

h\_name = gethostbyname(hostname);

sock\_in.sin\_family = AF\_INET;

sock\_in.sin\_port = htons(2000);

sock\_in.sin\_addr.s\_addr

= \*(u\_long \*)h\_name->h\_addr;

printf("port = %d -- %s\n",ntohs(sock\_in.sin\_port),inet\_ntoa(sock\_in.sin\_addr));

/\*

\* Send request, RETRY times, or until successful

\*/

for (retry = 4; --retry >= 0; ) {

if (sockFD < 0) {

sockFD = socket(AF\_INET, SOCK\_DGRAM, 0);

if (sockFD < 0) perror("CLIENT: SendRequest1");

temp.sin\_family = AF\_INET;

temp.sin\_port = htons(0);

temp.sin\_addr.s\_addr = INADDR\_ANY;

if (bind(sockFD,(struct sockaddr \*)&temp,sizeof(temp)) < 0)

printf("bind error errno = %d\n",errno);

}

while(1)

{

printf("send message\n");

if (sendto(sockFD, buf, strlen(buf), 0, (struct sockaddr \*) &sock\_in,

sizeof(sock\_in)) != strlen(buf)) {

perror("CLIENT: Sendto");

}

/\* Wait for reply \*/

timeout.tv\_sec = 4;

printf("timeout = %d\n",timeout.tv\_sec);

timeout.tv\_usec = 0;

dsmask = 1 << sockFD;

printf("CLIENT: mask = %d sockFD = %d\n",dsmask,sockFD);

n = select(sockFD+1, &dsmask, 0, 0, &timeout);

if (n < 0) {

perror("CLIENT: select error");

continue;

}

if (n == 0) {

/\* timeout \*/

printf("CLIENT: mask = %d after slect call\n",dsmask);

printf("CLIENT: Timeout %d\n", ++numTimeOuts);

continue;

}

printf("CLIENT: mask = %d sockFD = %d after select call\n",dsmask,

sockFD);

if ((resplen = recv(sockFD, answer, sizeof(answer), 0)) <= 0) {

printf("CLIENT: errno = %d resplen = %d\n",errno,resplen);

printf("CLIENT: fromlen = %d\n",from\_len);

perror("CLIENT: recvfrom error");

continue;

}

}//end of while(1)

buf[resplen] = 0;

printf("CLIENT: Got answer (%d bytes):\n", resplen);

printf("CLIENT: ==<%s>==",buf);

(void) close(sockFD);

sockFD = -1;

exit(0);

}

(void) close(sockFD);

sockFD = -1;

exit(1);

}

**4. UDP SERVER PROGRAM**

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <netdb.h>

#include <stdio.h>

#include <errno.h>

#include <unistd.h>

#define PORT

2000

struct

sockaddr\_in nsaddr, rcvaddr;

struct

sockaddr\_in from\_addr;

/\* Source addr of last packet \*/

int

from\_len;

/\* Source addr size of last packet \*/

extern int errno, opterr;

main(argc, argv, envp)

int argc;

char \*argv[], \*envp[];

{

int n, vs;

int i, blen, addlen, done=0;

char buf[4048];

rcvaddr.sin\_family = AF\_INET;

nsaddr.sin\_family = AF\_INET;

nsaddr.sin\_addr.s\_addr = INADDR\_ANY;

nsaddr.sin\_port = htons(PORT);

opterr = 0;

while ((i = getopt(argc, argv, "pft")) != -1)

{

switch (i)

{

case 't':

// this is a test flag to show how the flags work

// this will print out the parms

printf("%s\n", argv[optind]);

break;

case 'p':

// add code for the p flag set

break;

case 'f':

// add code for the f flag set

break;

case '?':

default:

done = 1;

break;

}

if (done) break;

}

/\*

\*\* Open stream port.

\*/

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

printf("SERVER: socket(SOCK\_DGRAM): %d\n",errno);

exit(1);

}

if (bind(vs, (struct sockaddr \*) &nsaddr, sizeof(nsaddr)) < 0) {

printf("bind(vs, %s[%d]) errno = %d\n "

,inet\_ntoa(nsaddr.sin\_addr), ntohs(nsaddr.sin\_port),errno);

exit(1);

}

printf("SERVER: bind(vs, %s[%d]):\n ",

inet\_ntoa(nsaddr.sin\_addr), ntohs(nsaddr.sin\_port));

printf("SERVER: waiting buf size = %d\n",sizeof(buf));

addlen = sizeof(rcvaddr);

while(1)

{

blen = recvfrom(vs,buf,sizeof(buf), 0 , (struct sockaddr \*) &rcvaddr, &addlen);

printf("SERVER: data from (vs, %s[%d]):\n ",

inet\_ntoa(rcvaddr.sin\_addr), ntohs(rcvaddr.sin\_port));

buf[blen] = 0;

printf("--<%s>--",buf);

strcpy(buf,"hello from server\n");

printf("SERVER: sending\n");

if (sendto(vs, buf, strlen(buf), 0, (struct sockaddr \*) &rcvaddr,

sizeof(rcvaddr)) != strlen(buf)) {

perror("Sendto");

}

}

sleep(5);

}

**Reference:**

1. http://www.dougj.net/modules/index.html