Cpre 530 - Program 4
Fall 2011
Suganya Baskaran

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