INTERNATIONAL ATOMIC ENERGY AGENCY
UNITED NATIONS EDUCATIONAL, SCIENTIFIC AND CULTURAL ORGANIZATION
INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS
I.C.T.P., P.O. BOX 586, 34100 TRIESTE, ITALY, CABLE: CENTRATOM TRIESTE

UNITED NATIONS INDUSTRIAL DEVELOPMENT ORGANIZATION

INTERNATIONAL CENTRE FOR SCIENCE AND HIGH TECHNOLOGY
c/o INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS 34100 TRIESTE (ITALY) VIA GRIGNANO, 9 (ADRIATICO PALACE) P.O. BOX 586 TELEPHONE 040-224572 TELEFAX 040-224575 TELEX 460449 APH I

SMR/643 - 14

SECOND COLLEGE ON MICROPROCESSOR-BASED REAL-TIME CONTROL - PRINCIPLES AND APPLICATIONS IN PHYSICS
5 - 30 October 1992

Additional material to lectures presented by

L. WEBER
Observatoire de Geneve
Ch. des Maillettes 51
1290 Sauverny
Switzerland

These are preliminary lecture notes, intended only for distribution to participants.

```
/*
 * inetsockread.c
 *
 * This program creates a socket in the internet domain.
 * After printing the socket's port it begins an infinite loop.
 * Each time through the loop it accepts a connection and prints out messages
 * from it. When the connection breaks, the program accepts a new connection.
 * If the termination message comes through, the program closes the connection
 * and exit.
 *
 * The form of the command line is:
 *      inetsockread
 */

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

#define TRUE 1
#define BUFSIZE 1024

extern int      errno;          /* system error code */
extern char    *sys_errlist[];  /* system error messages */

main(argc, argv)
int             argc;
char           *argv[];

{
        int             rval, sock, length;
        struct sockaddr_in sockin;
        int             msgsock;
        char            buf[BUFSIZE];
        int             i;

        /* Create an Internet domain stream socket */
        sock = socket(AF_INET, SOCK_STREAM, 0);
        if (sock < 0)
        {
                fprintf(stderr, "%s: error opening stream socket - %s\n",
                                        argv[0], sys_errlist[errno]);
                exit(1);
        }

        /* Name the socket using wildcards */
        sockin.sin_family = AF_INET;
        sockin.sin_addr.s_addr = INADDR_ANY;
        sockin.sin_port = 0;
        if (bind(sock, &sockin, sizeof(sockin)))
        {
                perror("birding stream socket");
                exit(1);
        }

        /* Find out assigned port number and print it out */
        length = sizeof(sockin);
        if (getsockname(sock, &sockin, &length))
        {
                perror("getting socket name");
                exit(1);
        }

        printf("%s: listening for connections on socket port# %d...\n",
                argv[0], ntohs(sockin.sin_port));

        /*
         * Start listening for connections on sock.
         * The maximum length of the queue of pending connections is set to 5.
         * If a connection request arrives with the queue full, the client
         * will receive an error with an indication of ECONNREFUSED.
         */
        if ( listen(sock, 5) < 0 )
```



```
        {
                fprintf(stderr, "%s: error listening stream socket - %s.\n",
                                argv[0], sys_errlist[errno]);
                exit(2);
        }

        do
        {
                /* Wait for someone to talk to us */
                msgsock = accept(sock, 0, 0);
                if (msgsock == -1)
                {
                        close(sock);
                        fprintf(stderr,
                                "%s: error accept - %s.\n",
                                argv[0], sys_errlist[errno]);
                        exit(2);
                }
                do
                {
                        bzero(buf, sizeof(buf));
                        if ((rval = read(msgsock, buf, BUFSIZE)) < 0)
                        {
                                close(msgsock);
                                close(sock);
                                fprintf(stderr,
                                        "%s: error reading stream message - %s.\n"
,
                                        argv[0], sys_errlist[errno]);
                                exit(2);
                        }
                        if (rval == 0)
                                printf("%s: Ending connection.\n", argv[0]);
                        else
                        {
                                printf("-->%s", buf);
                                if ( strcmp(buf, ".\n") == 0 )
                                {
                                        printf("%s: Closing connection.\n",
                                                argv[0]);
                                        close(msgsock);
                                        close(sock);
                                        unlink(argv[1]);
                                        return(0);
                                }
                        }
                } while (rval != 0);
                close(msgsock);
        } while (TRUE);

        /*
         * Since this program has an infinite loop, the socket "sock" is
         * never explicitly closed.  However, all sockets will be closed
         * automatically when a process is killed or terminates normally.
         */
}
```

```
/*
 * inetsockwrite.c
 *
 * This program connects to the internet domain socket named in the command
 * line and sends every characters read from the standard input to that socket,
 * until a line with a single dot is read or an EOF occurs.
 * Finally the socket is closed, ending the connection.
 *
 * The form of the command line is:
 *      inetsockwrite hostname portnumber < file
 */

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

#define BUFSIZE 1024

extern int      errno;          /* system error code */
extern char    *sys_errlist[];  /* system error messages */

main(argc, argv)
int             argc;
char           *argv[];

{
        int             rval, sock;
        struct sockaddr_in sockin;
        struct hostent *hp, *gethostbyname();
        char            buf[BUFSIZE];

        /* Test if required arguments are presents */
        if ( argc < 3 )
        {
                fprintf(stderr, "usage: %s hostname portnumber.\n", argv[0]);
                exit(1);
        }

        /* Create an Internet domain stream socket */
        sock = socket(AF_INET, SOCK_STREAM, 0);
        if (sock < 0)
        {
                fprintf(stderr, "%s: error opening stream socket - %s\n",
                                argv[0], sys_errlist[errno]);
                exit(1);
        }

        /*
         * Get host information from the network host data base '/etc/hosts'.
         */
        hp = gethostbyname(argv[1]);
        if (hp == (struct hostent *) NULL)
        {
                close(sock);
                fprintf(stderr, "%s: unknown host\n", argv[1]);
                exit(2);
        }

        /*
         * Connect socket using hostname and portnumber specified by the
         * command line arguments.
         */
        sockin.sin_family = AF_INET;
        bcopy(hp->h_addr, &sockin.sin_addr, hp->h_length);
        /*
         * htons convert is used to convert portnumber from host byte
         * order to network byte order.
         */
        sockin.sin_port = htons(atoi(argv[2]));
        if (connect(sock, &sockin, sizeof(sockin)) < 0)
        {
                close(sock);
```



```
                fprintf(stderr, "%s: error connecting stream socket - %s\n",
                                argv[0], sys_errlist[errno]);
                exit(1);
        }

        /*
         * Read characters from standard input and write them on the
         * socket until end of file.
         */
        while ((rval = read(0, buf, BUFSIZE)) > 0)
        {
                if (write(sock, buf, rval) < 0)
                {
                        close(sock);
                        fprintf(stderr,
                                "%s: error writing on stream socket - %s\n",
                                argv[0], sys_errlist[errno]);
                        exit(1);
                }
                if ( strncmp(buf, ".\n", 2) == 0 )
                        break;
        }

        printf("%s: Ending connection.\n", argv[0]);
        close(sock);
        return(0);
}
```

```
/*
 * unixsockread.c
 *
 * This program creates a socket in the UNIX domain and binds the name
 * specified on the command line to it.
 * After printing the socket's name it begins an infinite loop.
 * Each time through the loop it accepts a connection and prints out messages
 * from it.  When the connection breaks, the program accepts a new connection.
 * If the termination message comes through, the program closes the connection
 * and exit.
 *
 * The form of the command line is:
 *      unixsockread pathname
 */

#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <stdio.h>

#define BUFSIZE 1024

extern int      errno;          /* system error code */
extern char    *sys_errlist[];  /* system error messages */

main(argc, argv)
int             argc;
char           *argv[];

{
        int             sock, msgsock, rval;
        struct sockaddr_un sockun;
        char            buf[BUFSIZE];

        /* Test if required arguments are presents */
        if ( argc < 2 )
        {
                fprintf(stderr, "usage: %s pathname.\n", argv[0]);
                exit(1);
        }

        /* Create an Unix domain stream socket */
        sock = socket(AF_UNIX, SOCK_STREAM, 0);
        if (sock < 0)
        {
                fprintf(stderr, "%s: error opening stream socket - %s.\n",
                                argv[0], sys_errlist[errno]);
                exit(2);
        }

        /* Name the socket using filesystem name specified by command line. */
        sockun.sun_family = AF_UNIX;
        strcpy(sockun.sun_path, argv[1]);
        if (bind(sock, &sockun, sizeof(struct sockaddr_un)))
        {
                close(sock);
                fprintf(stderr,
                        "%s: error binding stream socket '%s' - %s.\n",
                        argv[0], sockun.sun_path, sys_errlist[errno]);
                exit(2);
        }

        printf("%s: listening for connections on socket '%s'...\n",
                argv[0], sockun.sun_path);

        /*
         * Start listening for connections on sock.
         * The maximum length of the queue of pending connections is set to 5.
         * If a connection request arrives with the queue full, the client
         * will receive an error with an indication of ECONNREFUSED.
         */
        if ( listen(sock, 5) < 0 )
        {
                fprintf(stderr, "%s: error listening stream socket - %s.\n",
```



```
                                argv[0], sys_errlist[errno]);
                exit(2);
        }

        for (;;)
        {
                /* Wait for someone to talk to us */
                msgsock = accept(sock, 0, 0);
                if (msgsock == -1)
                {
                        close(sock);
                        fprintf(stderr,
                                "%s: error accept - %s.\n",
                                argv[0], sys_errlist[errno]);
                        exit(2);
                }

                do
                {
                        bzero(buf, sizeof(buf));
                        if ((rval = read(msgsock, buf, BUFSIZE)) < 0)
                        {
                                close(msgsock);
                                close(sock);
                                fprintf(stderr,
                                        "%s: error reading stream message - %s.\n"
,
                                        argv[0], sys_errlist[errno]);
                                exit(2);
                        }

                        if (rval == 0)
                                printf("%s: Ending connection.\n", argv[0]);
                        else
                        {
                                printf("-->%s", buf);
                                if ( strcmp(buf, ".\n") == 0 )
                                {
                                        printf("%s: Closing connection.\n",
                                                argv[0]);
                                        close(msgsock);
                                        close(sock);
                                        unlink(argv[1]);
                                        return(0);
                                }
                        }
                }
                while (rval > 0);
                close(msgsock);
        }

        /*
         * The following statements are not executed, because they follow an
         * infinite loop.  However, most ordinary programs will not run
         * forever.  In the UNIX domain it is necessary to tell the file
         * system that one is through using argv[1].  In most programs one uses
         * the call unlink() as below. Since the user will have to kill this
         * program, it will be necessary to remove the name by a command from
         * the shell.
         */
        close(sock);
        unlink(argv[1]);
        return(0);
}
```

```
/*
 * unixsockwrite.c
 *
 * This program connects to the unix domain socket named in the command line
 * and sends every characters read from the standard input to that socket,
 * until a line with a single dot is read or an EOF occurs.
 * Finally the socket is closed, ending the connection.
 *
 * The form of the command line is:
 *      unixsockwrite pathname < file
 */

#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <stdio.h>

#define BUFSIZE 1024

extern int      errno;          /* system error code */
extern char    *sys_errlist[];  /* system error messages */

main(argc, argv)
int             argc;
char           *argv[];

{
        int             rval, sock;
        struct sockaddr_un sockun;
        char            buf[BUFSIZE];

        /* Test if required arguments are presents */
        if ( argc < 2 )
        {
                fprintf(stderr, "usage: %s pathname.\n", argv[0]);
                exit(1);
        }

        /* Create an Unix domain stream socket */
        sock = socket(AF_UNIX, SOCK_STREAM, 0);
        if (sock < 0)
        {
                fprintf(stderr, "%s: error opening stream socket - %s.\n",
                                argv[0], sys_errlist[errno]);
                exit(2);
        }

        /* Connect socket using pathname specified by command line. */
        sockun.sun_family = AF_UNIX;
        strcpy(sockun.sun_path, argv[1]);
        if (connect(sock, &sockun, sizeof(struct sockaddr_un)) < 0)
        {
                close(sock);
                fprintf(stderr,
                        "%s: error connecting stream socket '%s' - %s.\n",
                        argv[0], sockun.sun_path, sys_errlist[errno]);
                exit(2);
        }

        /*
         * Read characters from standard input and write them on the
         * socket until end of file.
         */
        while ((rval = read(0, buf, BUFSIZE)) > 0)
        {
                if (write(sock, buf, rval) < 0)
                {
                        close(sock);
                        fprintf(stderr,
                                "%s: error writing on stream socket - %s.\n",
                                argv[0], sys_errlist[errno]);
                        exit(2);
                }
                if ( strncmp(buf, ".\n", 2) == 0 )
```



```
                        break;
        }

        printf("%s: Ending connection.\n", argv[0]);
        close(sock);
        return(0);
}
```