**Simple tcp Server & Client**

**server:**
- Creates a server socket,
- Binds it to a port (e.g., 10101),
- Receives a message from a client and displays it.

**client:**
- Creates a client socket,
- Connect to server at <host> <port> (e.g., localhost and 10101)
- Sends "Hi" repeatedly out of this socket to the server.

**tcpServer.c**

```c
#include "def"
main()
{
    int sd, psd;
    struct sockaddr_in name;
    char   buf[1024];
    int    cc;

    sd = socket   (AF_INET,SOCK_STREAM,0);

    name.sin_family = AF_INET;
    name.sin_addr.s_addr = htonl(INADDR_ANY);
```
name.sin_port = htons(atoi(argv[1]));
binding = (SA *) &name, sizeof(name));
listen(sd, 1);
psd = accept(sd, 0, 0);
for(;;) {
    cc = recv(psd, buf, sizeof(buf), 0);
    if (cc == 0) exit(0);
    buf[cc] = NULL;
    printf("message received: %s\n", buf);
}

Usage example:

% tcpServer0 10101

tcpClient.c

#include "def"

main(argc, argv )
int argc;
char *argv[];
{
    int sd;
    struct sockaddr_in server;
    struct hostent *hp, *gethostbyname();
    sd = socket (AF_INET, SOCK_STREAM, 0);
    hp = gethostbyname(argv[1]);
    bcopy(h->h_addr, &h->h_addr, hp->h_length);
    server.sin_family = AF_INET;
server.sin_port = htons(atoi(argv[2]));
connect(sd, (SA *)&server, sizeof(server));
for (;;) {
    send(sd, "HI", 2, 0);
    printf("sent HI\n");
    sleep(2);
}

Usage example:
% tcpClient0 localhost 10101

Comprehensive tcp Client & Server

server:

✦ Creates a server socket,
✦ Binds it to a port. The port can be specified using either:
      • Argument to the program (e.g., argv[1]), or
      • Chosen by the system (0) and displayed after bind, using getsockname.
✦ Receives a message from a client and displays:
      • The received message,
      • The ip/name of the client, and
      • The port information of the client.
✦ Sends (echo back) the received message to its sender.

client:

✦ Creates a client socket and contacts the server using two arguments: <host> <port>
✦ Sends to the server a message typed by the user.
Recieves from the server the echoed message and displays it along with the ip/name and port information of the server.

TCPserver.c

main(...)
{

    /*get TCPServer1 Host information: NAME and INET ADDRESS*/
    gethostname(ThisHost, MAXHOSTNAME);
    printf("TCP/Server running at host NAME: %s\n", ThisHost);
    hp = gethostbyname(ThisHost);
    bcopy (hp->h_addr, &server.sin_addr, hp->h_length);
    printf(" (TCP/Server INET ADDRESS is: %s )\n", inet_ntoa(server.sin_addr));

    /* Construct name of socket */
    server.sin_family = AF_INET;
    server.sin_addr.s_addr = htonl(INADDR_ANY);
    if (argc == 1)
    {
        server.sin_port = htons(0)
    }
    else
    {
        server.sin_port = htons(atoi(argv[1]));
    }

    /* Create socket on which to send and receive */
    sd = socket(AF_INET,SOCK_STREAM,0);
    bind(sd, (SA *)&server, sizeof(server);

    /* get port information and prints it out */
    length = sizeof(server);
    getsockname(sd, (SA *)&server,&length);
    printf("Server Port is: %d\n", ntohs(server.sin_port));
/* accept TCP connections & fork a process to serve each client */
listen(sd,4);
fromlen = sizeof(from);
for(;;){
    psd  = accept(sd, (SA *) &from, &fromlen);
    childpid = fork();
    if ( childpid == 0) {
        close (sd);
        EchoServe(psd, from);
    }
    else{
        printf("My new child pid is %d\n", childpid);
        close(psd);
    }
}
}/*end of main*/

EchoServe(int psd, struct sockaddr_in from){
    ....

/* print client information */
    printf("Serving %s:%d\n",inet_ntoa(from.sin_addr),
           ntohs(from.sin_port));
    hp = gethostbyaddr((char *) &from.sin_addr.s_addr,
                       sizeof(from.sin_addr.s_addr),AF_INET));
    printf("(Name is : %s)\n", hp->h_name);

/* get data from clients and send it back */
for(;;){
    rc=recv(psd, buf, sizeof(buf), 0);
    if (rc > 0){
        buf[rc]=NULL;
        printf("Received: %s\n", buf);
        printf("From TCP/Client: %s:%d\n", 
               inet_ntoa(from.sin_addr),
               ntohs(from.sin_port));
        printf("(Name is : %s)\n", hp->h_name);
        send(psd, buf, rc, 0);
    }else {
        printf("Disconnected..\n");
        close (psd);
exit(0);
}

} /*End of EchoServ */

Usage example:

% TCPServer1

% TCPServer1 10101

TCPClient.c

main ( ..)
{
....

gethostname(ThisHost, MAXHOSTNAME);
printf("TCP/Cleint running at host NAME: %s\n", ThisHost);
hp = gethostbyname(ThisHost);
bcopy ( hp->h_addr, & (server.sin_addr), hp->h_length);
printf(" (TCP/Cleint INET ADDRESS is: %s )\n", inet_ntoa(server.sin_addr));

gethostname ( argv[1] ) == NULL ) {
    addr.sin_addr.s_addr = inet_addr(argv[1]);
hp = gethostbyaddr((char *) &addr.sin_addr.s_addr,
    sizeof(addr.sin_addr.s_addr),AF_INET);
}
printf("TCP/Server running at host NAME: %s\n", hp->h_name);
bcopy ( hp->h_addr, & (server.sin_addr), hp->h_length);
printf(" (TCP/Server INET ADDRESS is: %s )\n", inet_ntoa(server.sin_addr));

server.sin_family = AF_INET;
server.sin_port = htons(atoi(argv[2]));
/* Create socket on which to send and receive */
sd = socket (AF_INET, SOCK_STREAM, 0);

/** Connect to TCPServer1 */
connect(sd, (SA *) &server, sizeof(server));
fromlen = sizeof(from);
getpeername(sd, (SA *) &from, &fromlen);
printf("Connected to TCPServer1: ");
printf("%s:%d\n", inet_ntoa(from.sin_addr),
       ntohs(from.sin_port));
hp = gethostbyaddr((char *) &from.sin_addr.s_addr,
       sizeof(from.sin_addr.s_addr), AF_INET);
printf("(Name is : %s)\n", hp->h_name);
childpid = fork();
if (childpid == 0) {
    GetUserInput();
}

/* receive it from SERVER, display it back to USER */
for(;;) {
    recv(sd, rbuf, sizeof(rbuf), 0) ;
    printf(" Received: %s", rbuf);
}

} /* End of main */

/* get data from USER, send it SERVER */

GetUserInput(){

    for(;;) {
        printf("Type anything followed by RETURN, or type 
CTRL-D to exit\n");
        rc=read(0,buf, sizeof(buf));
        if (rc == 0) break;
        send(sd, buf, rc, 0);
    }
    printf ("EOF... exit\n");
    close(sd);
    kill(getppid(), 9);
    exit (0);
}

Usage example:
Concurrency using Select

\[
\text{n = select ( length, readset, writeset, exceptset, time) }
\]

values of time:

- **NULL**: indefinite wait
- **0**: no-wait (poll)
- **T**: wait T (sec + micro-sec)

A socket in readset is ready when:

- A connected socket has received data.
- A `listen` socket has a connection ready to accept.

Macros:

- `FD_ZERO(&mask);`
- `FD_SET(sd, &mask);`
- `FD_CLR(psd, &mask);`
- `if (FD_ISSET(sd, &mask)) ....`

TCP Server Sel.c

```c
main( ... )
```
/** accept TCP connections from clients and use select to serve each */
listen(sd, 4);
fromlen = sizeof(from);
for (i = 0; i < MAXCLIENTS; i++)
    client[i] = -1;  /* -1 indicates available
entry */
    FD_ZERO(&allset);
    FD_SET(sd, &allset);

for (;;) {
    readset = allset;  /* structure assignment */
    select(MAXCLIENTS, &readset, NULL, NULL, NULL);

    if (FD_ISSET(sd, &readset)) {  /* new client
connection */
        psd = accept(sd, (SA *) & from, &fromlen);
        for (i = 0; i < MAXCLIENTS; i++)
            if (client[i] < 0) {
                client[i] = psd;  /* save descriptor
*/
                FD_SET(psd, &allset);  /* add new
descriptor to set */
            }
        break;
    }
    if (i == MAXCLIENTS) {
        printf("too many clients");
        exit(0);
    }
}
for (i = 0; i <= MAXCLIENTS; i++) {  /* check
all clients for data */
    if ((psd = client[i]) < 0)
        continue;
    if (FD_ISSET(psd, &readset))
        EchoServe(psd, from, i);
}
}

EchoServe(psd, from, i)
int psd;
struct sockaddr_in from;
int i;
{
    /** get data from clients and send it back */
    printf("\n...server is waiting...\n");
    if ((rc = recv(psd, buf, sizeof(buf), 0)) < 0) {
        perror("receiving stream message");
    }
    if (rc > 0) {
        if (send(psd, buf, rc, 0) < 0)
            perror("sending stream message");
    } else {
        printf("Disconnected..\n");
        FD_CLR(psd, &allset);
        client[i] = -1;
        close(psd);
    }
}