Overview

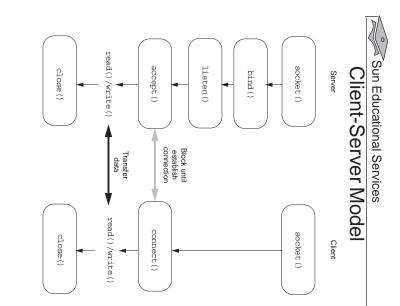
- Objectives
- Relevance

UNIX® System Interface Programming

March 2001

UNIX® System Interface Programming

Module 12, slide 2 of 54





Sun Educational Services

Socket Types

SOCK STREAM - TCP

- Connection-oriented
- No message boundaries
- Reliable
- Sequenced
- Easier to use with reliability, but more expensive

SOCK DGRAM - UDP

- Connectionless
- Message boundaries
- Not reliable, not sequenced
- Duplicates possible
- More efficient

Combining Socket Domains and Types

- AF_UNIX, SOCK_STREAM
- AF UNIX, SOCK DGRAM
- AF INET, SOCK STREAM
- AF INET, SOCK DGRAM

UNIX® System Interface Programming
Copyright 2001 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services, Revision I

Module 12, slide 5 of 54

Sun Educational Services

Combining Socket Domains and Types

Owner's Activities

Human Speak	Computer Speak	System Call
Buy a phone	Establish an end point	socket (3SOCKET)
Get a phone number	Establish a rendezvous	bind(3SOCKET)
Activate the phone line.	Set queue length and enable service	listen(3SOCKET)
Wait for a new client and redirect to Cesare's phone.	Acknowledge	accept (3SOCKET) and fork (2)
Cesare takes order	Full duplex conversation	<pre>read(2), write(2), recv(3SOCKET), and send(3SOCKET)</pre>

UNIX® System Interface Programming
Copyright 2001 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services, Revision

Module 12, slide 6 of 54



Sun Educational Services

Combining Socket Domains and Types

Client's Activities

Human Speak	Computer Speak	System Call
Walk to any phone.	Establish an end point	socket()
Look for "Pizza" in phone book	Use directory services	gethostbyname(3NSL)
Dial the number	Request connection	connect (3SOCKET)
Place order	Full duplex conversation	<pre>read(), write(), recv(), and send()</pre>
Hang up	Send EOF	close(2)



Sun Educational Services

Combining Socket Domains and Types

AF_UNIX, SOCK_STREAM Call Sequences

Server	Client
sd = socket	sd = socket
bind	
listen	
nsd = accept	connect
write, read, send, recv	write, read, send, recv
close(nsd)	close(sd)
close(sd)	

Combining Socket Domains and Types

AF UNIX, SOCK DGRAM Call Sequences

Server	Client
sd = socket	sd = socket
bind	bind
recvfrom, sendto	sendto, recvfrom
close(sd)	close(sd)

UNIX® System Interface Programming Copyright 2001 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services, Revision I Module 12, slide 9 of 54

Sun Educational Services

Combining Socket Domains and Types

AF INET, SOCK STREAM Call Sequences

Server	Client
sd = socket	sd = socket
bind	
listen	gethostbyname
nsd = accept	connect
write, read, send, recv	write, read, send, recv
close (nsd)	close(sd)
close(sd)	

UNIX® System Interface Programming Copyright 2001 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services, Revision I Module 12, slide 10 of 54



Sun Educational Services

Combining Socket Domains and Types

AF INET, SOCK DGRAM Call Sequences

Server	Client
sd = socket	sd = socket
bind	bind
	gethostbyname
recvfrom, sendto	sendto, recvfrom
close(sd)	close(sd)



Sun Educational Services

Creating and Destroying Sockets

- socket () Creates a socket
- shutdown() Destroys a socket

Binding an Address to a Socket

- bind() Assigns a local address (or name) to the socket descriptor
- listen() Activates a socket

UNIX® System Interface Programming Copyright 2001 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services, Revision I

Module 12, slide 13 of 54



Binding an Address to a Socket

UNIX Example

```
struct sockaddr un name;
    sd = socket (AF UNIX, SOCK STREAM, 0);
    name.sun family = AF UNIX;
    strcpy(name.sun path, "/tmp/socket1");
    unlink( name.sun path ); /* Otherwise bind could fail */
    if( bind(sd, (struct sockaddr*)&name, sizeof(name) ) == -1 )
10
      perror("bind");
12
      exit(1);
13
```

UNIX® System Interface Programming Copyright 2001 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services, Revision Module 12, slide 14 of 54



Sun Educational Services

Binding an Address to a Socket

Internet Example

```
struct sockaddr in sin;
    int sd;
3
    sd = socket (AF INET, SOCK STREAM, 0);
   memset((char *)&sin, '\0', sizeof(sin));
    sin.sin family = AF INET;
    sin.sin port = htons(7000); /* pick */
    sin.sin addr.s addr = htonl(INADDR ANY);
    if (bind(sd, (struct sockaddr *)&sin, sizeof(sin))
10
            == -1)
11
       perror("bind");
12
      exit(1);
13
14
15
   if (listen(sd, 5) == -1)
       perror("listen?");
17
       exit(1);
18
```



Sun Educational Services

Accepting a Connection

```
struct sockaddr un client;
    int len = sizeof(client), comm;
    while ((comm = accept(sd, &client, &len)) != -1) {
      if (fork() == 0)
        /* child to handle session */
        close(sd); /* do not need rendezvous */
        do service(comm); /* talk to client using
10
            comm */
11
        close (comm);
12
        exit(0);
13
14
      close(comm); /* do not need communication */
15
```

goto loop;

```
16  /* If a signal arrives during accept(), it fails
17  with -1. So, if catching signals, use the following
18  2 lines */
19  if (errno == EINTR) {
```

UNIX® System Interface Programming

21 }

March 2001



Connecting to the Server

Client Side Example

UNIX® System Interface Programming
Copyright 2001 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services, Revision

Module 12, slide 18 of 54



Sun Educational Services

Connecting to the Server

Internet Socket Example

```
int sd;
struct sockaddr_in sin;
struct hostent *he;

sd = socket(AF_INET, SOCK_STREAM, 0);

memset((char *)&sin, '\0', sizeof(sin));
sin.sin_family = AF_INET;

/* match server */
he = gethostbyname("Server");
sin.sin_port = htons(7000);
memcpy((char *)&sin.sin_addr, he->h_addr_list[0],
he->h_length);
```



Receiving Data

```
char message[80];
    int length = sizeof(message);
    int rval, i;
    if ((rval = recv(sd, message, length, 0)) == -1) {
        perror("recv");
      exit(1);
9
    for (i = 0; i < rval; i++) {
10
      putchar(message[i]);
11 }
```

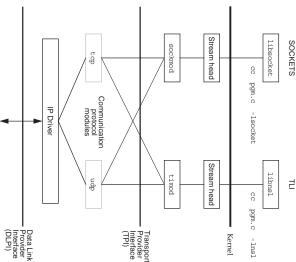
UNIX® System Interface Programming Copyright 2001 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services, Revision I Module 12, slide 21 of 54



Network Sockets and TLI

	Sockets	TLI
Performance	Comparable – both use STREAMS	Comparable - both use STREAMS
POSIX	Accepted for adoption	Accepted for adoption
Transport independence	No	Yes, but must purchase unbundled software if you want to support other than TCP and UDP





Network Sockets and T

Sun Educational Services

Sun Educational Services

Establishing a Connect to a Remote Host

```
/* gethost.c */
    #include <sys/types.h>
    #include <sys/socket.h>
    #include <netdb.h>
    #include <stdio.h>
    main(int argc, char *argv[]) {
10
      struct hostent *he;
11
      char
             **tmp;
12
      if (argc != 2) {
13
        fprintf(stderr, "Usage: %s hostname\n", argv[0]);
14
15
        exit(1);
16
17
      he = gethostbyname(argv[1]);
18
19
      if (he == NULL) {
```

```
20
        fprintf(stderr, "No such host.\n");
21
        exit(1):
22
      printf("Official name of %s is %s.\n",
23
24
              arqv[1], he->h name);
25
      for (tmp=he->h aliases; *tmp != NULL; tmp++)
26
27
        printf("\tAlias: %s\n", *tmp);
28
29
30
      tmp = he->h addr list;
31
      printf("IP address: %d.%d.%d.%d\n",
32
                 (unsigned char) (*tmp) [0],
33
                 (unsigned char) (*tmp) [1],
34
                 (unsigned char) (*tmp) [2],
35
                 (unsigned char) (*tmp) [3]);
36
      return 0;
37
```

UNIX® System Interface Programming
Copyright 2001 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services, Revision I

Module 12, slide 25 of 54

Copyright 2001 Guil Microsystems, His. All regiles read vad. Efficiplise del vices, revision



Sun Educational Services

Determining File Descriptor Activity

```
#include <sys/types.h>
    #include <sys/socket.h>
    #include <netinet/in.h>
    #include <poll.h>
    #include <errno.h>
    /* Number of file descriptors to poll. */
    #define NUM FDS TO POLL 2
10
    main() {
11
12
      int udpsd1, udpsd2;
13
      struct sockaddr in sin1;
      struct sockaddr in sin2;
      struct sockaddr in returnAddr;
15
      int returnAddrSize;
17
```

UNIX® System Interface Programming
Copyright 2001 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services, Revision I

Module 12, slide 26 of 54



Sun Educational Services

```
/* This array of struct pollfd defines fully
18
      what poll() is to wait on, and provides a place
19
20
      for poll() to respond. */
      struct pollfd fds[NUM FDS TO POLL];
22
23
      /* Set up two UDP sockets to wait on. */
24
25
     memset((char *)&sin1, '\0', sizeof(sin1));
26
      sin1.sin family = AF INET;
      \sin 1.\sin port = htons(7000);
27
28
      sin1.sin addr.s addr = htonl(INADDR ANY);
29
      /* create one udp sd */
30
31
      udpsd1 = socket (AF INET, SOCK DGRAM, 0);
32
      if (udpsd1 == -1) {
33
       perror ("udp socket");
        exit (errno);
34
35
36
```



```
if (bind(udpsd1, (struct sockaddr *)&sin1,
37
        sizeof(sin1)) == -1)
38
39
        perror ("tcp bind");
40
        exit (errno);
41
42
43
      memset((char *)&sin2, '\0', sizeof(sin2));
      sin2.sin family = AF INET;
45
      sin2.sin port = htons(7001);
46
      sin2.sin addr.s addr = htonl(INADDR ANY);
47
48
      /* create one udp sd */
      udpsd2 = socket (AF INET, SOCK DGRAM, 0);
49
50
      if (udpsd2 == -1) {
51
        perror ("udp socket");
52
        exit (errno);
53
54
```



```
55
      if (bind(udpsd2, (struct sockaddr *)&sin2,
56
        sizeof(sin2)) == -1) {
57
        perror ("udp bind");
58
        exit (errno);
59
60
61
62
      Initialize each element of the fds array with:
63
       - fd: the file descriptor of interest.
64
       - events: what type of event to wait on.
65
66
      Poll returns with the revents field set with the
      events which are ready to do. E.G. if it
67
68
      returns POLLIN, a read can be done without
69
      blocking.
70
      */
71
72
      fds[0].fd = udpsd1;
73
      fds[0].events = POLLIN;
74
      fds[1].fd = udpsd2;
75
      fds[1].events = POLLIN;
76
```

UNIX® System Interface Programming Copyright 2001 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services, Revision I

Module 12, slide 29 of 54



Sun Educational Services

```
77
78
      Timeout value (poll's third argument) can be:
79
       0: Immediate return whether or not fd is ready.
        -1 (INFTIM): Blocks until a fd is ready.
80
81
        <other value>: time in mS to wait for a fd
82
        before returning.
83
84
85
      for (;;) {
86
        int numfds;
87
        if ((numfds = poll(fds, NUM FDS TO POLL, -1)) < 0) {
88
          perror ("poll");
89
          exit (errno);
90
        } else
91
          printf ("Poll returned %d fds to read.\n",
92
            numfds):
93
94
```

UNIX® System Interface Programming Copyright 2001 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services, Revision I Module 12, slide 30 of 54

Sun Educational Services

```
/* if udp sd ready, use it */
95
        if (fds[0].revents == POLLIN) {
96
97
          char buffer[80];
98
          if (recvfrom(udpsd1, buffer, 80, 0,
            (struct sockaddr *) & returnAddr,
99
            &returnAddrSize) >= 0) {
100
101
          puts(buffer);
102
          } else {
103
          perror ("recvfrom");
104
105
106
```



```
/* if udp sd ready, use it */
107
        if (fds[1].revents == POLLIN)
108
109
          char buffer[80];
110
          if (recvfrom(udpsd2, buffer, 80, 0,
            (struct sockaddr *)&returnAddr,
111
112
            &returnAddrSize) >= 0) {
113
          puts(buffer);
114
          } else {
115
          perror ("recvfrom");
116
117
118
119
120
```

Asynchronous I/O

```
int char present;
    void handler() {
      char present++;
    int fd, flags;
    /* Install handler for SIGPOLL, see Signals module*/
9
10
        install disp(SIGPOLL, handler);
11
12
    /* Register fd to have SIGPOLL sent when IO is
13
     * possible. Usually fd is a pipe or socket. */
14
      ioctl(fd, I SETSIG, S RDNORM);
15
16
      while(1) {
17
```

UNIX® System Interface Programming Copyright 2001 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services, Revision I Module 12, slide 33 of 54



Sun Educational Services

```
/* Block SIGPOLL to prevent char present being changed */
19
      if (char present) {
20
        /* Can use poll(2) to determine which
21
22
        descriptor sent the SIGPOLL, if more than one. */
23
24
        char present = 0;
25
26
27
      /* Unblock SIGPOLL */
28
29
         /* Main program work is in this loop */
30
31
```

UNIX® System Interface Programming Copyright 2001 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services, Revision I Module 12, slide 34 of 54

Sun Educational Services

AF UNIX, SOCK STREAM Example

Server Side

```
/* srvr1.c - AF UNIX, SOCK STREAM
    * Server reads a value from client and
     * sends back twice that value */
    #include <stdio.h>
    #include <sys/types.h>
    #include <sys/socket.h>
    #include <svs/un.h>
9
10
      main() {
11
12
      int sd, comm;
13
      struct sockaddr un myname, client;
14
      int namesize = sizeof(struct sockaddr un);
15
      int clientsize = sizeof(struct sockaddr un);
16
      double dub;
17
```

```
/* Create server's rendezvous socket sd */
      if ( (sd = socket(AF UNIX, SOCK STREAM, 0)) == -1) {
19
20
        perror("srvr1.c:main:socket");
21
        exit(1);
22
23
24
      /* Fill in server's address and bind it to sd */
      myname.sun family = AF UNIX;
26
      strcpy(myname.sun path, "/tmp/socket1");
27
      unlink( myname.sun path );
28
29
      if (bind(sd, (struct sockaddr*)&myname, namesize)
30
              == -1 ) {
31
        perror("srvr1.c:main:bind");
32
        exit(1);
33
34
35
      /* Prepare to receive multiple connect requests */
36
      if ( listen(sd, 128) == -1 ) ·
37
        perror("srvr1.c:main:listen");
38
        exit(1);
39
```

```
40
41
      /* Infinite loop to accept client requests */
42
      while(1) {
        comm = accept(sd, (struct sockaddr*)&client,
43
            &clientsize):
44
45
       if(comm == -1) {
          perror("srvrl.c:main:accept");
46
47
          exit(1);
48
49
        read (comm, &dub, sizeof(dub));
50
        dub += dub;
51
        write( comm, &dub, sizeof(dub));
52
        close (comm);
53
54
      /* Install a signal handler for cleanup including:
56
       * close(sd);
57
       * unlink( myname.sun path ); */
58
```

UNIX® System Interface Programming

Module 12, slide 37 of 54

Copyright 2001 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services, Revision I



Sun Educational Services

Client Side

```
/* clnt1.c - AF UNIX, SOCK STREAM */
    #include <stdio.h>
    #include <svs/tvpes.h>
    #include <sys/socket.h>
    #include <sys/un.h>
    main() {
9
      int namesize = sizeof(struct sockaddr un);
11
12
      struct sockaddr un srvr;
      double dub=2.3, dub2;
13
14
15
      /* Create client's socket sd */
      if ( (sd = socket (AF UNIX, SOCK STREAM, 0) ) == -1)
17
        perror("clnt1.c:main:socket");
18
        exit(1);
19
20
```

UNIX® System Interface Programming

Module 12, slide 38 of 54

Copyright 2001 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services, Revision I



Sun Educational Services

```
/* Fill in server's address and connect to server */
21
      srvr.sun familv = AF UNIX;
23
      strcpy(srvr.sun path, "/tmp/socket1");
25
        if (connect(sd, (struct sockaddr*)&srvr, namesize)
26
          == -1 ) {
2.7
       perror("clnt1.c:main:connect");
28
        exit(1);
29
30
31
      /* Communicate with server */
      write(sd, &dub, sizeof(dub));
33
     read (sd, &dub2, sizeof(dub2));
34
      close(sd);
35
      printf("dub: %.2f, dub2: %.2f\n", dub, dub2);
36
```



Sun Educational Services

AF UNIX, SOCK DGM Example

Server Side

```
1 /* srvr2.c - AF UNIX, SOCK DGRAM
   * Server reads a string from client and
    * sends back a greeting appended to the string */
    #include <sys/types.h>
    #include <sys/socket.h>
    #include <sys/un.h>
    #define BUFSZ 256
10
11
   main() {
12
13
     int sd, comm;
14
     struct sockaddr un myname, client;
     int namesize = sizeof(struct sockaddr un);
16
      char buf [BUFSZ];
17
```

```
18
      /* Create server's socket sd */
19
      if ( (sd = socket (AF UNIX, SOCK DGRAM, 0) ) == -1) {
20
        perror("srvr2.c:main:socket");
21
        exit(1):
22
23
      /* Fill in server's address and bind it to sd */
24
      myname.sun family = AF UNIX;
25
26
      strcpy(myname.sun path, "/tmp/socket2srvr");
27
      unlink( myname.sun path );
28
29
      if (bind(sd, (struct sockaddr*) & myname, namesize)
30
                == -1 ) {
31
        perror("srvr2.c:main:bind");
32
        exit(1);
33
34
```

UNIX® System Interface Programming Copyright 2001 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services, Revision I Module 12, slide 41 of 54



Sun Educational Services

```
/* Infinite loop to accept client requests */
36
      while(1){
37
        if( recvfrom(sd, (char*)&buf, BUFSZ, 0,
38
            (struct sockaddr*)&client, &namesize) == -1)
39
          perror("srvr2.c:main:recvfrom");
40
          exit(1);
41
42
        strcat(buf, ", Hello from the server");
43
44
        if ( sendto(sd, buf, BUFSZ, 0,
45
            (struct sockaddr*) &client, namesize) == -1) {
46
          perror("srvr2.c:main:sendto");
47
          exit(1);
48
49
50
51
      /* Install a signal handler for cleanup including:
52
       * close(sd);
53
       * unlink( myname.sun path ); */
54
```

UNIX® System Interface Programming

Module 12, slide 42 of 54

Copyright 2001 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services, Revision I



Sun Educational Services

Client Side

```
/* clnt2.c - AF UNIX, SOCK DGRAM */
    #include <sys/types.h>
    #include <sys/socket.h>
    #include <sys/un.h>
    #define BUFSZ 256
8
9
    main() {
10
     int namesize = sizeof(struct sockaddr un);
11
12
      struct sockaddr un clnt, srvr;
13
      char buf [BUFSZ];
14
15
      /* Create client's socket sd */
16
17
      if ( (sd = socket (AF UNIX, SOCK DGRAM, 0)) == -1) {
        perror("clnt2.c:main:socket");
18
19
        exit(1);
20
```



```
21
      /* Fill in client's address and bind it to sd */
22
23
      clnt.sun family = AF UNIX;
      strcpy(clnt.sun path, "/tmp/socket2clnt");
24
25
      unlink( clnt.sun path );
26
2.7
      if( bind(sd, (struct sockaddr*)&clnt, namesize)
28
         == -1 ) {
29
        perror("clnt2.c:main:bind");
30
        exit(1); }
31
32
      /* Fill in server's address for sendto server */
      srvr.sun family = AF UNIX;
33
      strcpy(srvr.sun path, "/tmp/socket2srvr");
34
35
36
      /* Communicate with server */
      if (sendto(sd, "Deac", 5.0, (struct sockaddr*)&srvr,
37
38
        namesize) ==-1) {
39
        perror("clnt2.c:main:sendto");
40
        exit(1):
41
42
```

```
if( recvfrom(sd,buf,BUFSZ,0,(struct sockaddr*)&srvr,
43
44
            &namesize) ==-1) {
45
        perror("clnt2.c:main:recvfrom");
46
47
48
      printf("CLIENT REC'D: %s\n", buf);
49
      close(sd);
      unlink(clnt.sun path);
50
51
```

UNIX® System Interface Programming Copyright 2001 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services, Revision I Module 12, slide 45 of 54

Sun Educational Services

AF INET, SOCK STREAM Example

Server Side

```
/* srvr3.c - Server - full example
    * AF INET, SOCK STREAM Time Server
     * Server sends its current time */
      #include <stdio.h>
      #include <svs/tvpes.h>
      #include <sys/socket.h>
      #include <sys/utsname.h>
9
      #include <netdb.h>
      #include <netinet/in.h>
11
      #include <erroo h>
      #include <time.h>
12
13
      #define PORTNUM 5998
14
15
```

UNIX® System Interface Programming Copyright 2001 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services, Revision I Module 12, slide 46 of 54

Sun Educational Services

```
main() {
16
17
18
      struct utsname name;
      struct sockaddr in socketname, client;
20
      int sd, ns, clientlen = sizeof(client);
      struct hostent *host;
21
22
      time t today;
23
24
      /* determine server system name and internet address */
25
      if (uname(&name) == -1) {
26
       perror("uname");
27
        exit(1);
28
29
      if ((host = gethostbyname(name.nodename)) == NULL)
30
31
       perror("gethostbyname");
        exit(1);
32
33
34
```



```
/* fill in socket address structure */
      memset((char *) &socketname, '\0', sizeof(socketname));
      socketname.sin family = AF_INET;
37
38
      socketname.sin port = PORTNUM;
39
      memcpy( (char *) &socketname.sin addr,
40
            host->h addr,host->h length);
41
42
      /* open socket */
43
      if ((sd = socket(AF INET, SOCK STREAM, 0)) == -1)
        perror("socket");
44
45
        exit(1):
46
47
48
      /* bind socket to a name */
      if (bind(sd, (struct sockaddr *) & socketname,
49
50
                sizeof(socketname))) {
51
        perror("bind");
52
        exit(1);
53
54
```

```
/* prepare to receive multiple connect requests */
56
      if (listen(sd, 128)) {
57
        perror("listen");
58
        exit(1);
59
60
61
      while (1) {
62
        if ((ns = accept(sd, (struct sockaddr *)&client,
63
                          &clientlen)) == -1) {
64
          perror("accept");;
65
          exit(1);
66
67
68
        /* get current time and date */
69
        time(&today);
70
```

UNIX® System Interface Programming
Copyright 2001 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services, Revision I

Module 12, slide 49 of 54

UNIX® System Interface Programming
Copyright 2001 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services, Revision I

Sun Educational Services

/* send time through socket */

perror("write");

exit(1);

close(ns);

if(write(ns, &today, sizeof(today)) == -1) {

71

72

73

74

75

76

77

78

Module 12, slide 50 of 54



Sun Educational Services

Client Side

```
1 /* clnt3.c - Client full example
    * AF INET, SOCK STREAM Client of Time Server */
4
    #include <stdio.h>
    #include <sys/types.h>
   #include <sys/socket.h>
    #include <svs/utsname.h>
    #include <netdb.h>
    #include <netinet/in.h>
10 #include <errno.h>
    #include <time.h>
11
12
13
   #define PORTNUM 5998
14
15 main() {
16
17
      int sd:
18
      struct sockaddr in server:
19
     struct hostent *host;
20
     time t srvrtime;
```



```
21
22
      struct utsname name;
23
      /* create the socket for talking to server*/
      if ((sd = socket(AF INET, SOCK STREAM, 0)) == -1)
25
        perror("socket");
26
        exit(1);
27
28
29
      /* get server internet address and put into addr
       * structure fill in the socket address structure
30
31
       * and connect to server */
32
      memset((char *) &server, '\0', sizeof(server));
33
      server.sin family = AF INET;
34
      server.sin port = PORTNUM;
35
36
      /* Server is local system. Get its name. */
37
      if (uname(&name) == -1) {
38
        perror("uname");
39
        exit(1):
40
41
      if ((host = gethostbyname("name.nodename")) == NULL) {
        perror("gethostbyname");
42
```



```
43
        exit(1);
44
45
     memcpy((char *)&server.sin addr, host->h addr,
46
             host->h length);
47
     /* connect to server */
48
     if (connect(sd, (struct sockaddr *)&server,
49
50
               sizeof(server))) {
51
       perror("connect");
52
       exit(1);
53
54
     /* read the time and date passed from the server */
55
     if(read(sd, &srvrtime, sizeof(srvrtime)) == -1) {
56
       perror("recv");
57
58
       exit(1);
59
60
     close(sd);
61
     printf("Server's time is: %s", ctime(&srvrtime));
62
63 }
```

UNIX® System Interface Programming
Copyright 2001 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services, Revision I

Module 12, slide 53 of 54



Exercise: Sockets

- Objectives
- Tasks
- Discussion
- Solutions

UNIX® System Interface Programming
Copyright 2001 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services, Revision

Module 12, slide 54 of 54