3 Running Your Program Under iwdb

The `run` Command

To run your program under iwdb, use the `run` command. The program must already have been specified as an argument to the `iwdb` command or with a file specification command. The `run` command has the following syntax:

```
run [arguments]
```

arguments are any arguments passed to the program being run. The arguments are passed to a shell, which expands wildcard characters and performs redirection of I/O, and then to the program. If no arguments are included, the run command uses the same arguments used by the previous `run` command.

You can use the `set args` command to specify the arguments to be used the next time the program is run. `set args` with no arguments means to use no arguments the next time the program is run. If you have run your program with arguments and want to run it again with no arguments, you must use the `set args` command.

Working Directory

Each time you start your program with the `run` command, it inherits the working directory from the current working directory of the debugger. The debugger's working directory is initially inherited from its parent process, typically the shell, but you can specify a new working directory with the `cd` command as follows:

```
cd  directory
```

This command sets the debugger's working directory to *directory.*

The debugger's working directory also serves as a default for the commands that specify files for the debugger to operate on. You can use the `pwd` command to list the debugger's working directory.

Setting the Program Environment

The program environment consists of a set of environment variables and their values. Environment variables typically record such things as your user name, home directory, terminal type, and search path for programs to run. Environment variables are usually set with the shell and they are inherited by all the other programs you run. When debugging, it can be useful to try running the program with different environments without having to start the debugger over again.

The iWarp debugger provides the following commands to set the program environment:

`info environment [`*varname*`]`

Prints the value of environment variable *varname* to be given to your program when it is started. If *varname* is not specified, the debugger prints the names and values of all environment variables to be given to your program when it is started.

`set environment` *varname* `[=` *value*`]`

Sets environment variable *varname* to *value*, for the current program only. *value* may be any string, and it is optional. If *value* is not specified, the variable is set to a null value.

For example, the command:

```
set env USER = bob
```

tells the program, when subsequently run, to assume it is being run on behalf of the user named `bob`.

`delete environment` *varname*

Remove variable *varname* from the environment to be passed to your program. This is different from `set env` *varname* because `delete environment` leaves the variable with no value, which is distinguishable from an empty value.

`unset environment` *varname*

Same as `delete environment` *varname*

The debugger must know the path to the RPC server and other execution software. The default execution path for the debugger is /iwarp/bin, the iWarp installation directory. To override this default path, assign a new path to the environment variable IWDB_PATH. The following example shows how to set a new path:

```
set environment IWDB_PATH /usr/local/tools
```

This example would cause the debugger to invoke /usr/local/tools/iwdb_serv on the array server.

Post-Mortem Debugging

Post-mortem debugging is the ability to debug a cell program at the time an error occurs. The iWarp debugger provides limited support for this ability with the `connect` command. In post-mortem debugging, you can only examine and modify memory locations and register contents. You cannot use breakpoints or handle signals.

When starting the user interface, you must use the following command line option:

```
-pm
```

This option attaches to an array without resetting it or loading the resident kernel. If this option is used and the array is not currently executing a user program, the following error message occurs:

```
200: <sib number>  is not running an array program.
```

When the array server portion of the debugger attempted to send a connection request to the SIB, it found that the SIB was in a reset state, and no user program was running. This is a fatal error.

After the user interface starts up, the `connect` command is used to establish a cell as the current context as follows:

`connect` *cell_id*

This command makes *cell_id* the current context. This is similar to the `load` command, except the program is not actually loaded onto the cell.