3 The iWarp Runtime System

The iWarp system provides the program developer with a resident runtime system kernel to manage resources and provide services that give the user more control at program execution time. The iWarp Runtime System (RTS) provides the following utilities for loading, executing, and debugging programs:

rtslink	links RTS startup code to a compiled user program
lgo	loads and executes a program
iwmon	provides an interactive environment for loading, executing, and debugging

Each of these utilities is described in a later section.

RTS also supports limited versions of the following system calls on the SIB and array cells:

ioctl()	supports isatty() only
close()	supports cleanup code in libc
exit()	supports the iwmon wait command, the -w option to lgo, and notifies the host monitor when a cell exits
read()	reads from fd 0 to a buffer
write()	writes a buffer to fd 1 or fd 2

There is no support for open(). Applications have stdin, stdout, and stderr open by default.

RTS also supports malloc(), free(), and realloc(), but supplies them as system calls rather than Section 3 library functions.

Memory Allocation

The memory map for an iWarp cell is as follows:

0x0..0x1fff	on-chip ROM
0x2000..0x120ff	RTS
0x12100..end	user program, followed by heap and stack

The memory map for the SIB is as follows:

0x0..0x1fff	on-chip ROM
0x2000..0x7fff	diagnostic array kernel EPROM
0x8000..0x200ff	RTS
0x20100..end	user program, followed by heap and stack

The Stack and the Heap

The system stack is initialized by RTS to the top of memory, 0x7fffc. The heap is initialized to a round value following the user program's bss section. The stack low limit (SPLO) is always the top of the heap. As malloc() needs space, it calls a combination of internal versions of sbrk() and brk(). Brk() adjusts SPLO up, but within reason, that is, never let SPHI – SPLO be less than 8K bytes, don't move SPLO above SP, and so on. If memory is exhausted, a stack low-limit event occurs, indicating that the stack and heap collided.

Once the top of the heap is incremented, that memory remains the property of the heap. SPLO is not decremented when free() is used to deallocate memory. This means that the allocation of a large portion of the heap, however briefly it may be needed, limits the size of the stack for the duration of the application.

Preparing an Executable File: rtslink

To create an executable load module for the iWarp system, you must combine RTS startup code with your compiled program. rtslink prepares a loadable COFF file from a compiled user program by adding the RTS startup code. The rtslink command has the following syntax:

`rtslink [`*`option`*` ...] `*`filename`*` [`*`arglist`*`]`

filename is the name of the compiled user program.

arglist is any command line arguments to be passed to the user program.

option can be any of the following:

-o *filename*	Defines the name of the loadable COFF file. The default name is rts.*pid,* where *pid* is the process id.
-sib	Specifies that the program is to run on the interface cell. The default is to run on an array cell.
-V	Prints version information and continues.
-VV	Prints version information and exits.

In the following example, rtslink creates the loadable file a.run from the compiled user file a.out:

```
rtslink –o a.run  a.out
```

Loading and Executing Programs: lgo

lgo loads and runs a program on the iWarp system by doing the following:

- locking the specified array
- creating a script that specifies a sequence of load and run commands
- invoking the iWarp monitor and passing the script to the monitor
- unlocking the array when execution has completed

NOTE

> The modules being loaded and executed with lgo must first be linked to the RTS startup code by rtslink.

The lgo command has the following syntax:

```
lgo [option ...] device
```

device specifies an iWarp interface cell to which the array is attached.

option can be any of the following:

-f *filename* — Executes iwmon commands from the file specified by *filename* . This option cannot be used with the -r or -w options.

-n — This option is used with -o to generate an iwmon script file without executing the program.

-o *filename* — Outputs the iwmon script to the file specified by *filename* .

-R *file id* — Generates commands in the iwmon script to load and run *file* on the interface cell specified by *id*.

-r *file id_list* — Generates commands in the iwmon script to load and run *file* on the cells specified by *id_list* , where *id_list* has the following syntax:

```
id [ .. id ]
```

where *id* is a cell id number.

-w *n* — Waits for *n* cells to exit before returning control to iwmon.

-s — Forces stdin, stdout, and stderr to be redirected from or to the files stdin*c*, stdout*c*, and stderr*c*, where *c* is the cell number of the cell making the I/O request.

-V — Prints lgo version information and continues.

-VV — Prints lgo version information and exits.

In the following example, lgo loads and runs hello.out on cell 0 of the array attached to /dev/isib0:

```
lgo −r hello.out 0 −w 1 /dev/isib0
```

In the following example, lgo loads and runs hello.out on all cells of a 64-cell array attached to /dev/isib1:

```
lgo −r hello.out 0..63 −w 64 /dev/isib1
```

In the following example, lgo generates the script lgo.script for running hello.out on cells 0 and 10:

```
lgo −n −r hello.out 0 10 −o lgo.script −w 2 /dev/isib0
```

In this example, lgo runs the script generated in the previous example on the array attached to /dev/isib1:

```
lgo −f lgo.script /dev/isib1
```

The iWarp Monitor: iwmon

The iWarp monitor provides an interactive environment to load, execute, and debug programs. The monitor allows you to deposit and examine values in cell local memory, general-purpose registers, and control space registers. It also provides for the use of breakpoints.

Invoking the Monitor

To invoke the iWarp monitor, use the iwmon command as follows:

```
iwmon [option ...] [sibnum]
```

sibnum is the number of the Sun Interface Board (SIB) to which the iWarp array is attached.

The SIB number is set when the iWarp system is installed. The SIB number will default to zero if *sibnum* is not specified on the command line. If no SIB exists at zero, iwmon will issue an error message, and you will have to specify an SIB number.

More than one iWarp system can be attached to the same SUN workstation. If this is the case each SIB will have a unique SIB number and the monitor can be used to interact with any of the iWarp systems.

option can be any of the following:

-b *bootfile*	Boots the array-resident portion of iwmon from *bootfile*, instead of from the default package.
-l *logfile*	Specifies *logfile* as the file to receive logging output generated by the programs on the array.
-s	Instructs the monitor to use stdfiles for standard output and standard error. Output on the stdout and stderr streams from cell *n* is directed to files stdout*n* and stderr*n* in the current working directory. Input to the cells stdin streams is taken from stdin*n*.

The following interactive commands are available when you invoke the monitor:

Monitor commands

?	Lists all available commands offered by *iwmon.*
boot [*filename*]	Boot, download, and run the array-resident monitor. If *filename* is not included, the default bootfile is used.
ex *cellid addr* [*count*]	Examine cell memory. Displays *count* number of words of memory on cell *cellid* from memory location at word address *addr*. If *count* is not supplied, the monitor examines one word.
exr *cellid rfaddr*	Examine register file. Examines the word at *rfaddr* in the register file on cell *cellid.*
exc *cellid csaddr*	Examine control space register. Examines the word at *csaddr* in the control space on cell *cellid*.

dep *cellid addr data* [*count*]	Deposit into cell memory. Deposits *data* on cell *cellid* in *count* memory locations starting at address *addr* . If count is not supplied, *data* is deposited in one word.
depr *cellid rfaddr data*	Deposit into register file. Deposits *data* into the word at *rfaddr* in the register file on cell *cellid.*
depc *cellid csaddr data*	Deposit into control space register. Deposits *data* into the word at *csaddr* in the control space on cell *cellid.*
load *cellid filename*	Load a program into cell memory. Loads the COFF file *filename* on cell *cellid* at the location determined by the COFF information in the file. The file must be linked using rtslink.
run *cellid addr*	Run a program. Starts execution of a program on cell *cellid.* The execution starts at address *addr* .
sbp *cellid addr*	Set a breakpoint. Sets a breakpoint at address *addr* on cell *cellid.* A status line is printed that indicates the breakpoint number, which can later be used to unset the breakpoint. Up to 12 breakpoints can be set on each cell. Breakpoints will not work if set at a Compute and Access instruction or any flow of control instruction (including call, return, and end loop).
ubp *cellid bp*	Unset a breakpoint. Unsets breakpoint *bp* on cell *cellid.*
status *cellid*	List breakpoints. Lists currently set breakpoints and their corresponding addresses.
cbp *cellid bp*	Continue execution. Continues execution from breakpoint *bp* on cell *cellid* .
wait *N*	Suspend processing. Suspends command processing until *N* exit messages are received from the array.
log [[-a] *filename*]	Redirect messages. Redirects monitor-based log messages to a file. The -a option appends messages to the file, otherwise, a new file is created. By default, log messages are sent to stdout.
exit	Exit the iWarp monitor.
quit	Same as exit.

Input Conventions

The monitor commands are subject to the following conventions:

- The monitor accepts numeric arguments in decimal, hexadecimal, or octal bases.
- Local memory locations must be specified as word-aligned, byte addresses. Each cell is assumed to have 512K bytes of memory available. The monitor accepts values in the range of 0 to 0x7FFFC.

NOTE

The monitor can handle memory configurations of up to 1.5 Mbytes.

- Register file addresses must be word-aligned, byte addresses. The monitor accepts values in the range of 0 to 0x1FC. The monitor also recognizes register file names defined by the iWarp assembler.
- Control space addresses must be byte addresses. The monitor accepts values in the range 0 to 0x1ff. The monitor also recognizes control space register names defined by the iWarp assembler.
- Cell id numbers range from 0 to the number of cells in the array, including the SIB. The SIB cell id is always equal to the number of cells in the array
- The maximum count for examine and deposit memory commands is 128.
- File names are SunOS pathnames.

Monitor Example

The following is an example of running a single-cell program on the iWarp system using the iWarp monitor. Explanations of the commands used in the example follow the example.

```
sunhost> siblock 1

sunhost> iwmon 1
```

iWarp Monitor V2.1 Mon July 1 14:20:24 PST 1991 coff V2.1

```
resetting the array...

booting /iwarp/lib/iwmon.package...

iwmon>load 2 myprog

iwmon>run 2 4000

iwmon>exit

sunhost> sibunlock 1
```

The commands shown in the example perform the following functions:

siblock	locks the iWarp SIB interface for exclusive use.
iwmon	starts the monitor program and initializes the iWarp array.
load 2 myprog	loads the user program “myprog” on cell-id “2” of the iWarp array .
run 2 4000	runs the program loaded on cell-id “2” starting at word-address “4000” hex.
quit	exits the iWarp monitor.

sibunlock unlocks the iWarp SIB interface and frees it for use by others.