3 *iWarp Runtime System*

The iWarp runtime system (iW/RTS) is a resident runtime system kernel that lets you manage iWarp resources and provides services that let you control your iWarp programs at execution time. iW/RTS provides the following utilities for loading and executing programs:

lgo	generates a shell script to load and execute cell programs using the iWarp Monitor (iwmon) command set
iwmon	provides a set of UNIX commands to interface with the iWarp monitor daemon

Loading and Executing Programs: lgo

Before you can execute your program on the iWarp system, you must create an executable load module using the iwcc command. The iwcc command is described in Chapter 2.

Once you have an executable load module, you can use lgo to generate a script that loads and executes the program on the iWarp system. When you run the script for your program it does the following:

1. locks the array
2. starts the HPI daemon
3. loads and runs your program
4. waits for exit status
5. unlocks the array

The lgo command has the following syntax:

```
lgo program [options]
```

```
lgo [options] -r program cell_list [%][args]
[-r  program cell_list [%][args]] ...
```

program is name of the executable module you want to load on an iWarp array.

option can be any of the following:

-S *n*	Selects the array defined by SIB number *n*. This option overrides the environment variable IWARP_SIBNUM. The default SIB number is 0.
-H *hostname*	Selects array host *hostname*. This option overrides the environment variable IWARP_HOSTNAME. The default is the local host on which you logged in.
-C *configpath*	Selects an alternate root pathname for the iWarp configuration files. The default is /iwarp/etc3.0/cfg. If the environment variable IWARP_CFGPATH is defined, its value overrides the default, but doesn't override the command line option.

-ne	Skips the run phase; for use with -o.
-nl	Prevents locking and unlocking of the array. lgo assumes that the array is already locked and leaves the array locked after execution completes. When you use this option, lgo does not start the host proxy interface (HPI) daemon process.
-w *n*	Waits until *n* programs have reported exit status. By default, lgo waits for all programs to report exit status.
-s	Uses stdfiles.
-o *script*	Produces a script of monitor commands used to run the program and writes it to *script.*
-R rootpath	Selects an alternate root pathname for the iWarp software distribution. The default is /iwarp. If the environment variable IWARP_ROOTPATH is defined, its value overrides the default, but doesn’t override this command line option.
-r *program cellid* [%*args*] . . .	Loads and runs *program* with arguments specified by argument array *args* on a cell or range of cells specified by *cellid*. The lgo command line can have multiple -r switches.
–t timelimit	Set a timelimit on your lock. By default, after 10 minutes your lock on the array will expire and the next user who tries to get the array will break your lock. If timelimit is specified, the lock will expire after timelimit minutes. If timelimit is zero, the lock will never expire.
–m message	During an iwstat, a message will be displayed. The message will be displayed while you have the array locked. Note that message must appear as a single word on the command line; because of this you will usually want to put quotes around message and avoid using special shell characters such as !, ’, etc.
-V	Prints command version information.
-VV	Prints command version information, and exits without executing the command.

Environment Variables

You can set environment variables so that you do not have to specify an array number or an array host. You can set the array number with the IWARP_SIBNUM environment variable, and the array host with the IWARP_HOSTNAME environment variable. The following example commands illustrate the use of the setenv command to set these environment variables. The commands set the array number to 0 and the array host to stargazer.

```
% setenv IWARP_SIBNUM 0

% setenv IWARP_HOSTNAME stargazer
```

If these environment variables are set, lgo will execute on the defined array host and array number unless another array or host are specifically requested with the -S or -H options.

lgo Examples

This section provides several examples showing how to use the lgo command. The examples assume that the IWARP_SIBNUM variable is set to 0, and the IWARP_HOSTNAME variable is set to *stargazer.*

In the following example, lgo loads and runs a.out on cell 0 of the iWarp array attached to sib 0 on the host named stargazer.

```
lgo a.out
```

In the following example, lgo loads and runs a.out on cells 0 to 64 of a 64-cell array attached to SIB 1 on stargazer. Also, 3, arg0, arg1, and arg2 are passed to a.out as arguments. The percent sign (%) is a delimiter that optionally separates a cell ID argument from other arguments.

```
lgo -S1 -r a.out 0..64 % 3 arg0 arg1 arg2
```

In the following example, lgo generates a script that can load and execute a.out on cell 0 and 10, and wait for 2 exit status reports from the array. The -ne switch prevents the execution of the script.

```
lgo -ne -o script -r a.out 0 10 -w 2
```

The script contains monitor commands and can be executed using the host shell.

In the following example, lgo runs a.out on cells 0 to 10, b.out on cells 21 to 30, and c.out on cell 40 on array 1 on a host named hubcap. Notice that the command line options -S and -H override the environment variables IWARP_SIBNUM and IWARP_HOSTNAME respectively.

```
lgo -S1 -Hhubcap -r a.out 0..10 -r b.out 21..30 -r c.out 40
```

iWarp Monitor: iwmon

The iWarp Monitor is a set of commands that use the Sun RPC/XDR interface to access the iWarp monitor daemon. These programs allow the monitor daemon to service users of Unix shells and facilitate development of array-program execution utilities. Each of these commands provide one or more control functions for an iWarp array. When you invoke lgo to execute a program, lgo creates a script of monitor commands to control execution.

NOTE

All monitor daemon logging goes to /var/adm/messages. For a complete description of the iWarp monitor daemon logging facility, refer to the iWarp `iwmond` man page.

The monitor commands can also be invoked from the shell command line. The following options are standard across all monitor commands.

-S *n*	Selects the array defined by SIB number *n*. This option overrides the environment variable IWARP_SIBNUM. The default SIB number is 0.
-H *hostname*	Selects array host *hostname*. This option overrides the environment variable IWARP_HOSTNAME. The default is the local host on which you logged in.
-R *rootpath*	Selects an alternate root pathname for the iWarp software distribution. This option overrides the environment variable IWARP_ROOTPATH. The default is /iwarp.

- C *configpath* — Selects an alternate root pathname for the iWarp configuration files. This option overrides the environment variable IWARP_CONFIGPATH. The default is /iwarp/etc3.0/cfg.

Monitor Commands

The following is a brief description of each iWarp monitor command. Each description includes examples of the command. The examples assume that the environment variables IWARP_SIBNUM and IWARP_HOSTNAME are set to 0 and stargazer.

iwlock

The iwlock command locks the array and gives exclusive access to the user. The array should be locked before using any other commands described in this section. If the array is available, you will get the following message when you use the iwlock command:

```
% iwlock
    SIB 0 on stargazer has been locked.
```

If someone else has already locked the array, you will get the following message when you use the iwlock command:

```
% iwlock
  Array is locked by another user.
```

iwbrklock

The iwbrklock command breaks another user's lock. The following is an example of the iwbrklock command:

```
% iwbrklock
    The lock of SIB 0 on stargazer has been broken.
```

iwunlock

The iwunlock command removes an array lock you previously set using iwlock. The iwunlock command gives the following message when you unlock an array:

```
% iwunlock
    SIB 0 on stargazer unlocked.
```

If you use the unlock command when you don't have the array locked, you will get the following message:

```
% iwunlock
    You have not locked the array.
```

iwreset

The iwreset command performs a hard reboot on an iWarp array. It clears the previous cell exit status, kills the HPI daemon, and kills the debugger daemon. This command does not return a message. The following is an example of the iwreset command:

```
% iwreset
    SIB 0 on stargazer has been reset.
```

iwload

The iwload command loads iWarp executable files on cells. You can only load one program with the iwload command, but you can load it on one or more cells. The following example loads the executable file a.out on cells 45 to 56 and cells 58 to 61, and passes arg0, arg1, and arg2 to a.out on these cells.

```
% iwload -r 45..56 -r 58..61 a.out arg0 arg1 arg2
```

iwrun

The iwrun command starts the execution of cell programs that were loaded using iwload. The execution starts at the address specified by the entry field of the optional header for the executable file loaded. The following example starts running cell programs loaded in the iwload command example.

```
% iwrun a.out -r 45..56 -r 58..61
```

iwstat

The iwstat command displays status information about the use of an iWarp array and about the cell program exit status of the cells. If no exit status exists for a cell program, this is indicated by a dash in the display for that cell. The following is an example of the iwstat command:

```
% iwstat -se

     Version: 1

Locker: iw_user (4217) since Tue Jul 7 10:38:32 PST 1992
     No time limit exists on this lock.
     No locking message.
Exit status:

4/   0: --/-- --

     2:   -- --
```

iwclrstat

The iwclrstat command clears the monitor's cell exit status buffer by marking all entries as invalid. This command does not return a message. The following is an example of the iwclrstat command:

```
% iwclrstat

%
```

iwtime

The iwtime command reports the execution time of a cell program in seconds. You must use this command before any reset or unlock. The following example reports the execution time of the cell program that executed on cell 0:

```
% iwtime 0

     2.312345 seconds.
```

iwvers

The iwvers command prints a brief messsage to standard output indicating the version of the iWarp monitor daemon, iwmond, that is running for the selected array.

```
% iwvers -S3 -Hiwarpsd
     The server on stargazer for SIB 0 is at version 1.
```

iwkill

The iwkill command terminates the execution of a cell program. Control on the cell goes back to the init process of the runtime system. The following example terminates the execution of the cell program running on cell 0:

```
% iwkill 0

%
```

iwhpi

The iwhpi command starts the HPI daemon. Once started, the HPI daemon runs until the array is reset or unlocked. I/O is done through the HPI daemon, so the daemon must be running to use I/O. The following is an example of the iwhpi command:

```
% iwhpi

%
```

iwdepcsr

The iwdepcsr command deposits data to the control space register on a cell. The following example deposits value 0xff in control space register rpi0 on cell 34. You can specify the data to be a hexadecimal or decimal value:

```
% iwdepcsr 34 rpi0 0xff

%
```

The control space registers are defined in the include file regnums.h. For a complete listing of the control space register definitions, refer to the *iWarp C User's Guide*.

iwdepmem

The iwdepmem command deposits data into memory locations on a cell. You can specify data words as hexadecimal or decimal values. A memory address is a word-aligned address that you can use to display as hexadecimal or decimal values. The data words are separated by blank spaces. The following example deposits data into the three memory locations starting at memory location 0x20000 on cell 25:

```
% iwdepmem 25 0x20000 0xff 0xfc 23

%
```

iwexcsr

The iwexcsr command examines a control space register on a cell. The returned value of the control space register is expressed in hexadecimal. The following example returns the value of the rpi0 control space register on cell 34:

```
% iwexcsr 0 rpi0
    rpi0: 0xaff0c
```

iwexmem

The iwexmem command examines a specified number of bytes of memory starting at a specified address. The starting address is expressed in hexadecimal and the byte count is expressed in decimal. The following example examines 12 bytes of memory on cell 25 starting at 0x20000:

```
% iwexmem 3 0x12340 32
    0x12340: 0x1d004962 0x3ae02ed8 0x20a02c49  0xd43c25a
    0x12440:  0xf0000da 0x20a02d48 0x24a0005d 0x1c0053c0
```

System Calls Supported by the iWarp Runtime System

iW/RTS support system calls for UNIX System V standard I/O compatibility. Table 3-1 lists the system calls supported by iW/RTS.

Table 3-1: System Calls Supported by iW/RTS

System Call	Purpose
access	determine accessibility of file
chdir	change working directory
chmod	change mode of file
close	close file descriptor
create	create new file
dup	duplicate open file descriptor
dup2	duplicate open file descriptor
_exit	terminate cell program
fchdir	change working directory
fchmod	change mode of file
fstat	get file status
ftruncate	set file length
getcwd	get current working directory
ioctl	control device
link	link to a file
lseek	move read/write file pointer

Table 3-1: System Calls Supported by iW/RTS (cont.)

System Call	Purpose
lstat	get file status
mkdir	make a directory
open	open file for reading or writing
read	read from a file
rename	change file name
rmdir	remove a directory
stat	get file status
truncate	set file length
unlink	remove a directory entry
write	write to a file

Memory Allocation

Figure 3-1 shows how memory mapping takes place on a typical iWarp cell. The map on the left shows the mapping when a user program is loaded onto a cell. Initially, the stack exists from the end of the user program to the top of the physical memory. A memory allocation request from the user program causes the heap to grow and the stack to shrink. The map on the right shows mapping after the user program has dynamically allocated some memory using malloc(). A memory allocation request will fail if a stack of less than 8 Kbytes results.

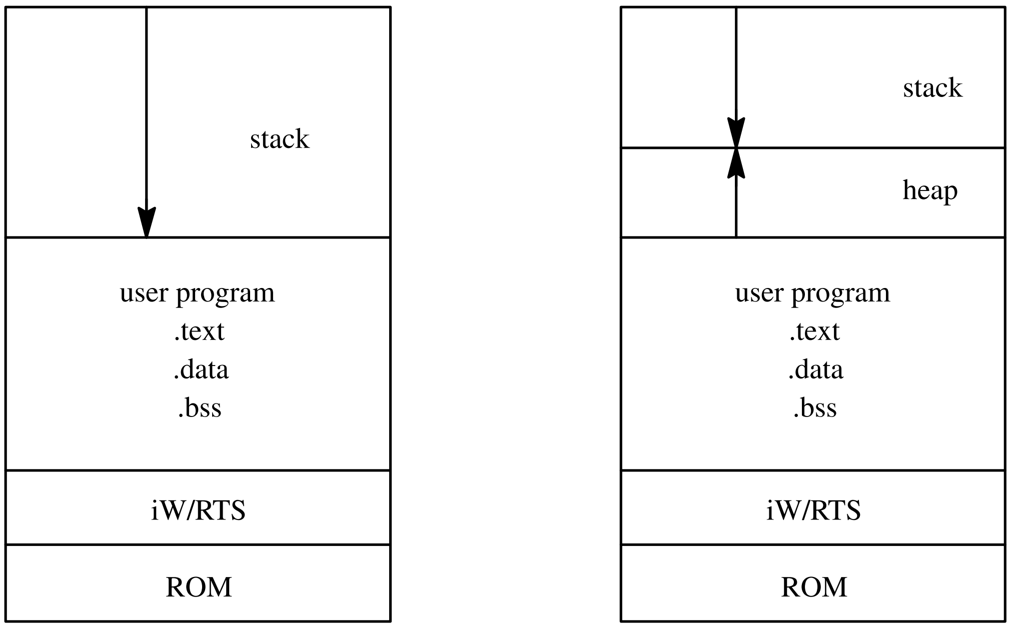

Figure 3-1: Memory Mapping on an iWarp Cell (Static RAM Only)

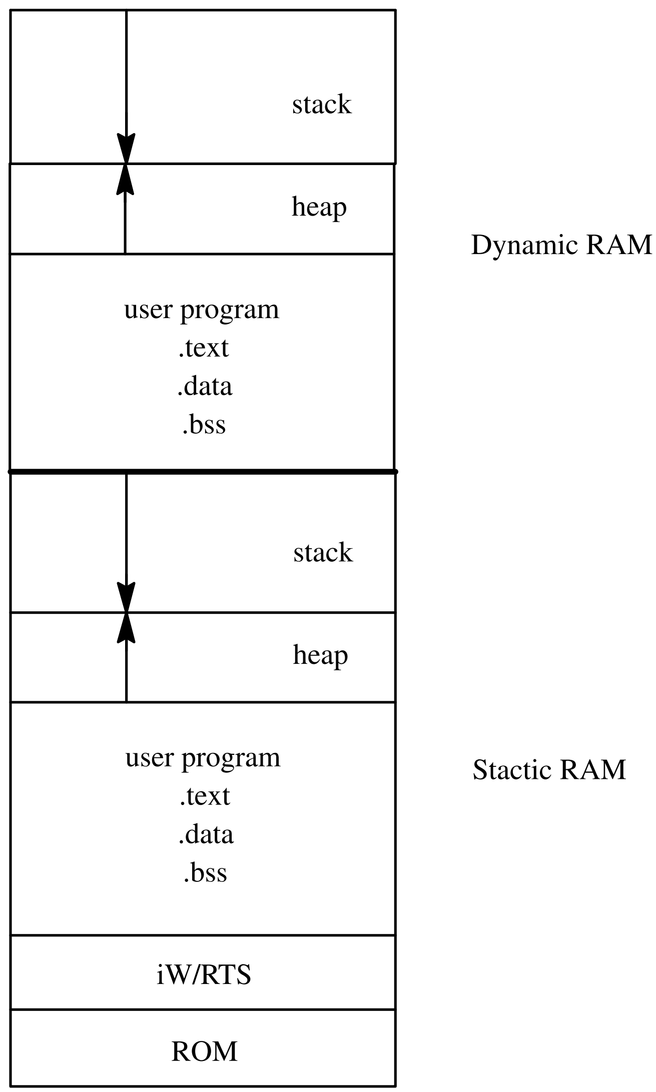

Figure 3-2: Memory Mapping on an iWarp Cell (Dynamic and Static RAM)

Figure 3-2 shows that when an iWarp cell uses both dynamic and static RAM that the dynamic RAM and the static RAM each have their own stack, heap, and user program memory areas.