2 Program Development Environment

Introduction

The iWarp programming tools reside on the iWarp host, which is a Sun Microsystems Unix-based workstation. The program development environment includes the following tools for developing programs:

- C compiler
- FORTRAN compiler
- assembler
- linker
- disassembler
- profiler
- debugger

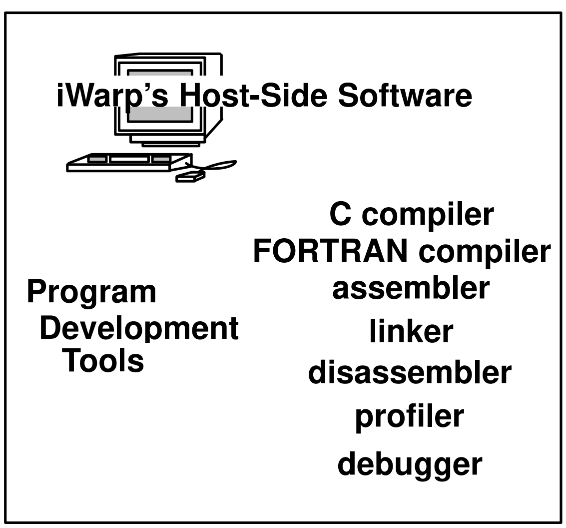
The program development environment also includes the following tools for file manipulation:

- archiver
- lister
- dumper
- sizer

The following sections describe these tools and the commands and options used to invoke them.

To view the manual pages for the iWarp program development tools, use the iwman command. For information on the iwman command, see Appendix A “Commands Quick Reference.”

Program Development Tools



iWarp C

iWarp C is a superset of the C language as defined in the 1978 edition of *The C Programming Language*, by Brian W. Kernighan and Dennis M. Ritchie. iWarp C contains several iWarp-specific extensions, as well as a number of widely recognized extensions. The following list summarizes the iWarp-specific extensions:

- communication support (send and receive primitives).
- condition code checking support (iWarp condition codes).
- assembly code inlining facility.
- pragma support for inlining specification.

Refer to the *iWarp C Manual* for a complete description of the iWarp-specific implementations of the C language and the iWarp C extensions. For a complete description of the C programming language, refer to *The C Programming Language*, Brian W. Kernighan and Dennis M. Ritchie, 1978, Prentice-Hall.

The iWarp C compiler is an optimizing compiler for the iWarp processor. The compiler is a cross-compiler that runs on the Sun host and produces code for the iWarp system. The iWarp C compiler translates C programs into iWarp executable load modules or iWarp relocatable binary programs for loading with the iWarp linker. The iWarp C compiler produces object code files in Common Object File Format (COFF).

Invoking the C Compiler: iwcc

To invoke the iWarp C compiler, use the iwcc command as follows:

```
iwcc [option ...]  filename ...
```

filename can be any of the following:

- A C source file (.c file extension).
- A C source file that is not to be run through the preprocessor (.i file extension).
- An assembly language source file (.s file extension). These files are passed to the assembler and an object file is produced.
- An intra-file inlined intermediate file (.K file extension). These files are used by the global optimizer for inter-file inlining of any specified C programs.

All other files are passed to the linker.

option can be any of the following options.

iwcc Options

-#	List each phase of compilation being executed and the command options used to invoke the phase.
-##	Same as -#, but also prints the full path used to load each phase of the compilation.
-###	Same as -##, but do not execute any phases of the compilation.
-c	Suppress the link edit phase of the compilation and produce an object file for each source file.

-C	Prevent the C preprocessor from removing comments.
-D*name* [=def]	Define *name* for the preprocessor as if by a #define directive. If =def is not included, *name* is defined as 1. If both the -D and -U options are used with the same *name*, *name* will be undefined.
-E	Run only the preprocessor on the named C programs and send the result to standard output. The output includes cpp-type line number information.
-fasmwiden	Convert floating-point operands to double-precision operands for asm insert function parameters and function return types.
-fnoasmwiden	Suppress the conversion of floating-point operands to double-precision operands for asm insert function parameters and function return types. This is the default.
-fnowiden	Suppress the automatic conversion of floating-point operands to double-precision operands within expressions, and suppress the conversion of floating-point parameters to functions and function return types. This option is not recommended, and there is no library support for float parameters or function returns.
-fsingle	Suppress the automatic conversion of floating-point operands to double-precision operands within expressions.
-g	Generate information for symbolic debugging. This suppresses all optimization.
-G*c*	Retain intermediate files from pass *c* for debugging, where *c* is one of the following: p preprocessor 0 compiler first pass q Portable Code Optimizer (PCO) pass 1 compiler second pass a assembler l link editor m listing merger
-I*pathname*	Add *pathname* to the list of directories in which to search for #include files with relative pathnames (not beginning with /). The preprocessor first searches for #include files in the directory containing the source file, then in directories named with the -I option, then in /iwarp/include.
-K	Invoke PCO to generate an intra-file inlined intermediate file. Arguments whose names end with .c are compiled, intra-file inlined by PCO, and each intermediate file is left in a file whose name is that of the source file with .K substituted for .c. The output file name cannot be changed with the -o option.
-l*x*	This option is an abbreviation for the library name lib*x*.a, where *x* is a string. This option is used with the -L option.
-L*dir*	Add *dir* to the list of directories containing object library routines. Directories specified with -L are searched before the standard directory, /iwarp/lib. This option only affects -l*x* options which follow on the command line.

-M	Merge the C and generated iWarp assembler files into a file suffixed .m. For a single file, the output file name can be changed with the -o option.
-MM	Same as -M, but includes the text of #include files in the listing file.
-N*letter*	Suppress aspects of the link process. *letter* can be C, L, or S. C — suppress the automatic use of a default link control file L — suppress automatic linking with libc.a (and libg.a if -g is used) S — suppress the automatic linking with iwcrt0.o (or iwmcrt0.o if -p is used)
-o *filename*	Specifies *filename* as the name of the object file, overriding the default name a.out. Use with -c modifies the name of the object file. Use with -P modifies the name of the preprocessed file. Use with -S modifies the name of the output assembly file. Use with -M modifies the name of the output merged listing file.
-O[*level*]	Invoke optimization as indicated by *level*, where *level* is one of the following: 0 — no optimizations 1 — global optimizations on intermediate code (PCO) 2 — basic block compaction generation of Compute and Access instructions, as well as PCO optimizations 3 — software pipelining, basic block compaction, and PCO optimizations 4 — Same as -O3, but uses the full capabilities of C&A instructions and optimizes accesses to external variables. You should not use this optimization level if your program uses spooling. For further information, refer to the `spooling` manual page. 5 — Same as -O4, but allows generation of code using short-term spools to improve the performance of loops that access memory heavily. You should not use this optimization level if your program uses spooling. For further information, refer to the `spooling` manual page. If *level* is not specified, this option will invoke all available optimizations.
-p	Produce code that counts the number of times each routine is called. If link editing takes place, replace the standard startoff routine by one that automatically calls the monitor at the start and arranges to write a mon.out# file at normal termination of execution of the object program (# is the cell number). This allows generation of an execution profile with the iwprof command.
-P	Run only the preprocessor on the specified C programs and leave results in the corresponding files suffixed .i. For a single file, the output file name can be changed with the -o option.
-Q*options*	Invoke optimizations as defined by *options*, where *options* can be any or all of the following:

G	invoke global optimizations (PCO)
C	compaction of basic block for Compute and Access instruction
S	software pipelining

If compile time is an issue, but you want to do some optimization, use the -O*level* option to reduce the amount of optimization done.

-S
: Compile the specified C programs and leave the assembler-language output in corresponding files suffixed .s. For a single file, the output file name can be changed with the -o option.

-T
: Causes the compiler to retain all intermediate files in the same directory as the source code.

-u*name*
: Causes the linker to enter *name* as an undefined symbol in the symbol table.

-U*name*
: Remove any initial definition of *name*, where *name* is a reserved symbol predefined by the preprocessor.

-V
: Display the version of the C compiler and all invoked passes, and perform compilation as specified.

-VV
: Display the version of the C compiler and all possible passes, but do not perform compilation.

-Wc, *arg1* [, *arg2*...]
: Hand off any specified arguments to pass *c*, where *c* is one of the following:

p	preprocessor
0	compiler first pass
q	PCO pass
1	compiler second pass
a	assembler
l	link editor
m	listing merger

-x
: Preserve only global (non-.globl) symbols in the output symbol table. Enter only external symbols.

-Yc, *dir*
: Get *c* from directory *dir*, where *c* is one of the following:

p	preprocessor
0	compiler first pass
q	PCO pass
1	compiler second pass
a	assembler
l	link editor
m	listing merger
C	link.ctl
L	default search directory for libraries (/iwarp/lib)
S	startup code (iwcrt0 or iwmcrt0)

iWarp Fortran

The iWarp FORTRAN compiler is a port of the AT&T Portable FORTRAN 77 Compiler (PCC2 technology), with a few iWarp-specific extensions and support for VMS extensions added. The iWarp FORTRAN compiler compiles standard ANSI 77 FORTRAN source code.

Refer to the *iWarp FORTRAN Manual* for a complete description of the iWarp implementation of the FORTRAN language. For a description of ANSI FORTRAN, refer to the *American National Standard Programming Language FORTRAN,* ANSI C3.9-1978, American National Standards Institute, New York, 1978.

Invoking the Fortran Compiler: iwf77

To invoke the iWarp FORTRAN compiler, use the iwf77 command as follows:

```
iwf77 [option ...]  filename ...
```

filename can be any of the following:

- A FORTRAN source file (.f file extension).
- A FORTRAN source file with C comments and C preprocessor directives (.F file extension).
- An assembly language source file (.s file extension). These files are passed to the assembler and an object file is produced.
- An intra-file inlined intermediate file (.K file extension). These files are used by the global optimizer for inter-file inlining of any specified source programs. Intra-file and inter-file inlining are described in Chapter 4.
- RATFOR files (.r file extension) or EFL files (.e file extension). These files are transformed by the appropriate preprocessor, then compiled by the FORTRAN compiler. Intel does not distribute RATFOR or EFL preprocessors, but the iWarp FORTRAN driver does support their use.

All other files are passed to the linker.

option can be any of the options listed in the following section.

iwf77 Options

-#	List each phase of compilation being executed and the command options used to invoke that phase.
-##	Same as -#.
-###	List each phase of compilation and the command options used to invoke that phase, but do not execute.
-1	Compile DO loops so they are performed at least once if reached This is the same as -onetrip.
-66	Disable FORTRAN 77 and VMS F77 language features to enhance FORTRAN 66 compatibility.
-77	Disable VMS extensions.
-auto	Make local variables automatic (non-persistent) by default. This option does not affect variables explicitly declared as persistent using SAVE, variables initialized with DATA statements, or variables equivalenced to persistent objects. This option is the default condition.

-c	Suppress the link-edit phase of the compilation and produce an object file for each source file. For a single file, you can change the output file name with the -o option.
-C	Generate code for run-time subscript range checking. If an array subscript is outside the declared array bounds, this option causes compiled code to abort with an error message.
-E*arg*	Use *arg* as an EFL flag argument when processing a .e file.
-f *filename*	Specifies *filename* as an input file.
-F	Run the EFL and RATFOR preprocessor on relevant files, and put the result in files with a .f file extension.
-g	Generate additional information needed by the debugger. This option suppresses all optimizations.
-G*c*	Stop after the specified pass and retain all intermediate files. *c* can be either of the following: 1 first pass q PCO pass
-iwarp	Specifies that the code is to be executed on the iWarp system. This option is useful for makefiles.
-K	Invoke PCO to generate an intra-file inlined intermediate file. Arguments whose names end with .c or .f are compiled and intra-file inlined by PCO. Each intermediate file is left in a file whose name is that of the source file with .K substituted for .c or .f.
-l*x*	Link with the object library lib*x*.a.
-L*dir*	Add *dir* to the list of directories searched for libraries by the linker. Directories specified by the -L option are searched before the standard directory, /iwarp/lib.
-m	Apply the M4 preprocessor to each EFL or RATFOR source file before transforming with the efl or ratfor processors. This option can also be specified as -m4.
-M	Merge the C or FORTRAN and generated iWarp assembler files into a file suffixed .m. For a single file, you can change the output file name with the -o option.
-MM	Same as -M, but incorporate the text of included files in the listing file.

-N*letter nnn*

Change the size of the table defined by *letter* to *nnn*. The compiler provides a diagnostic message when a table overflows. Letter can be any of the following (the default table size, *nnn*, is included:

q	equivalences (150)
x	common blocks, subroutine, and function names (200)
s	statement numbers (2000)
c	depth of loops, if-then-elses (20)
n	variable and common block names (401)
l	labels for computed and assigned gotos and the number of alternate returns (125)

-onetrip

Compile DO loops so they are performed at least once if reached. Without -onetrip, FORTRAN DO loops are not performed at all if the upper limit is smaller than the lower limit.

-o *filename*

Specifies *filename* as the name of the object file, overriding the default name a.out. When used with the -c, -S, or -M options, the default output file name will be overridden only if a single source file is specified.

-O[*level*]

Invoke optimization as indicated by *level*, where *level* is one of the following:

0	no optimizations
1	global optimizations on intermediate code (PCO)
2	basic block compaction, generation of Compute and Access instructions, and PCO optimizations
3	software pipelining, basic block compaction, and PCO optimizations

If *level* is not specified, this option will invoke all available optimizations.

-p

Produce code that counts the number of times each routine is called. If link editing takes place, replace the standard startup routine with one that automatically calls the monitor at the start and arranges to write a mon.out# file at normal termination of execution of the object program (# is the cell number). This allows generation of an execution profile with the iwprof command.

-Q*options*

Invoke optimizations as defined by *options*, where *options* can be any or all of the following:

G	invoke global optimizations (PCO)
C	compaction of basic blocks, generation of compute and access instructions
S	software pipelining, compaction of basic blocks, and generation of Compute and Access instructions

If compile time is an issue, but you want to do some optimization, use the -O*level* option to reduce the amount of optimization.

-R*arg*

Use *arg* as a RATFOR flag argument when processing a .r file.

-S

Compile the specified FORTRAN programs and leave the assembler-language output in corresponding files suffixed .s. For a single file, the output file name can be changed with the -o option.

-static	Make local variables persistent. This option does not affect variables explicitly declared as AUTOMATIC. Use this option only when it is necessary to run non-standard programs that require static variables. Use of this option negatively impacts optimization.
-T	Retain all intermediate files in the same directory as the source code.
-target iwarp	Specifies that the compiled code will be executed on the iWarp system. This option is useful for makefiles.
-u	Make the default type of a variable undefined.
-U	Treat upper and lower cases as distinct.
-V	Display the version of the F77 compiler and all invoked passes, and perform compilation as specified.
-V*options*	Invoke the specified *options* during compilation. *options* can be any of the following. The default conditions are in bold typeface:

check	same as check=bounds, check=overflow, check=underflow
nocheck	same as check=nobounds, check=nooverflow, check=nounderflow
check=all	same as -Vcheck
check=none	same as -Vnocheck
check=bounds	check array subscript bounds
check=nobounds	do not check array subscript bounds
check=overflow	check for floating-point overflow
check=nooverflow	do not check for floating-point overflow
check=underflow	check for floating-point underflow
check= nounderflow	do not check for floating-point underflow
d_lines	compile lines with “d” or “D” in column 1
nod_lines	treat lines with “d” or “D” in column 1 as comments
debug	same as -Vd_lines
nodebug	same as -Vnod_lines
extend_source	extend the range of FORTRAN source to columns 1-132; lines over 132 characters will stop compilation
noextend_source	restrict source to columns 1-72
f77	compile as standard FORTRAN 77 code
nof77	compile using FORTRAN 66 interpretations in cases of incompatibility

-vms	Enable VMS extensions. This is the default.
-VV	Display the version of the F77 compiler and all possible passes, but do not perform compilation.
-w	Suppress all warning messages.
-w66	Suppress only FORTRAN 66 compatibility warnings.

-Wc, *arg1* [, *arg2...*]	Hand off any specified arguments to pass *c*, where *c* is one of the following:

0	compiler first pass
q	PCO pass
1	compiler second pass
a	assembler
l	link editor
f	iwcrt0 and mcrt0
m	listing merger
s	sort
c	iwcc
e	EFL
r	RATFOR

-Yc, *dir*	Get pass *c* from directory *dir*, where *c* is one of the following:

0	compiler first pass
q	PCO pass
1	compiler second pass
a	assembler
l	link editor
f	iwcrt0 and mcrt0
m	listing merger
s	sort
c	iwcc
e	EFL
r	RATFOR

iWarp Assembler

The iWarp assembler is a cross-assembler that runs on the Sun host and produces code for the iWarp system. You can use the assembler directly to translate assembly language programs, or indirectly as a compiler option. The assembler output file contains the following:

- binary iWarp operation codes
- relocation information
- a symbol table
- symbolic debugging information

Assembler output is suitable for processing by the iWarp linker. The assembler produces object code files in COFF.

Invoking the Assembler: iwas

To invoke the iWarp assembler, use the iwas command as follows:

```
iwas [option ...]  filename
```

filename is the name of an assembly language source file. Only one source file may be specified.

option can be any of the following options.

Assembler Options

-#	Print the active command line options.

-##	Print the active command line options and a message listing permitted command line options.
-dl	Remove the line number entries from the symbol table of the object file.
-L	Create a listing file with the extension .l.
-m	Use the m4 preprocessor on the source file. Uses macro definitions in /iwarp/lib/cm4defs by default.
-o *filename*	Specifies *filename* as the name of the object file and overrides the default name.
-O	List intermediate file information for debugging.
-rom	Prohibit Compute and Access instructions from straddling a 64-byte boundary. The default does not allow straddling a 256-byte boundary.
-unlink	Remove the source file on completion of assembly. Useful when assembly is used in a chain of programs, as with **iwcc**.
-V	Write the version number of the assembler.
-VV	Write the version number of the assembler on standard error output and suppress assembling the source file.
-W	Suppress warning messages.
-XA	Enable the use of the alternate register names r0-r127 (general purpose registers) and cs000-cs3ff.
-XD	Dump the internal tables of machine-specific names. The file named on the command line is not assembled.
-XF	List all instructions, registers, and condition codes.
-XN	Suppress printing of the date on listings.

iWarp Linker

The iWarp linker combines object files into a single file. The iWarp linker is a cross-linker that executes on the Sun host. This linker maps input files onto output files and leaves a completely linked object file for the iWarp loader. Key features of the linker are:

- links multiple object files into a single object program
- resolves external global addresses
- provides a special command language to specify explicit link instructions
- provides a default memory map that matches the logical address space
- allows the user to override the default memory layout and to specify a different layout of memory, into which the linked object is to be mapped
- binds output sections to default machine addresses
- binds output sections to specific addresses, overriding the defaults
- places sections relative to one another through explicit commands
- binds individual names to specific addresses per explicit commands
- creates a load map
- associates input sections with specific addresses chosen by the user

Invoking the Linker: iwld

To invoke the iWarp linker, use the iwld command as follows:

```
iwld [option ...]  filename ...
```

filename is the name of an object file.

option can be any of the following options.

Linker Options

-e *entry*	Define *entry* as the entry point of the program.
-l*x*	This option is an abbreviation for the library name lib*x*.a, where *x* is a string. This option is used with the -L option.
-L*dir*	Add *dir* to the list of directories that contain object library routines. Directories specified with -L are searched before the standard directory, /iwarp/lib. This option only affects -l*x* options which follow on the command line.
-m	Produce a simple load map, listing the names of the files which will be loaded plus their start addresses, input sections, output sections, and lengths.
-o *filename*	Specifies *filename* as the name of the object file, overriding the default name a.out.
-r	Prevent final definitions from being given to common symbols, and suppress undefined symbol diagnostics.
-s	Removes the symbol table and relocation information to save space.
-u *name*	Enter *name* as an undefined symbol.
-x	Preserve only global symbols in the output symbol table.

iWarp Disassembler

The iWarp disassembler translates a machine language file and produces an assembly language source file suitable for the iWarp assembler. The input must be a COFF file.

Invoking the Disassembler: diwas

To invoke the iWarp disassembler, use the diwas command as follows:

```
diwas [option ...]  filename
```

filename is the name of a COFF object file. Only one file may be specified.

option can be any of the following options.

Disassembler Options

-#	Print version information and executes disassembly.
-##	Print version information, but does not execute disassembly.
-a	Print ASCII interpretations of data.
-b	Produce byte-mode addresses. This is the default.
-B	Suppress printing of byte-mode addresses.
-c	Interpret all file contents as instructions.

-d	Dump the symbol table for debugging. A list of the symbols contained in the symbol table is included in the output. Comment markers precede the list of symbol table entries so that the source code produced by the -q option can still be directly assembled.
-D	Produce listing style output using obsolete double word addresses.
-f	Treat the input file as if the object file being disassembled had been assembled with the .flags pseudo-operation. When the -f option is used, all instructions that do not affect flags are preceded by the prefix *noflags*, while instructions that do affect flags have no prefix. The default behavior is the reverse: instructions that do not affect status flags have no prefix, while instructions that do affect flags have the prefix *flags*.
-H *path*	Define *path* as the path to the directory where the disassembler creates its symbol and hash table.
-i	Force the specified disassembly range to be initialized data rather than separate data and text sections.
-l	Invoke listing mode, producing hexadecimal location and object code output. This is the default.
-L *path*	Define *path* as the path to the standard symbol mnemonics (registers, condition codes).
-n	Create a new hash table for symbols.
-o *filename*	Specify *filename* as the name of the disassembly output file and overrides the default name.
-q	Suppress listing mode, causing the output to look like assembly language source code.
-r	Disable symbolization of addresses. The disassembler prints only addresses, not labels, as the operands of instructions.
-R *start:end*	Define a range between address *start* and address *end* for disassembly. *start* and *end* are hexadecimal byte addresses. The default for *start* is 0, and the default for *end* is infinity. The -R option suppresses headers and trailers.
-s	Enable symbolization of addresses. The disassembler uses what symbol table information is available to try to replace numeric addresses by labels in instructions. This is the default.
-t	Print elapsed user and system times for the phases of disassembly: load, startup, symbol table initialization, and code disassembly.
-u	Reuse the symbol hash table constructed at the last invocation of the disassembler.
-V	Print the disassembler version number.
-VV	Print the disassembler version number and terminate.
-w	Produce listing style output using obsolete word addresses.
-W	Specify that the labels stored in the symbol table are given as word addresses.
-XA	Handle unknown condition codes and control space registers by using generic names.
-XD	Compile the standard symbol names from the text output produced by the assembler into a fast binary format.

iWarp Profiler

The profiler generates profile information from a file produced by the monitor function. The profiler lists the percentage of time spent executing between the address of each external symbol and the following symbol. The profiler also lists the number of times each function was called and the average time per call.

You must use the iwcc -p option when compiling to generate a profile file.

Invoking the Profiler: iwprof

To invoke the iWarp profiler, use the iwprof command as follows:

`iwprof [option ...]  filename`

filename is the name of an object file.

option can be any of the following options.

Profiler Options

-a	Sort by increasing symbol address.
-c	Sort by decreasing number of calls.
-h	Suppress the heading normally printed on the report.
-m *mdata*	Use the file *mdata* instead of mon0.out as the input profile file.
-n	Sort lexically by symbol name.
-o	Print each symbol address (in octal) along with the symbol.
-s	Print a summary of the monitoring parameters and statistics on the standard error output.
-t	Sort by decreasing percentage of total time. This is the default.
-V	Print iwprof version information on the standard error output.
-x	Print each symbol address (in hexadecimal) along with the symbol name.
-z	Include all symbols in the profile range, even if associated with zero number of calls and zero time.

iWarp Debugger

The iWarp symbolic debugger, `iwdb`, provides a symbolic source code debugger for cell programs that run on iWarp arrays. The iWarp debugger is based on the public domain GNU debugger, `gdb`, with extensions added to handle the additional functionality of the iWarp array.

The iWarp debugger is a line-oriented, interactive tty interface that allows debugging on any iWarp array connected to any server in the network. The iWarp debugger supports the following operations:

- start a program
- make the program stop on specified conditions
- examine what has happened when the program has stopped so you can identify bugs
- change things in the program, so you can correct the effects of one bug and go on to learn about another without having to recompile

The iWarp debugger also supports a window-based user interface, `iwdb_tool`, that supports the full set of features of `iwdb`. You can invoke `iwdb` or `iwdb_tool` on any node in the network that has RPC access to an iWarp array server node. In addition, `iwdb_tool` requires a graphic display running X11, release 5 or later.

For a complete description of the iWarp debugger, refer to the *iWarp Debugger Manual*.

Invoking the Debugger: iwdb

To invoke the debugger, use the iwdb command as follows:

```
iwdb [option ...]
```

option can be any of the following options.

iwdb Options

`-nx`	Do not execute commands from the init files .iwdbinit. Normally, the commands in these files are executed after all command options and arguments have been processed.
`-q`	Do not print introductory messages.
`-batch`	Run in batch mode. Exit with code 0 after processing the command files specified with `-x` and `.iwdbinit`, if not inhibited. Exit with nonzero status if an error occurs in executing the debugger commands in the command files.
`-x` *file*	Execute debugger commands from *file*.
`-d` *directory*	Add *directory* to the path to search for source files.
`-as` *nodename*	Identify *nodename* as the array server network name. *nodename* must be a valid network machine name and correspond to an entry in the iWarp configuration file. The default is the local machine on which the debugger is running.
`-sib` *sibnum*	Use the SIB defined by *sibnum*. *sibnum* must correspond to an entry in the iWarp configuration file. The default value for *sibnum* is zero.
-pm	Specifies post-mortem debugging. Refer to the section *Post-Mortem Debugging* in Chapter 3 for a description of post-mortem debugging.

There are many debugger commands available with the iWarp debugger. These commands are described in the *iWarp Debugger Manual*.

File Manipulation Tools

The iWarp file manipulation tools operate on COFF files. COFF is an acronym for common object file format. The COFF definition describes a complex data structure that represents object files, executable files, and archive files. The COFF data structure defines fields for things such as machine code, relocation information, and symbolic information.

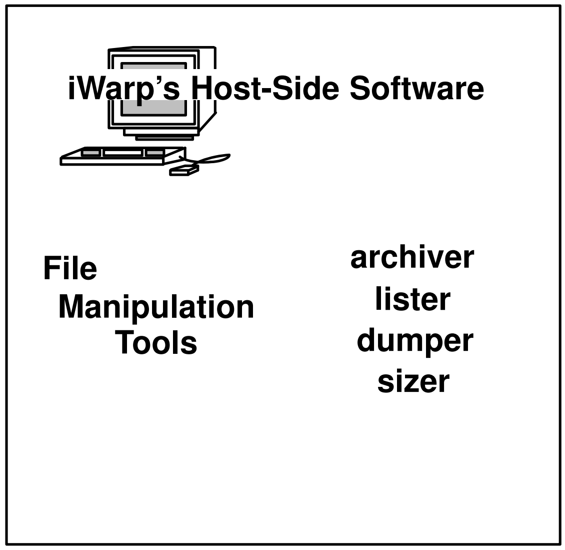
Key features of COFF are:

- applications may add system-dependent information to the object file without causing access utilities to become obsolete
- COFF provides space for symbolic information used by debuggers and other applications

- users may make some modifications in the object file construction at compile time

If you are unfamiliar with UNIX Common Object File Format and want to know more, we suggest two books;

- *Understanding and Using Coff* by Gintaras R. Gircys, O'Reilly and Associates, Inc..
- *UNIX System V/386 Programmer's Guide*, Prentice Hall.



iWarp Archiver

The iWarp archiver maintains groups of files combined in a single archive file. Its main purpose is to create and update library files to be used by the linker. The magic string and the file headers used by the archiver consist of printable ASCII characters. If an archive is composed of printable files, the entire archive is printable.

Invoking the Archiver: iwar

To invoke the iWarp archiver, use the iwar command as follows:

`iwar key [posname] [option ...] afile [filename ...]`

posname is the name of a file in the archive used as a reference point to position other files in the archive.

afile is the name of an archive file.

filename is the name of a file within the archive file.

A single *key* must be included, and it can be any of the following:

Archiver Keys

d	Delete the named files from the archive file.
m	Move the named files either to the end of the archive, or the position specified by a positioning argument and *posname* as in the case of the r key.
p	Print the named files. If no names are given, all files in the archive are written to the standard output in the order they appear in the archive.

q	Append the named files to the end of the archive file without searching the archive for duplicate names. This key is useful only to avoid quadratic behavior when creating a large archive piece-by-piece. If this key is used to add a member to an archive, and a member with the same name as that member already exists in the archive, the old member will not be removed; two members with the same name will exist in the archive. Optional positioning characters are invalid with this key.
r	Replace the named files in the archive. If the optional character u is used with r, only those files with dates of modification later than the archive file are replaced. If an optional positioning character a, b, or i is used, *posname* must be present to specify that new files are to be placed after or before *posname*. Otherwise, new files are placed at the end.
t	Print a table of contents of the archive file. If no names are given, all files in the archive are listed.
x	Extract the named files. If no names are given, all files in the archive are extracted and moved into the current directory. The archive is not altered.

option can be any of the following options.

Archiver Options

-a	Place new files after *posname*. This applies only to the r and m keys.
-b	Place new files before *posname*. This applies only to the r and m keys.
-c	Create an archive. Suppress the message that is produced by default when the archive is created.
-i	Identical to the b option.
-l	Place temporary files in the current working directory rather than in the default temporary directory, TMPDIR.
-s	Force the regeneration of the archive symbol table even if the archiver is not invoked with a command that will modify the archive contents. This command is useful for restoring the archive symbol table after the strip command has been used on the archive.
-u	Replace those files that have changed since they were put in the archive. -u is used with the r key.
-v	Use verbose mode. When used with the r, d, m, or q keys, give a file-by-file description of the creation of a new archive file from the old archive and the constituent files. When used with x, give a file-by-file description of the extraction of archive files. When used with t, give a long listing of information about the files. When used with p, write each member's name to the standard output before writing the member to the standard output.

iWarp Symbol Table Lister

The symbol table lister displays the symbol table of common object files. The object files can be relocatable or absolute common object files. They can also be archives of relocatable or absolute common object files. For each symbol, the following information is printed;

- the name of the symbol
- the value of the symbol, expressed as an offset or an address depending on its storage class
- the storage class of the symbol
- the type and derived type of the symbol
- the size of the symbol in bytes (if compiled with the iwcc -g option)
- the source line number at which the symbol is defined (if compiled with the iwcc -g option)
- The object file section that contains the symbol (for storage classes static and external).

Each symbol name in the list is preceded by its value and one of the following:

A	absolute
B	bss segment
C	common symbol
D	data segment symbol
f	filename
t	static function symbol
T	text segment symbol
U	undefined

Invoking the Symbol Table Lister: iwnm

To invoke the iWarp symbol table lister, use the iwnm command as follows:

```
iwnm [option ...]  filename ...
```

filename is the name of an object file or an archive of object files.

option can be any of the following options.

Symbol Table Lister Options

-e	Print only external and static symbols.
-f	Produce full output. Print redundant symbols (.text, .data, .lib, and .bss) normally suppressed.
-h	Do not display the output header data.
-n	Sort external symbols by name before they are printed.
-o	Print the value and size of a symbol in octal instead of decimal.

-p	Produce easily parsable, terse output. Each symbol name is preceded by its value (blanks if undefined) and one of the following letters: U (undefined) A (absolute) T (text segment symbol) D (data segment symbol) S (user defined segment symbol) R (register symbol), F(file symbol) C (common symbol) If the symbol is local (non-external), the type letter is in lower case.
-r	Prepend the name of the object file or archive to each output line.
-T	Truncate every name which would otherwise overflow its column, and place an asterisk as the last character in the display name to mark it as truncated. By default, iwnm prints the entire name of the symbols listed. Since object files can have symbol names with an arbitrary number of characters, a name that is longer than the width of the column set aside for names will overflow its column and force every column after the name to be misaligned.
-u	Print only undefined symbols.
-v	Sort external symbols by value before they are printed.
-V	Print the version of the iwnm command executing on the standard error output.
-x	Print the value and size of a symbol in hexadecimal instead of decimal.

iWarp Dumper

The dumper dumps selected parts of object files. The dumper processes both object files and archives of object files.

Invoking the Dumper: iwdump

To invoke the iWarp dumper, use the iwdump command as follows:

```
iwdump [option ...]  filename ...
```

filename is the name of an object file or an archive of object files.

option can be any of the following:

Dumper Options

-a	Dump the archive header of each member of each archive file specified.
-c	Dump the string table.
-f	Dump the file header.
-g	Dump the global symbols in the symbol table of an archive.
-h	Dump section headers.
-l	Dump line number information.
-L	Interpret and print the contents of the .lib sections.
-o	Dump each optional header.

-r	Dump relocation information.
-s	Dump contents of sections.
-t	Dump symbol table entries.
-z *name*	Dump line number entries for the named function.

The following modifiers can be used with the iwdump options:

Dumper Option Modifiers

-d *number*	Dump the section number, number, or the range of sections starting at number and ending the number specified by +d.
+d *number*	Dump sections in the range either beginning with first section or beginning with section specified by -d.
-n *name*	Dump information pertaining only to the named entity. This modifier can be used with -h, -s, -r, -l, -t
-p	Suppress printing of the headers.
-t *index*	Dump only the indexed symbol table entry. The -t used in conjunction with +t, specifies a range of symbol table entries.
+t *index*	Dump the symbol table entries in the range ending with the indexed entry. The range begins at the first symbol table entry or at the entry specified by the -t modifier.
-u	Underline the name of the file for emphasis
-v	Dump information in symbolic representation rather than numeric. This modifier can be used with all the above options except -s and -o options of iwdump.
-z *name, number*	Dump line number entry or range of line numbers at *number* for the function *name*.
-+z *number*	Dump line numbers starting at either function *name* or *number* specified by -z, up to *number* specified by +z.

iWarp Sizer

The sizer produces section size information in bytes for each loaded section in common object files. The size of the text, data, and bss sections are printed, as well as the sum of the sizes of these sections. The sizer also produces size information for all members of an archive. The size of uninitialized non-static global data structures is not included in the size information.

Invoking the Sizer: iwsize

To invoke the iWarp sizer, use the iwsize command as follows:

`iwsize [`*`option`*` ...]  `*`filename ...`*

filename is the name of an object file or an archive of object files.

option can be any of the following options.

Sizer Options

-f	Print the size of every loaded section, followed by the section name in parentheses.
-n	Include NOLOAD sections in the size.
-o	Print numbers in octal.
-V	Supply version information on the iwsize command.
-x	Print numbers in hexadecimal.