A TCP Client for the ECHO Service

```
/* TCPecho.c - main, TCPecho */

#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

extern int errno;

int TCPecho(const char *host, const char *service);
int errexit(const char *format, ...);
int connectTCP(const char *host, const char *service);

#define LINELEN 128

/ *------------------------------------------------------
  * main - TCP client for ECHO service
  *------------------------------------------------------
  */

int main(int argc, char *argv[])
{
        char *host = "localhost";       /*host to use if none supplied */
        char *service = "echo";         /*default service name */

        switch (argc) {
        case 1:
                host = "localhost";
                break;
        case 3:
                service = argv[2];
                /*FALL THROUGH*/
        case 2:
                host = argv[1];
                break;
```

```
        default:
                fprintf(stderr, "usage: TCPecho [host [port]]\n");
                exit(1);
        }
        TCPecho(host, service);
        exit(0);
}

/ *-------------------------------------------------------------------------------
 * TCPecho - send input to ECHO service on specified host and print reply
 *-------------------------------------------------------------------------------
 */

 int TCPecho(const char *host, const char *service)
 {
        char buf[LINELEN + 1];          /* buffer for one line of text */
        int s, n;                       /* socket descriptor, read count */
        int outchars, inchars;          /* characters sent and received */

        s = connectTCP(host ,service);

        while (fgets(buf, sizeof(buf), stdin)) {
                buf[LINELEN] = '\0';    /* insure line null-terminated */
                outchars = strlen(buf);
                (void) write(s, buf, outchars);

                /* read it back */
                for(inchars = 0; inchars < outchars; inchars += n ){
                        n = read(s, &buf[inchars], outchars - inchars);
                        if( n < 0){
                                errexit("socket read falied: %s\n", strerror(errno));
                        }
                }
                fputs(buf, stdout);
        }
 }
```

A UDP Client for the ECHO Service

```
/* UDPecho.c - main, UDPecho */

#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

extern int errno;

int UDPecho(const char *host, const char *service);
int errexit(const char *format, ...);
int connectUDP(const char *host, const char *service);

#define LINELEN 128

/ *------------------------------------------------
  * main - UDP client for ECHO service
  *------------------------------------------------
  */

int main(int argc, char *argv[])
{
        char *host = "localhost";
        char *service = "echo";

        switch (argc) {
        case 1:
                host = "localhost";
                break;
        case 3:
                service = argv[2];
                /*FALL THROUGH*/
        case 2:
                host = argv[1];
                break;
```

```
        default:
                fprintf(stderr, "usage: UDPecho [host [port]]\n");
                exit(1);
        }
        UDPecho(host, service);
        exit(0);

}

/ *-------------------------------------------------------------------------------
 * UDPecho - send input to ECHO service on specified host and print reply
 *-------------------------------------------------------------------------------
 */

 int UDPecho(const char *host, const char *service)
 {
        char buf[LINELEN + 1];          /*buffer for one line of text */
        int s, nchars;                  /*socket descriptor, read count */

        s = connectUDP(host ,service);

        while (fgets(buf, sizeof(buf), stdin)) {
                buf[LINELEN] = '\0';    /* insure line null-terminated */
                nchars = strlen(buf);
                (void) write(s, buf, nchars);

                if( read(s, buf, nchars)< 0)
                        errexit("socket read falied: %s\n", strerror(errno));

                fputs(buf, stdout);
        }
 }
```