Chapter 4

Socket Programming Examples

This chapter illustrates AF_HIPPI socket programming with an example of a Mandelbrot set calculation.

The sample program consists of two components — a CM Fortran program, `mandel`, which performs the Mandelbrot calculations and a C routine, `setup_hippi_fb`, which `mandel` calls. `setup_hippi_fb` creates and initializes a socket, which mandel uses to send packets to a framebuffer device.

These components are discussed, in code segments, on the following pages. Both components are also presented in their complete, unsegmented form in Figures 21 and 22.

Copyright © 1994 Thinking Machines Corporation

mandel

1

```
c
c    mandelbrot set calculation
c    042288    created      thorp/alex
c
       program mandel

       parameter (si=3.0)
       parameter (left_side=-2.5)
       parameter (top_side =-1.2)
       integer res
       parameter (res = 256)

       real zc(1280,1024), zi(1280,1024)      ! real and imaginary components of z

       real zcs(1280,1024), zis(1280,1024)    ! square of real and imaginary z

       real cc(1280,1024), ci(1280,1024)      ! real and imaginary components of c
       real cell_size                         ! size of each pixel

       integer ittr(1280,1024)                ! # of cycles before cell warped out
       integer pixel(1280,1024)               ! output pixel
       integer temp(1280,1024)                ! output pixel
       integer i                              ! loop counter
```

2

```
       call setup_hippi_fb(ifd)
       if (ifd < 0) then
         print *, 'open of framebuffer failed'
         stop
       endif
```

3

```
       call CMF_FILE_FDOPEN(9, ifd, istatus)
       if (istatus < 0 ) then
         print *,'setup of framebuffer failed', istatus
         call CMFS_PERROR("open failure")
         stop
       endif
```

4

```
  5    ittr = 0
       zc   = 0
       zi   = 0
       zcs  = 0
       zis  = 0
```


Copyright © 1994 Thinking Machines Corporation

Segment 1 initializes the program variables.

Segment 2 calls `setup_hippi_fb`, which returns a socket descriptor for the socket it creates. If `setup_hippi_fb` fails, it returns a `-1`.

Segment 3 associates the socket descriptor returned by `setup_hippi_fb` with a CM Fortran unit number. unit is used later in the program as an argument to `CMF_CM_ARRAY_TO_FILE`.

Segment 4 zeros the seed values for the Mandelbrot calculation, including the iteration count.

NOTE: See the *CM Fortran Utility Library Reference Manual* for detailed descriptions of `CMF_CM_ARRAY_TO_FILE` and other CM Fortran routines used in `mandel`.

Copyright © 1994 Thinking Machines Corporation

mandel (continued)

5

```
c    find the size of each cell
       cell_size = si/float(1280)

c    initialize cc
   7    forall(i=1:1280) cc(i,:) = float(i-1) * cell_size + left_side

c    initialize ci
       forall(i=1:1024) ci(:,i) = float(i-1) * cell_size + top_side

       print *,'Initialized arrays'
```

6

```
c    main loop here
       do 12 i= 1,res - 1
         where (zcs + zis .le. 4.0)
           zi = 2.0 * zc * zi + ci
           zc = zcs - zis + cc
           zcs = zc * zc
           zis = zi * zi
           ittr = i
         end where
```

7

```
         if (MOD(i,2) == 0) then
           where (ittr == MAXVAL(ittr))
             temp = 0
           elsewhere
             temp = ittr
           endwhere
           pixel = ISHFT(MIN((3*255*temp)/i, 255), 16) +
      $                ISHFT(MIN((380*temp)/i, 255), 8) +
      $                ISHFT(MAX(((3*255*temp)/i) - 512, 0), 0
```


Copyright © 1994 Thinking Machines Corporation

Segment 5 computes the individual cell size and initializes the arrays.

Segment 6 contains the program's main loop.

Segment 7 computes the value of each pixel to be displayed, determining the order in which they are to be displayed. `temp` is used to display a blank where `ittr` has reached its maximum.

Copyright © 1994 Thinking Machines Corporation

mandel (continued)

8

```
            call CMF_CM_ARRAY_TO_FILE(9, pixel, istatus)
            if (istatus < 0) then
              print *,'Write of array failed'
              call CMFS_PERROR("write failure")
              stop
            endif
            print *,'wrote frame', i
          endif
```

9

```
 12   continue
      goto 7
```

10

```
      call CMF_FILE_CLOSE(9,istatus)
      if (istatus < 0) then
        print *,'CMF_FILE_CLOSE failed'
        call CMFS_PERROR("close failed")
      endif
      stop
      end
```


Copyright © 1994 Thinking Machines Corporation

Segment 8 calls `CMF_CM_ARRY_TO_FILE` utility, which writes the array generated in segments 6 through 8 to the socket associated with unit.

Segment 9 returns the outer loop to the entry for array initialization.

Segment 10 closes the file specified by `unit`, which in turn causes the associated socket to be closed.

Copyright © 1994 Thinking Machines Corporation

setup_hippi_fb

1

```
#include <sys/types.h>
#include <sys/socket.h>
#include <nethippi/hippi.h>
#include <sys/errno.h>
#include <stdio.h>

#ifndef errormsg
/*****************************************************************************
 *
 *  errormsg
 *
 * A macro which turns errno into a string (reliably).  This macro
 * may be used, e.g.,:
 *
 *   if((fd = open(file, mode)) < 0) {
 *       fprintf(stderr, "%s: can't open %s for reading; %s\n",
 *         progname, file, errormsg);
 *
 * and therefore makes for far more useful error messages than can be
 * produced using perror()
 *
 *****************************************************************************/
extern int errno;
extern int sys_nerr;
extern char *sys_errlist[];

#define errormsg \
  (((errno < 0) || (errno > sys_nerr))? \
  "Errno out of range": sys_errlist[errno])
#endif /* errormsg */
```


Copyright © 1994 Thinking Machines Corporation

Segment 1 references (includes) the necessary header files and initializes the program variables.

Copyright © 1994 Thinking Machines Corporation

setup_hippi_fb (continued)

2

```
/*
 * Set up a HIPPI connection to the framebuffer,
 * and return (in argument "fd") the CMFS file descriptor to use
 * for Fortran writes to it.
 */
setup_hippi_fb_(fd)
int *fd;
{
    char *getenv();
    int ifield;
    char *p;
    int sock;
    u_int sndbuf , arglen ;
    struct sockaddr_hippi client;
```

3

```
 /*
  * Get a socket for accessing the HIPPI.
  */
 sock = socket(AF_HIPPI, SOCK_RAW, 0);
 if (sock < 0) {
       perror("socket AF_HIPPI SOCK_RAW");
       *fd = -1;
       return -1;
 }
```

4

```
    /*
     * If the environment variable HIPPI_FB_IFIELD is set,
     * use it for the IFIELD of the framebuffer.  Otherwise use
     * "2", which happens to be where the framebuffer is on our switch.
     */
    if ((p = getenv("HIPPI_FB_IFIELD")) == NULL)
          ifield = 0x2;
    else
          sscanf(p, "%i", &ifield);
```

5

```
    /*
     * Set the "Start of frame" bit in the vendor unique part
     * of the Ifield.  This is specific to the IOSC framebuffer.
     */
    if (! (ifield & 0x20000000))
         ifield |= 0x20000000;
```


Copyright © 1994 Thinking Machines Corporation

Segment 2 establishes the HIPPI connection to the framebuffer and returns the file descriptor (`fd`) that `mandel` will use when writing to it.

Segment 3 creates a PH socket.

Segment 4 uses the `ifield` value specified by the environment variable `HIPPI_FB_IFIELD`. If this environment variable is not set, it defaults to 2.

Segment 5 specifies a ”start of frame” value (2) in the `ifield`’s vendor unique field. This value is specific to the IOSC framebuffer.

Copyright © 1994 Thinking Machines Corporation

setup_hippi_fb (continued)

6

```
/*
 * Set the socket send buffer size.
 * This value determines the max write size.
 * Again, use environment variable if set, otherwise default to
 * 128 MB.
 */
if ((p = getenv("HIPPI_MAX_SNDBUF_SIZE")) == NULL)
      sndbuf = 128 * 1024 *1024 ; /* default value */
else
      sscanf(p, "%i", &sndbuf); /* use env. var value if set */

arglen = sizeof(sndbuf) ;
```

7

```
/*
 * Make request to set socket snd_buf size.
 */
if (setsockopt(sock, SOL_SOCKET, SO_PAR_SNDBUF, &sndbuf, arglen) <0) {
      perror("setsockopt parallel write");
      close(sock);
      *fd = -1;
      return -1;
}
```

8

```
      /* Get the actual value allowed. */
if (getsockopt(sock, SOL_SOCKET, SO_PAR_SNDBUF, &sndbuf, &arglen) < 0)
  {
    perror("getsockopt parallel write");
    exit(2);
  }

 /*
  * print what it actually got set to (the os may have limited the max size)
  */
 fprintf(stderr, "Max PARALLEL write size : %d (0x%x)\n", sndbuf, sndbuf) ;
```


Copyright © 1994 Thinking Machines Corporation

Segment 6 sets the socket send buffer size, which determines the maximum size of each write packet. If the `HIPPI_MAX_SNDBUF_SIZE` environment variable is set, use its value. Otherwise, default to 128 megabytes.

NOTE: The socket send buffer size should be the same or larger than the framebuffer size. If memory limitations require it to be smaller than the framebuffer size, the packets sent to the framebuffer must be constructed so that the sum of *N* packets equals the framebuffer size. In this program example, the vendor unique bits in the `ifield` for the first packet in each *N*-set of packets must be set to 2, which tells the IOSC framebuffer to return to the top of the frame. All other `ifields` must have these bits set to 0. This is necessary so that the framebuffer returns to the top of the frame only at the beginning of each set of frame data.

Segment 7 issues a request to set the socket send buffer size.

Segment 8 checks to see the actual socket send buffer size allocation and prints the value.

Copyright © 1994 Thinking Machines Corporation

setup_hippi_fb (continued)

9

```
    /*
     * Connect this socket to the desired HIPPI Ifield.
     */
    bzero((char *)&client, sizeof(client)) ;
    client.family = AF_HIPPI;
    client.if_unit = 0
    client.protocol = HIPPIPROTO_PH;
    client.ifield.c_ifield = ifield;

    if (connect(sock,(struct sockaddr *) &client, sizeof(client)) < 0) {
          fprintf(stderr, "Can't connect to 0x%x: %s\n", ifield, errormsg);
          close(sock);
          *fd = -1;
          return -1;
    }
    printf("HIPPI ifield set to 0x%x\n", ifield);
```

10

```
     /*
      * Translate the UNIX socket identifier into
      * a CMFS file descriptor number for use by Fortran.
      */
     *fd = CMFS_fdopen(sock, 0);
     return 0;
}
```


Copyright © 1994 Thinking Machines Corporation

Segment 9 initializes the socket address structure:

- It sets the family parameter to `AF_HIPPI`, which is always the case.
- It sets the interface value to 0.
- It sets the protocol to `HIPPIPROTO_PH`, which is the appropriate protocol for the `SOCK_RAW` socket type specified in Segment 3.
- It initializes the `ifield`.

And then connects the socket to the appropriate HIPPI ifield.

Segment 10 translates the UNIX socket descriptor into the CMFS file descriptor that will be used by the Fortran program (`mandel`).

Copyright © 1994 Thinking Machines Corporation

```
c
c    mandelbrot set calculation
c    042288    created      thorp/alex
c
       program mandel     parameter (si=3.0)
       parameter (left_side=-2.5)
       parameter (top_side =-1.2)
       integer res
       parameter (res = 256)
       real zc(1280,1024), zi(1280,1024)      ! real and imaginary components of z
       real zcs(1280,1024), zis(1280,1024)    ! square of real and imaginary z
       real cc(1280,1024), ci(1280,1024)      ! real and imaginary components of c
       real cell_size                         ! size of each pixel
       integer ittr(1280,1024)                ! # of cycles before cell warped out
       integer pixel(1280,1024)               ! output pixel
       integer temp(1280,1024)                ! output pixel
       integer i                              ! loop counter


       call setup_hippi_fb(ifd)
       if (ifd < 0) then
         print *, 'open of framebuffer failed'
         stop
       endif


       call CMF_FILE_FDOPEN(9, ifd, istatus)
       if (istatus < 0 ) then
         print *,'setup of framebuffer failed', istatus
         call CMFS_PERROR("open failure")
         stop
       endif


   5   ittr = 0
       zc   = 0
       zi   = 0
       zcs  = 0
       zis  = 0c   find the size of each cell

       cell_size = si/float(1280)
c    initialize cc
   7   forall(i=1:1280) cc(i,:) = float(i-1) * cell_size + left_side

c    initialize ci
       forall(i=1:1024) ci(:,i) = float(i-1) * cell_size + top_side

       print *,'Initialized arrays'
```

Figure 14. Mandelbrot set program example — calculation.
(1 of 2)

Copyright © 1994 Thinking Machines Corporation

```
c    main loop here
       do 12 i= 1,res - 1
         where (zcs + zis .le. 4.0)
           zi = 2.0 * zc * zi + ci
           zc = zcs - zis + cc
           zcs = zc * zc
           zis = zi * zi
           ittr = i
         end where

         if (MOD(i,2) == 0) then
           where (ittr == MAXVAL(ittr))
             temp = 0
           elsewhere
             temp = ittr
           endwhere

           pixel = ISHFT(MIN((3*255*temp)/i, 255), 16) +
       $               ISHFT(MIN((380*temp)/i, 255), 8) +
       $               ISHFT(MAX(((3*255*temp)/i) - 512, 0), 0

           call CMF_CM_ARRAY_TO_FILE(9, pixel, istatus)
           if (istatus < 0) then
             print *,'Write of array failed'
             call CMFS_PERROR("write failure")
             stop
           endif
           print *,'wrote frame', i
         endif

   12  continue
       goto 7

       call CMF_FILE_CLOSE(9,istatus)
       if (istatus < 0) then
         print *,'CMF_FILE_CLOSE failed'
         call CMFS_PERROR("close failed")
       endif
       stop
       end
```

Figure 14. Mandelbrot set program example — calculation
(2 of 2)

Copyright © 1994 Thinking Machines Corporation

```
#include <sys/types.h>
#include <sys/socket.h>
#include <nethippi/hippi.h>
#include <sys/errno.h>
#include <stdio.h>

#ifndef errormsg
/*****************************************************************************
 *
 *  errormsg
 *
 * A macro which turns errno into a string (reliably).  This macro
 * may be used, e.g.,:
 *
 *   if((fd = open(file, mode)) < 0) {
 *       fprintf(stderr, "%s: can't open %s for reading; %s\n",
 *         progname, file, errormsg);
 *
 * and therefore makes for far more useful error messages than can be
 * produced using perror()
 *
 *****************************************************************************/
extern int errno;
extern int sys_nerr;
extern char *sys_errlist[];

#define errormsg \
  (((errno < 0) || (errno > sys_nerr))? \
  "Errno out of range": sys_errlist[errno])
#endif /* errormsg */

/*
 * Set up a HIPPI connection to the framebuffer,
 * and return (in argument "fd") the CMFS file descriptor to use
 * for Fortran writes to it.
 */
setup_hippi_fb_(fd)
int *fd;
{
    char *getenv();
    int ifield;
    char *p;
    int sock;
    u_int sndbuf , arglen ;
    struct sockaddr_hippi client;

    /*
     * Get a socket for accessing the HIPPI.
     */
    sock = socket(AF_HIPPI, SOCK_RAW, 0);
    if (sock < 0) {
          perror("socket AF_HIPPI SOCK_RAW");
          *fd = -1;
          return -1;
    }
```

Figure 15. Mandelbrot set program example — socket setup code.
(1 of 3)

Copyright © 1994 Thinking Machines Corporation

```
/*
 * If the environment variable HIPPI_FB_IFIELD is set,
 * use it for the IFIELD of the framebuffer.  Otherwise use
 * "2", which happens to be where the framebuffer is on our switch.
 */
if ((p = getenv("HIPPI_FB_IFIELD")) == NULL)
      ifield = 0x2;
else
      sscanf(p, "%i", &ifield);

/*
 * Set the "Start of frame" bit in the vendor unique part
 * of the Ifield.  This is specific to the IOSC framebuffer.
 */
if (! (ifield & 0x20000000))
      ifield |= 0x20000000;

/*
 * set the socket send buffer size
 * this value determines the max write size
 * again, use environment variable if set, otherwise default to
 * 128 MB.
 */
if ((p = getenv("HIPPI_MAX_SNDBUF_SIZE")) == NULL)
      sndbuf = 128 * 1024 *1024 ; /* default value */
else
      sscanf(p, "%i", &sndbuf); /* use env. var value if set */

arglen = sizeof(sndbuf) ;
/*
 * make request to set socket snd_buf size
 */
if (setsockopt(sock, SOL_SOCKET, SO_PAR_SNDBUF, &sndbuf, arglen) <0) {
      perror("setsockopt parallel write");
      close(sock);
      *fd = -1;
      return -1;
}
/*
 * get the actual value allowed
 */
if (getsockopt(sock, SOL_SOCKET, SO_PAR_SNDBUF, &sndbuf, &arglen) < 0) {
      perror("getsockopt parallel write");
      close(sock);
      *fd = -1;
      return -1;
}
/*
 * print what it actually got set to (the os may have limited the max size)
 */
fprintf(stderr, "Max PARALLEL write size : %d (0x%x)\n", sndbuf, sndbuf) ;
```

Figure 15. Mandelbrot set program example — socket setup code.
(2 of 3)

Copyright © 1994 Thinking Machines Corporation

```
    /*
     * Connect this socket to the desired HIPPI Ifield
     */
    bzero((char *)&client, sizeof(client)) ;
    client.family = AF_HIPPI;
    client.protocol = HIPPIPROTO_PH;
    client.ifield.c_ifield = ifield;

    if (connect(sock,(struct sockaddr *) &client, sizeof(client)) < 0) {
          fprintf(stderr, "Can't connect to 0x%x: %s\n", ifield, errormsg);
          close(sock);
          *fd = -1;
          return -1;
    }
    printf("HIPPI ifield set to 0x%x\n", ifield);

    /*
     * Translate the UNIX socket identifier into
     * a CMFS file descriptor number for use by Fortran
     */
    *fd = CMFS_fdopen(sock, 0);
    return 0;
}
```

Figure 15. Mandelbrot set program example — socket setup code.
(3 of 3)

Copyright © 1994 Thinking Machines Corporation