**For: dwight**
**Printed on: Fri, Jul 16, 1993 10:04:27**
**From book: guide**
**Document: ch6**
**Last saved on: Fri, Oct  2, 1992 15:05:27**

# 6 PCS Tools

The PCS programming tools are for developing parallel applications for iWarp systems. The PCS tools reside on the iWarp host, a Sun Microsystems UNIX-based workstation. The PCS tools include the following:

- PCS preprocessor (iwpp)
- PCS array combiner (iwcb)
- PCS array loader (iwamld)

You use these tools together with the iWarp C compiler (iwcc) to develop your applications.

To develop cell programs, you use the PCS data movement library. To develop array programs, use the combiner library. For more information about these libraries, see Chapter 5, "Program Communication: PCS."

For more information about developing PCS applications, see Chapter 7 "Developing iWarp Programs."

## PCS Preprocessor

The PCS preprocessor preprocesses cell program source files before you compile them. The preprocessor takes source code that contains PCS data movement library calls and expands the code to equivalent inline code. It also runs the source code through the iWarp C preprocessor. The PCS preprocessor writes the result to standard output. Preprocessing is an optional step in developing your PCS applications.

Preprocessing cell programs makes your application code run faster by avoiding function call overhead. If you do not preprocess your cell programs, the PCS array loader links in the PCS data movement library and resolves the definitions for PCS calls in the cell program.

**NOTE**

The iwpp command and the iwamld command provide error and non-error checking versions. The non-error checking versions use an optimization of inline PathLib code. You must keep your error checking levels consistent throughout your application. For example, if you use iwpp –c0 for preprocessing, you must use iwamld –c0 for loading. If you use iwamld –c0 for loading, you must have preprocessed you code using iwpp –c0 or compiled your code with using the –DPL_INLINE option. Failure to do this prevents your application from linking.

Using the non-error checking versions can be a significant performance improvement for your application. It is recommended that you develop your application with error checking first for debugging, then build with non-error checking for your final runtime version.

### Invoking the PCS Preprocessor: iwpp

To invoke the PCS preprocessor, use the iwpp command as follows:

```
iwpp [options] program
```

*program* is the source file containing the cell program source code.

### Preprocessor Options

| | |
|---|---|
| -c *error_level* | Set the error level of the included code. Error level 1 includes all error checking. Error level 0 includes no error checking. Default is level 1. |
| -D define | Run the C preprocessor with define option specified. |
| -I *dir* | Search the *dir* directory in addition to the standard directories for include files. |
| -V | Print the version number of iwpp and continue. |
| -VV | Print the version number and usage information for iwpp and exit. |

## PCS Array Combiner

The PCS array combiner creates an array module that can execute on an iWarp array. The array combiner creates an array module by combining the following:

- cell programs compiled to the iWarp object format
- existing array modules
- network information

You can combine several array modules and cell programs to create larger array modules.

The array combiner takes an array program, compiles it, and runs it to create an array module. An array program provides the following information about an array module:

- the cell programs and array modules to include in the new array module
- the ports that cell programs use to access a networks
- the network connections and routes between cells on the target iWarp system
- the cell program placement on array cells

Using the `-col` and `-row` command line options, you can specify the number of rows and columns for the array module. The array program works on an array subsection of that size. The default size is an 8 x 8 array.

## Automatically Placement and Routing

If an array program does not specify cell IDs for the cells in a network it creates, the array combiner automatically specifies the cell program placement and network routing. There are a limited number of cells in a physical array and only one cell program can be mapped to a cell. The array combiner searches the array for available cells. It places the cell programs and determines the network route for your application. The number of cells in the network for your application cannot exceed the number of cells available on the physical array.

The array combiner automatically routes point-to-point networks. It cannot automatically route complex networks, such as torus networks or dynamic networks.

For an example showing automatic placement and routing, see Chapter 7, “Developing iWarp Programs.”

## Invoking the PCS Array Combiner: iwcb

To invoke the PCS array combiner, use the iwcb command as follows:

```
iwcb [options] array_program
```

*array_program* is the source file name for your application’s array program.

## Combiner Options

| | |
|---|---|
| -d | Print debugging information and suppress removing the executable array program and the array network description file archived in the array module. |
| -col *num* | Specify an array grid with *num* columns. By default the grid has 8 columns. |
| -row *num* | Specify an array grid with *num* rows. By default the grid has 8 rows |
| -o *output* | Write the array module to a file specified by *output*. The default file name is *array.out*. |
| -w | Specify the array grid as a torus and use the back edges in routes. By default, an iWarp array does not use back edges. |
| –D*define* | Compile the array program with *define* defined for the preprocessing phase. |
| -I *dir* | Search the *dir* directory for files to be included in the array program, in addition to the standard include files. |
| -L *dir* | Search the *dir* directory for library files, in addition to the standard library directories . |
| -M *dir* | Search the *dir* directory for modules to be included in the array module, in addition to the current directory and the */iwarp/lib/modules* directory. |
| -V | Print the version number and continue. |
| -VV | Print the version number and usage information, then exit. |
| –Yc,*dir* | Get pass *c* from directory *dir*, where *c* is pass A indicating the location of the combiner library. |

# PCS Array Loader

The PCS array loader loads an array module onto an iWarp system and executes the array module. The loader does the following during the loading process:

- checks that the array module has sufficient resources to run on the target iWarp array
- creates and compiles the initialization information for the array module
- links the cell programs with the system libraries and initialization code
- loads the cell programs onto the appropriate array cells
- creates a run script for the array module
- starts the cell programs executing

By default, the array loader loads and runs the array module. Use the -n option to request that the array loader not run the array module.

**NOTE**

The iwpp command and the iwamld command provide error and non-error checking versions. The non-error checking versions use and optimization of inline PathLib code. You must keep your error checking levels consistent throughout your application. For example, if you use iwpp –c0 for preprocessing, you must use iwamld –c0 for loading. If you use iwamld –c0 for loading, you must have preprocessed you code using iwpp –c0 or compiled your code with using the –DPL_INLINE option. Failure to do this prevents your application from linking.

## Invoking the PCS Array Loader: iwamld

To invoke the PCS array loader, use the iwamld command as follows:

```
iwamld array_module [options]
```

*array_module* is the name of the array module you are loading.

## Array Loader Options

| | |
|---|---|
| -c *error_level* | Link in the PCS library with the specified error level. Legal error levels are 0 and 1. 1 means all error checking. 0 means no error checking. |
| -d | Print additional debugging information. |
| -e | Execute the array module. This is the default. |
| -n | Do not execute the array module. Creates a script to run array module later. |
| -offset *row*,*col* | Load the the array module by placing all cell programs at the offset specified by the *row* and *col* arguments. For example, the cell program on cell (0,0) is loaded on cell (*row*,*col*). |

-*row*[..*row*],*col*[..*col*],arg[,*arg*...]
: Pass the specified arguments to the program on cell(row, col). Specify ranges using *row..row*,*co*l..*co*l for contiguous blocks only. For example, using 0..1,0..1 specifies giving the same arguments to the four cells in the upper left corner of the array.

-H *host_name*
: Execute the array module on the iWarp array attached to the host system with the name *host_name*. By default iwload tries to execute the array module on the current machine. This option overrides IWARP_HOSTNAME environment variable.

–S *sib_num*
: Execute the array module on the iWarp system attached to the SIB, *sib_num*. By default, iwamld tries to execute the array module attached to SIB 0. This option overrides the IWARP_SIBNUM environment variable.

–W*c*,*arg*[,*arg*...]
: Hand off the argument(s) *arg* to pass *c* where *c* is one of [01] indicating iwcc compilation of init files or iwcc final link of the executable, respectively.

–l*x*
: This option is an abbreviation for the library name libx.a, where *x* is a string. You can provide additional libraries for linking using this switch. At link time, the PCS array loader searches for the library in any directories provided with the –L switch (on the command line or using the –W1 switch). This is a shortcut alternative to using –W1.

–L*di*r
: Add *dir* to the list of directories in which libraries are searched by iwld during the link phase. Directories specified with –L are searched before the standard directory /iwarp/lib. This is a shortcut alternative to using –W1.

–Y*c,dir*
: Get pass *c* from directory *dir*, where *c* is one of [01ID] indicating the iwcc compilation of init files, iwcc final link of executables, PCS initialization library, or PCS data movement library, respectively.

-V
: Print the version number and continue.

-VV
: Print the version number and usage information, then exit.

# PCS Array Module Display

The PCS array module display (iwmd) is a Motif interface for displaying the logical array and networks defined in an array module. It also displays additional information about any object you point at with the mouse. A dashed line is used to show an array module that is included in another array module.

## Invoking the PCS Array Module Display: iwmd

To invoke the PCS array module display, use the iwmd command as follows:

```
iwmd array_module
```

*array_module* is the name of the array module you are loading. The array module must exist in the current directory.

## Using PCS Tools to Develop iWarp Programs

You use the PCS tools in the following sequence to develop iWarp programs:

1. Use the PCS preprocessor (iwpp) to preprocess the cell programs. This is an optional step that optimizes the cell program code.
2. Use the iWarp C Compiler (iwcc) to compile the cell programs into object files. Use the -c compiler option to suppress linking and loading for the cell programs.
3. Use the PCS array combiner (iwcb) to compile the array program and create the array module.
4. Use the PCS array loader to map the array module to a physical array, create network initialization information, load the array module, and run the array module.

Figure 6-1 shows the sequence of steps for developing PCS applications.

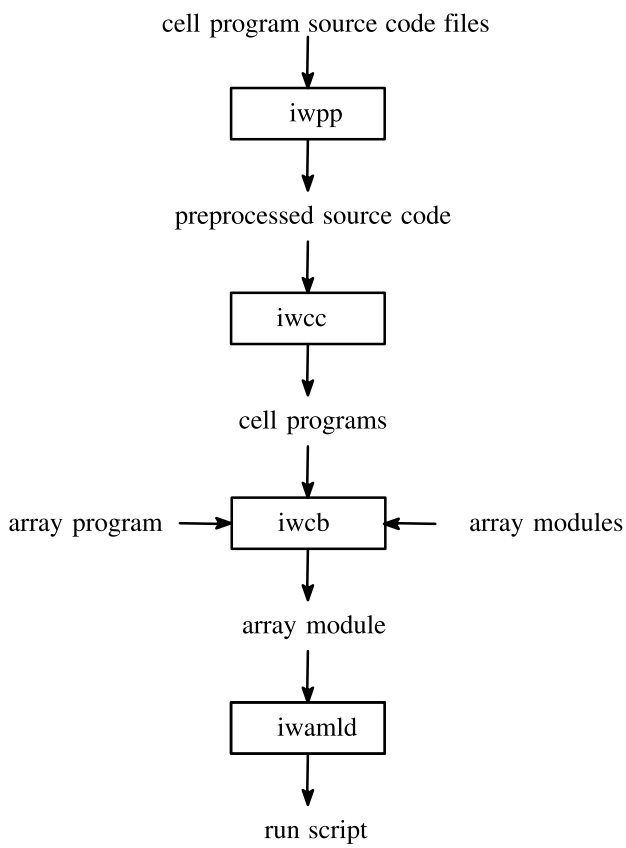


*Figure 6-1: PCS Application Development*

You use the PCS tools to create array modules for your iWarp applications. The following sections describe in more detail how to develop PCS applications.

## Preprocessing Cell Programs

Use the PCS preprocessor (iwpp) to produce more efficient code for the cell programs. Preprocessed code produces more efficient code at compile time and the applications execute faster at runtime.

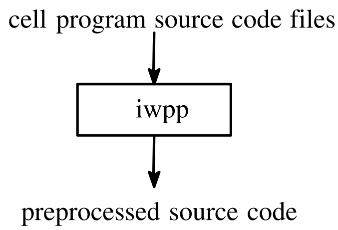


*Figure 6-2: Preprocessing Cell Programs*

Figure 6-2 shows that the preprocessor takes the source code files for the cell programs as input and produces preprocessed source code. The preprocessed source code is sent to standard output.

The preprocessor does the following to cell program source code files:

- expands PCS calls to equivalent PathLib calls
- uses assembler code where it is most efficient
- produces inline code to avoid function call overhead

Preprocessing your cell programs is optional. If you do not preprocess your source code, the compiler compiles your code, but does not expand the PCS code.

## Compiling Cell Programs

Use the iWarp C compiler (iwcc) to compile the cell programs' source code. Because the compiler is a single-cell compiler, separately compile the source code for each cell program. If you preprocessed your source code files, compile the preprocessed files instead of the original source code files. Use the -c compiler option to suppress linking and loading.

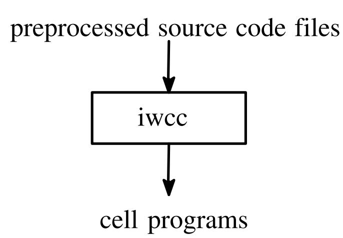


*Figure 6-3: Compiling Cell Program Source Code*

Figure 6-3 shows the input to the compiler is the preprocessed source code files. The compiler produces the cell programs. References to data movement library calls are resolved during loading.

## Creating Array Modules

Use the PCS array combiner (iwcb) to create the array module for your application. The array combiner takes as input an array program that defines the array module. The array combiner compiles and executes the array program to create the array module. Executing the array program combines the application's cell programs and, optionally, existing array modules to create the array module.

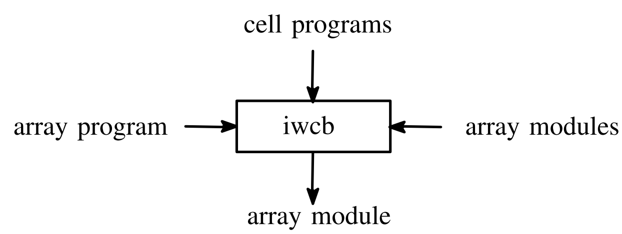


*Figure 6-4: Creating Array Modules*

Figure 6-4 shows the combiner input is the array program, the cell programs, and any array modules that make up the resulting array module.

The combiner does the following when creating array modules:

- produces placement information of the cell programs to logical iWarp cells. If a cell location is not specified in the array program, the array combiner automatically places the cell program on a cell.
- determines the network routing for the cell programs. If a network routing is not specified for the cell programs in the array program, the array combiner automatically determines the routing.
- produces network description tables for the PCS array loader
- creates an array module from the cell programs, existing array modules, and the network mapping information

After the combiner creates an array module, the array module can be loaded and executed on an iWarp system.

## Loading Array Modules

Use the PCS array loader (iwamld) to load the array module onto an iWarp system and execute the array module. The array loader sets the physical mapping for the application, and can either execute the application or provide a script to run the application later.

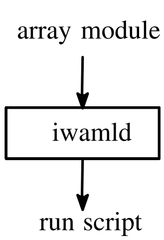


*Figure 6-5: Loading Array Modules*

Figure 6-5 shows the input for the PCS array loader is the array module for your application. The array loader loads the cell programs onto the iWarp system and produces a run script in your current directory. By default, the array loader loads and runs the array module.

The array loader does the following when it loads an array module:

- performs resource checking to make sure the application can run on the physical array
- assigns the cell program code to the physical cell it runs on
- produces communication initialization code for the network
- links the program code with the the appropriate libraries and the communication initialization code
- creates a run script for the application
- loads and runs the array module, by default

When the array loading is complete, your application is ready to execute.

## PCS Tools Example

The following example shows how to use the PCS tools to create a PCS application. The example creates a two-cell array network application that transfers data between the two cells. This section includes the commands you enter from the command line to create and run the example application. You can find the source code for this example at the end of Chapter 5, "Program Communication: PCS."

The example assumes that the example source code is in your current directory and the PCS tools and iWarp program development tools are on your execution path.

### Preprocessing the Cell Programs

After creating the source code files for your PCS applications, use the PCS preprocessor to preprocess the source code for your cell programs. The two-cell example has the following cell program source code files:

- 2cell00.c the cell program source file for cell (0,0)
- 2cell01.c the cell program source file for cell (0,1)

Enter the following command lines to preprocess these files:

```
% iwpp 2cell00.c > 2cell00.i
% iwpp 2cell01.c > 2cell01.i
```

The preprocessor output is redirected to the the following files:

- 2cell00.i the preprocessed cell program source file for cell (0,0)
- 2cell01.i the preprocessed cell program source file for cell (0,1)

### Compiling the Cell Programs

Compile the preprocessed cell program source files with iWarp C compiler. Enter the following command lines to compile these files:

```
% iwcc -O -c 2cell00.i
% iwcc -O -c 2cell01.i
```

The -O option requests the compiler to generate optimized code. The -c option requests that the compiler suppress linking, but produce an object file.

## Creating the Array Module

Create the array module by submitting the array program to the PCS array combiner. Enter the following command line to create the array module:

```
% iwcb -row 1 -col 2 -o 2cell 2cell.c
```

The combiner compiles the array program 2cell.c and creates an array module named 2cell. The -row option requests that the array module run on an array grid with one row. The -col option requests that the array module run on an array grid with two columns. The -o option requests the combiner to write the array module to an output file named 2cell.

## Loading and Running the Array Module

Load the array module using the PCS array loader. Enter the following command line to load the array module without running it:

```
% iwamld 2cell -H stargazer -S 0 -n
```

The `-H` option specifies loading the array module onto the iWarp host system named stargazer. The –S option specifies to execute the array module on the iWarp system attached to SIB 0. The `-n` options requests not to run the array module. This generates a run script named go. To execute the array module, enter the following:

```
% go
```

This executes the array module on the iWarp system.