- **A Simple Stream Server**

- This server sends the string **"Hello, World!\n"** out over a stream connection, to any client that connects.

- You can test this server: run it in one machine, and telnet to it from another with:

- $ telnet *ServerName* 3490

- where *ServerName* is the name of the machine on which you're running the server.

- **The server code (Skeleton):**

- #include .....

- #define MYPORT 3490 /* the port clients will be connecting to */

- #define BACKLOG 10 /* how many pending connections queue I will hold */

- **main()**
- {
- int **sockfd**, **new_sockfd**;
/* listen on **sock_fd**, new connection on **new_sockfd** */
- struct sockaddr_in **my_addr**;
/* my address information */
- struct sockaddr_in **their_addr**;
/* Client's address information */

- int **sin_size, pid**;

- **sockfd** =
  socket(AF_INET,
  SOCK_STREAM, 0)
- **my_addr.sin_family** = AF_INET;

- **my_addr.sin_port** =
  htons(MYPORT);
  /* short, network byte order */

my_addr.sin_addr.s_addr =
INADDR_ANY;
/* auto-fill with my IP */

- **bind**(**sockfd**, (&**my_addr**,
  sizeof(struct sockaddr)))

- **listen**(**sockfd**, BACKLOG)

```
• while(1) {  /* main accept() loop */
• new_sockfd = accept(sockfd,
                      &their_addr);
}
printf("server: got connection from
%s\n", their_addr.sin_addr);

pid = fork();

• if (pid == 0) { /* this is the child
process */
• send(new_sockfd, "Hello, world!\n",
•                              14,0)
• close(new_sockfd);
• exit(0);/* End Child Process */
• }
• close(new_sockfd);  /* parent doesn't
need  this */
• }
```

- **A Simple, Skeleton Stream Client**

- This client connects to the host you specify on the command line, port 3490.

- It gets the string that the server sends and prints it on the screen.

- usage: client *hostname*

- **The client code (skeleton):**

```
- #include <stdio.h>
- #include .....

#define PORT 3490   /* the port I will
                       be connecting to */

- #define MAXDATASIZE 100 /* Max
  number of bytes we can get at once */

- int main(int argc, char *argv[])
- {
- int sockfd, numbytes;
- char buf[MAXDATASIZE];
- struct hostent *he;
- struct sockaddr_in  their_addr;
      /* Server's address information */

- he = gethostbyname(argv[1])
```

- **sockfd** = **socket**(AF_INET,
  SOCK_STREAM, 0)

- **their_addr.sin_family** = AF_INET;

- **their_addr.sin_port** = **htons**(PORT);
  /* short, network byte order */

- **their_addr.sin_addr** = **he->h_addr**);

- **connect**(**sockfd**, &**their_addr**,
  sizeof(struct sockaddr));

- **numbytes** = **recv**(**sockfd**, **buf**,
  MAXDATASIZE, 0);

- **buf**[**numbytes**] = '\0';

- **printf**("Received: %s", **buf**);
- **close**(**sockfd**);

- return 0; }

Example Datagram programs:

- talker.c and listener.c.
- **listener** sits on a machine waiting for an incoming packet on port 4950.
- **talker** sends a packet to that port, on the specified machine, that contains whatever the user enters on the command line.

- **The server (listener.c) code (skeleton):**

```
- #include .....

#define MYPORT 4950    /* the port
      users will be connecting to */

- #define MAXBUFLEN 100

- main()
- {
- int sockfd;
- struct sockaddr_in  my_addr;
      /* my  address information */
- struct sockaddr_in   their_addr;
/*  connector's address information */
- int addr_len, numbytes;
- char buf[MAXBUFLEN];

- sockfd socket(AF_INET,
                  SOCK_DGRAM, 0)
```

- **my_addr.sin_family** = AF_INET;
- **my_addr.sin_port** = **htons**(MYPORT);
  /* short, network byte order */
- **my_addr.sin_addr.s_addr** = INADDR_ANY;
  /* auto-fill with my IP */
- **bind**(**sockfd**, (**my_addr**, **sizeof**(struct sockaddr)));
- **numbytes = recvfrom**(**sockfd**, **buf**, MAXBUFLEN, 0, &**their_addr**, &**addr_len**);
- **printf**("got packet from %s\n", their_addr.sin_addr);
- **printf**("packet is %d bytes long\n", **numbytes**);
- **buf**[**numbytes**] = '\0';

```
• printf("packet contains \"%s\"\n", buf);
• close(sockfd);
• } /* End Datagram Server */
```

The Client (talker.c) code (skeleton):

```
#include ....

#define ServPORT 4950
/* the port clients will be connecting to */

main(int argc, char *argv[])
{
int sockfd;
struct sockaddr_in  their_addr;
/* Server's address information */
struct hostent *he;
int numbytes;

he = gethostbyname(argv[1]))

sockfd = socket(AF_INET,
SOCK_DGRAM, 0) ;
```

- **their_addr.sin_family** = AF_INET;

- **their_addr.sin_port** = **htons**(ServPORT);
  /* short, network byte order */

- **their_addr.sin_addr** = **he->h_addr**;

- **numbytes = sendto**(**sockfd**, argv[2], **strlen**(argv[2]), 0, &**their_addr**, **sizeof**(struct sockaddr));

- **printf**("sent %d bytes to %s\n", **numbytes**, **their_addr.sin_addr**);

- **close**(**sockfd**);

- return 0;
- }/* End Datagram Client */

- **Example code for select:**

- The following code waits 2.5 seconds for something to appear on standard input:

- #include ...

- #define **STDIN** 0
/* file descriptor for standard input */

- **main**()
- {
- struct timeval **timeout**;
- fd_set **readfds**;

- **timeout.tv_sec** = 2;
- **timeout.tv_usec** = 500000;

- **FD_ZERO**(&**readfds**);
- **FD_SET**(STDIN, &**readfds**);

- /* don't care about writefds and exceptfds: */
- **select**(STDIN+1, &**readfds**, NULL, NULL, &**timeout**);
- if (**FD_ISSET**(STDIN, &**readfds**))
- **printf**("A key was pressed!\n");
- else
- **printf**("Timed out.\n");
- }

- Note: if you have a socket that is **listen()**'ing, you can check to see if there is a new connection by putting that socket's file descriptor in the **readfds** set.