2 The 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:

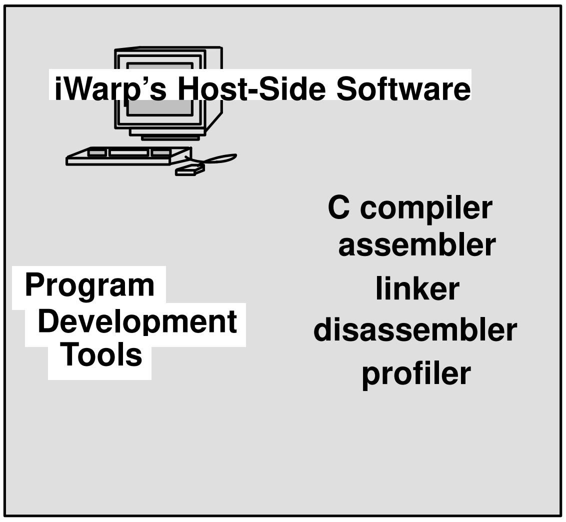
- C compiler
- assembler
- linker
- disassembler
- profiler

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.

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, running on the Sun host and producing 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*	*letter* can be C, L, or S. C suppresses the automatic use of a default link control file. L suppresses automatic linking with libc.a (and libg.a if -g is used). S suppresses 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 PCI and PCO optimizations
3	software pipelining, basic block compaction, and PCO optimizations

NOTE

Software pipelining and basic block compaction will be available in the next release of the software.

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

NOTE

Software pipelining and basic block compaction will be available in the next release of the software.

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.

-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)

The iWarp assembler

The iWarp assembler is a cross-assembler, running on the Sun host and producing 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, overriding the default name.
-O	List intermediate file information for debugging.
-rom	Prohibit data 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.
-XN	Suppress printing of the date on listings.

The 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 are the linker are:

- links multiple object files into one 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 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	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.

The 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.
-b	Produce byte mode addresses. This is the default.
-B	Suppress printing of byte mode addresses.
-c	Decode the given disassembly range as instructions rather than data.
-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 psuedo-operation. When the -f option is used, all instructions which do not affect flags are preceded by the prefix *noflags* , while instructions which do affect flags have no prefix. The default behavior is the reverse: instructions which do not affect status flags have no prefix, while instructions which 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, overriding 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 .
-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.

The 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.

File Manipulation Tools

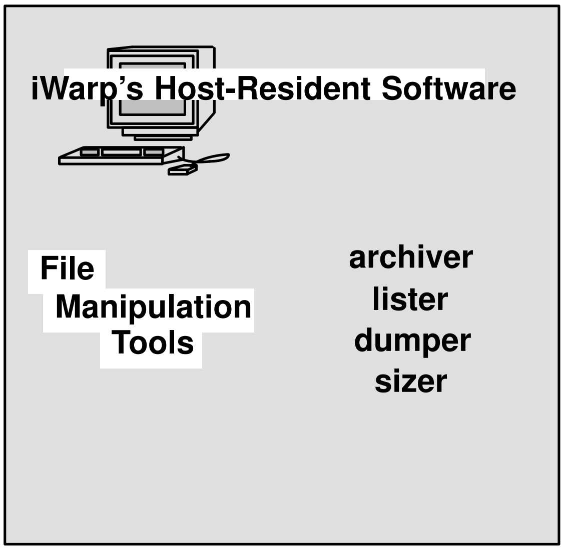
The iWarp file manipulation tools operate on COFF files. COFF is an acronym for Common Object File Format. The COFF definition describes a complex data structure that represents object files, executable files, and archive files. The COFF data structure defines fields for things such as machine code, 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.



The 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, then 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 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 to restore 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 form 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.

The 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 letters U (undefined), A (absolute), T (text segment symbol), D (data segment symbol), S (user defined segment symbol), R (register symbol), F(file symbol), or 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, forcing 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.

The 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 options.

Dumper Options

-a	Dump the archive header of each member of each archive file specified.
-c	Dump the string table.
-f	Dump the header file.
-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 sections contents.
-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.

The 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 will also produce 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.