```
:::::::::::::: fopendemo.c ::::::::::::::

#include       <stdio.h>

/* fopendemo.c
 *      purpose: a simple review of how fopen works
 *      Notes:   fopen() returns a FILE *
 *               fgets(), getc(), fscanf() can be used
 *               need fclose() when done
 */

main()
{
        FILE    *fp;
        char    b[512];

        fp = fopen( "/etc/passwd", "r" );
        while( fgets( b, 512, fp ) != NULL )
                fprintf(stdout, "%s", b );
        fclose(fp);
}

:::::::::::::: popendemo.c ::::::::::::::
#include       <stdio.h>

/*
 * popendemo.c
 *      demonstrates how to open a program for standard i/o
 *      important points:
 *              1. popen() returns a FILE *, just like fopen()
 *              2. the FILE * it returns can be read/written
 *                 with all the standard functions
 *              3. you need to use pclose() when done
 */
main()
{
        FILE    *fp, *popen();
        char    buf[100];
        int     i = 0;


        fp = popen( "who", "r" );

        while ( fgets( buf, 100, fp ) != NULL )
                printf("%3d %s", i++, buf );

        pclose( fp );
}

:::::::::::::: popen_ex2.c ::::::::::::::
#include       <stdio.h>

/*
 *      popen_ex2.c
 *              shows how to use popen() to read from rwho
 *              to get list of all users on local network
 *              Note: could be useful for an expanded version
 *              of watch.c
 */

main()
{
        FILE    *fp;
        char    b[512];

        fp = popen( "rwho", "r" );
        while( fgets( b, 512, fp ) != NULL )
                fprintf(stdout, "%s", b );
        pclose(fp);
}
```

```
:::::::::::::: popen_ex3.c ::::::::::::::
#include       <stdio.h>

/*
 * popen_ex3.c
 *      shows how to use popen to write to a process that
 *      reads from stdin.  This program writes email to
 *      two users.  Note how easy it is to use fprintf
 *      to format the data to send.
 */

main()
{
       FILE    *fp;

       fp = popen( "mail user1 user2", "w" );
       fprintf( fp, "hello...how are you?\n" );
       pclose( fp );
}

:::::::::::::: timeserv.c ::::::::::::::
#include  <stdio.h>
#include  <sys/types.h>
#include  <sys/socket.h>
#include  <netinet/in.h>
#include  <netdb.h>

#define   PORTNUM  8822       /* our time service phone number */
#define   oops(msg)      { perror(msg) ; exit(1) ; }


void main(ac, av)
char **av;
{
        struct  sockaddr_in   saddr;   /* build our address here */
        struct  hostent       *hp;   /* this is part of our    */
        char    hostname[256];         /* address              */
        int     slen,sock_id,sock_fd;  /* line id, file desc     */
        FILE    *sock_fp;              /* use socket as stream   */
        char    *ctime();              /* convert secs to string */
        long    time(), thetime;       /* time and the val       */


       /*
        *      step 1: build our network address
        *               domain is internet, hostname is local host,
        *               port is some arbitrary number
        */

       gethostname( hostname, 256 );           /* where am I ?         */
       hp = gethostbyname( hostname );         /* get info about host  */

       bzero( &saddr, sizeof(saddr) );         /* zero struct          */
                                               /* fill in hostaddr     */
       bcopy( hp->h_addr, &saddr.sin_addr, hp->h_length);
       saddr.sin_family = AF_INET ;            /* fill in socket type  */
       saddr.sin_port = htons(PORTNUM);        /* fill in socket port  */

       /*
        *      step 2: ask kernel for a socket, then bind address
        */

       sock_id = socket( AF_INET, SOCK_STREAM, 0 );    /* get a socket */
       if ( sock_id == -1 ) oops( "socket" );

       if ( bind(sock_id, &saddr, sizeof(saddr)) != 0 )/* bind it to   */
              oops( "bind" );                          /* an address   */

       /*
        *      step 3: tell kernel we want to listen for calls
        */
```

```
        if ( listen(sock_id, 1) != 0 ) oops( "listen" );

        while ( 1 ){
                printf("Wow! got a call!\n");
                sock_fd = accept(sock_id, NULL, NULL); /* wait for call */
                if ( sock_fd == -1 )
                        oops( "accept" );       /* error getting calls  */

                sock_fp = fdopen(sock_fd,"w");  /* we'll write to the   */
                if ( sock_fp == NULL )          /* socket as a stream   */
                        oops( "fdopen" );       /* unless we can't      */

                thetime = time(NULL);           /* get time             */
                                                /* and convert to strng */
                fprintf( sock_fp, "%s", ctime(&thetime) );
                fclose( sock_fp );              /* release connection   */
        }
}

:::::::::::::: timeclnt.c ::::::::::::::
#include       <stdio.h>
#include       <sys/types.h>
#include       <sys/socket.h>
#include       <netinet/in.h>
#include       <netdb.h>

#define        HOSTNAME        "host2"
#define        PORTNUM         8822
#define        oops(msg)       { perror(msg); exit(1); }

main()
{
        struct sockaddr_in  servadd;        /* the number to call */
        struct hostent      *hp;            /* used to get number */
        int    sock_id, sock_fd;            /* the socket and fd  */
         char   message[BUFSIZ];             /* to receive message */
         int    messlen;                     /* for message length */

        /*
         *      build the network address of where we want to call
         */

        hp = gethostbyname( HOSTNAME );
        if ( hp == NULL ) oops("no such computer");

        bzero( &servadd, sizeof( servadd ) );   /* zero the address     */
        servadd.sin_family = AF_INET ;          /* fill in socket type  */
                                                /* and machine address  */
        bcopy( hp->h_addr, &servadd.sin_addr, hp->h_length);
        servadd.sin_port = htons(PORTNUM);      /* host to num short    */

        /*
         *        make the connection
         */

        sock_id = socket( AF_INET, SOCK_STREAM, 0 );    /* get a line   */
        if ( sock_id == -1 ) oops( "socket" );          /* or fail      */
                                                        /* now dial     */
        if ( connect( sock_id, &servadd, sizeof(servadd)) !=0  )
                oops( "connect" );

        /*
         *      we're connected to that number, read from socket
         */

        messlen = read( sock_id, message, BUFSIZ );     /* read stuff   */
        if ( messlen == - 1 )
                oops( "read" ) ;
        if ( write( 1, message, messlen ) != messlen )  /* and write to */
                oops( "write" );                        /* stdout       */
        close( sock_id );
}
```

```
:::::::::::::: rem_execd.c ::::::::::::::

#include       <stdio.h>
#include       <sys/types.h>
#include       <sys/socket.h>
#include       <netinet/in.h>
#include       <netdb.h>

#define PORTNUM 2001
#define oops(msg)      { perror(msg) ; exit(1) ; }


void main(ac, av)
char **av;
{

        struct  sockaddr_in    saddr;  /* a struct to hold a socket address */
        struct  hostent        *hp;
        char    hostname[256];         /* get hostname here   */
        int     slen, sock_id, sock_fd, ch;
        FILE    *sock_fp, *cmdfp;
        char    cmd[BUFSIZ];           /* command to execute */
        int     cmdlen;

        /*
         *      step 1: build our network address
         *              domain is internet, hostname is local host,
         *              port is some arbitrary number
         */

        gethostname( hostname, 256 );         /* where am I ?               */
        hp = gethostbyname( hostname );               /* get info about host */

        bzero( &saddr, sizeof(saddr) );               /* zero struct         */
                                              /* fill in hostaddr    */
        bcopy( hp->h_addr, &saddr.sin_addr, hp->h_length);
        saddr.sin_family = AF_INET ;          /* fill in socket type */
        saddr.sin_port = htons(PORTNUM);      /* fill in socket port */

        /*
         *      step 2: ask kernel for a socket, then bind address
         */

        sock_id = socket( AF_INET, SOCK_STREAM, 0 ); /* get a socket       */
        if ( sock_id == -1 ) oops( "socket" );
        if ( bind(sock_id, &saddr, sizeof(saddr)) != 0 )/* bind it to       */
               oops( "bind" );                                /* an address  */

        /*
         *      step 3: tell kernel we want to listen for calls
         */

        if ( listen(sock_id, 1) != 0 ) oops( "listen" );
        while ( 1 ){
               sock_fd = accept(sock_id, NULL, NULL); /* wait for call */
               if ( sock_fd == -1 )
                      oops( "accept" );      /* error getting calls */

               cmdlen = read( sock_fd, cmd, BUFSIZ );
               if ( strncmp( cmd, "quit", 4 ) == 0 )
                      exit(0);
               if ( cmdlen == -1 ) oops( "read" );
               cmd[cmdlen] = '\0';

               sock_fp = fdopen(sock_fd,"w");        /* we'll write to the  */
               if ( sock_fp == NULL )         /* socket as a stream  */
                      oops( "fdopen" );       /* unless we can't     */

               cmdfp = popen( cmd, "r" );     /* and read from the   */
               if ( cmdfp == NULL )           /* command             */
                      fputs( "didnt work\n", sock_fp);
               else {
                      while ( (ch = getc( cmdfp )) != EOF )
                             putc(ch, sock_fp );
                      pclose( cmdfp );
               }
               fclose( sock_fp );             /* release connection  */
        }
}
```

```
:::::::::::::: rem_execc.c ::::::::::::::
#include       <stdio.h>
#include       <sys/types.h>
#include       <sys/socket.h>
#include       <netinet/in.h>
#include       <netdb.h>

#define        HOSTNAME       "host2"
#define        PORTNUM        2001
#define        oops(msg)      { perror(msg); exit(1); }

main()
{

        struct sockaddr_in rxa;
        struct hostent          *hp;
        FILE    *sock_fp;
        int     sock_id, sock_fd, ch;
        char    cmd[BUFSIZ];

        /*
         *      build the network address of where we want to call
         */

        hp = gethostbyname( HOSTNAME );
        if ( hp == NULL ) oops("no such computer");

        bzero( &rxa, sizeof( rxa ) );         /* zero the address    */
        rxa.sin_family = AF_INET ;            /* fill in socket type */
                                              /* and machine address */
        bcopy( hp->h_addr, &rxa.sin_addr, hp->h_length);
        rxa.sin_port = htons(PORTNUM);               /* format number      */

        /*
         *      make the connection
         */

        sock_id = socket( AF_INET, SOCK_STREAM, 0 ); /* get a line  */
        if ( sock_id == -1 ) oops( "socket" );              /* or fail     */
        if ( connect( sock_id, &rxa, sizeof(rxa))!=0 )      /* dial number */
                oops( "connect" );

        /*
         *      we're connected to that number,
         *      open socket as a readable stream and copy to stdout
         */

        gets( cmd );
        if ( write( sock_id, cmd, strlen(cmd) ) == -1 )
                oops( "write" );
        sock_fp = fdopen( sock_id , "r" );           /* open to read       */
        if ( sock_fp == NULL )
                oops( "fdopen" ) ;
        while ( (ch = getc(sock_fp)) != EOF )        /* copy from   */
                putchar( ch );                       /* stream      */
        fclose( sock_fp );
}
```