Chapter 3

Compiling a C* Program

This chapter describes how to compile and link a C* program. It is organized as follows:

- Section 3.1 describes the compilation process.
- Section 3.2 describes basic options of the **`cs`** command.
- Section 3.3 describes **`cs`** options in common with the **`cc`** command.
- Section 3.4 discusses options more likely to be used by advanced programmers.
- Section 3.5 discusses issues involved in linking CM-5 C* programs.
- Section 3.6 explains how to compile a C* program that calls a CM Fortran subroutine.
- Section 3.7 discusses compiling a CM Fortran program that calls a C* function.
- Section 3.8 discusses symbols for which **`cs`** provides **`#defines`**.
- Section 3.9 describes how to use **`make`** with C* programs.

3.1 The Compilation Process

To compile a C* program, use the **`cs`** command. Typically, the compiler is available on a CM-5 partition manager, or on a designated server on the network. If the compiler is available on a server, it is generally a good idea to compile on it rather than the partition manager, to avoid tying up the CM-5.

The **cs** command takes a C* source file (which must have a **.cs** suffix) and produces an executable load module. The command also accepts **.c** source files, **.o** output files, **.s** assembly files, **.dp** DPEAC files, and **.a** library files, but all parallel code must be in a **.cs** file.

NOTE: **.c** files are by default passed to the Sun C compiler. This means that ANSI C features are not allowed in **.c** files, but the code will be compiled more efficiently and faster, since the C* compiler is not an efficient serial code compiler.

3.2 Basic Options

The options accepted by **cs** include some that are specific to C* and the Connection Machine system, as well as versions of **cc** options. This section describes commonly used options. All options are listed in Table 1.

Table 2. C* compiler options.

Option	Meaning
Basic options:	
–cm2 **–cm200** **–cm5**	Specify the version of the compiler to use.
–cmsim	Compile to run on a Sun-4.
–h **–help**	Give information about **cs** without compiling.
–sparc	Compile to run on a CM-5 without vector units.
–version	Print the compiler version number.
–vu **–vecunit**	Compile to run on a CM-5 with vector units.

Option	Meaning
Options in common with **cc**:	
–c	Compile only.
–D*name* **[=***def***]**	Define a symbol name to the preprocessor.
–g	Produce additional symbol table information for debugging with Prism, at the expense of slower performance.
–I*dir*	Search the specified directory for **#include** files.
–L*dir*	Add *dir* to the list of directories in the object library search path.
–l*lib*	Link with the specified library.
–o *output*	Change the name of the final output file to *output*.
–U*name*	Undefine the C preprocessor symbol *name*.
Advanced options:	
–cc *compiler*	Use the specified C compiler when compiling **.c** files.
–cmdebug	Compile for debugging (requires less execution time than **–g**).
–cmmd_root *dir*	Use *dir* as the location of the CMMD library (for use with **–node**.)
–cmprofile	Compile for Prism performance analysis.
–dirs	List the compiler's **bin**, **lib**, **include,** and **temp** directories.
–dryrun	Show, but do not execute, compilation steps.
–force	Force **.c** files through the C* compiler.
–keep *ext*	Keep the intermediate **.s**, **.o**, or **.dp** file.
–node	Link to run on a single node.
–overload	Display symbol names used by the compiler to call overloaded functions.

Option	Meaning
–S	Create an assembly source file for each input source file. (No assembly or linking performed.)
–temp *directory*	Change the location of temporary files to *directory.*
–v **–verbose**	Show the compilation and linking steps as they are executed.
–warn	Suppress warnings from the C* compiler.
–Wimplicit	Display a warning when a function is called before it is declared or defined.
–Z*comp switch*	Pass option *switch* to component *comp*, where *comp* is **cc**, **ld**, or **cmld**.

3.2.1 Choosing the Compiler: The –cm2, –cm200, –cmsim, and –cm5 Options

If you have more than one CM model at your site, separate C* compilers may be available for each machine. In any case, a version of the compiler is available for creating programs to run on a Sun-4. You invoke this version of the compiler via the **–cmsim** option; see below.

Your system administrator determines which version of the compiler you get by default when you issue the **cs** command. You can override this either by setting the environment variable **CS_DEFAULT_MACHINE** to the machine you want (**cm2**, **cm200**, **cm5**, or **cmsim**), or by specifying the **–cm2**, **–cm200**, **–cm5**, or **–cmsim** option on the **cs** command line. Using the command-line option also overrides the setting of the environment variable.

Use the **–cmsim** option to compile and link a version of your program that can run on a Sun-4 SPARC workstation. The program can't do parallel I/O, graphics, or call other non-C* library routines. Typically you would use this option to try out a program on a smaller data set before running it on a Connection Machine.

You can determine the default compilation target by simply typing **cs** at your UNIX prompt. If the resulting help message begins

```
C* driver [CM5 SPARC 7.1]
```

the target is a CM-5 without vector units. If it begins

```
C* driver [CM5 VECUNIT 7.1]
```

the target is a CM-5 with vector units.

See also the discussion of the **–vu** and **–sparc** options in Section 3.2.4.

Note that if you specify **–cm2** or **–cm200**, you should consult the user's guide for CM-200 C* for a discussion of supported compiler options.

3.2.2 Getting Help: The –help Option

Specify **–help** or **–h** to print a summary of available command-line options for **cs**, without compiling.

3.2.3 Printing the Version Number: The –version Option

Specify the **–version** option to cause **cs** to print its version number before compiling. If you do not specify a source file, **cs** simply prints the version number and exits.

3.2.4 Compiling for CM-5s with or without Vector Units: The –vu, –vecunit, and –sparc Options

Specify the **–vu** or **–vecunit** option to compile your program to run on a CM-5 that has vector units.

Specify the **–sparc** option to compile your program to run on a CM-5 without using the vector units.

The **–vu**, **–vecunit**, and **–sparc** options imply **–cm5**; if you specify any of them, you do not have to specify **–cm5** in addition.

If you specify **–cm5** but omit one of these options, you get the default for your site; typically, this is **–vu**.

3.3 Options in Common with cc: The –c, –D, –g, –I, –l, –L, –o, and –U Options

The C* compiler allows you to specify the **cc** options **–c**, **–D**, **–g**, **–I**, **–l**, **–L**, **–o**, and **–U** on the **cs** command line. See Table 2 for a brief description of these options; for more information, consult your documentation for **cc**.

Include the **–g** option if you want to debug the compiled program using Prism; see Chapter 5. Using the **–g** option increases execution time considerably; use the **–cmdebug** option instead if this matters. See Section 3.4.2.

3.4 Advanced Options

This section describes **cs** options that would typically be used by an advanced programmer. All options are listed in Table 2, above.

3.4.1 Using Another C Compiler: The –cc Option

The C* compiler works in conjunction with the standard C compiler available on your Sun partition manager or compile server. *The use of C* with other C compilers is not supported and can lead to incorrect results.* However, you can use another compiler if you want to, by including the **–cc** switch, followed by the name of the compiler. For example, to use the Gnu C compiler, specify the **–cc** option as in this example:

```
% cs -cc gcc myfile.c
```

3.4.2 Debugging Your Program: The –cmdebug Option

Use the **–cmdebug** option (instead of the standard **–g** option) to create symbol table information for debugging your program. The **–cmdebug** option speeds up execution time, at the expense of making debugging somewhat less precise. (For example, you may not be able to set a breakpoint at every executable line of code.)

3.4.3 Obtaining Performance Data: The –cmprofile Option

Use the **-cmprofile** option if you want to run your program under Prism to obtain performance data. See Chapter 5 for more information about Prism.

3.4.4 Displaying the bin, lib, include, and temp Directories: The –dirs Option

Use the **-dirs** option to find out where the compiler searches for binaries, libraries, include files, and temporary files. It produces output like this:

```
% cs -cm5 -dirs
    C* driver [CM5 SPARC 7.1]
    bin_dir is /usr/cm5/cstar/7.1/bin/
    lib_dir is /usr/cm5/cstar/7.1/lib/
    include_dir is /usr/cm5/cstar/7.1/include/
    temp_dir is /usr/temp
```

The binary search path is:

1. the "**bin_dir**" directory specified in this message
2. your **$PATH**
3. **/bin**, **/usr/bin**, and **/usr/local/bin**

The library search path is:

1. any directories you specified via the **-L** option
2. the "**lib_dir**" directory specified in the **-dirs** message
3. directories you specify via **$LD_LIBRARY_PATH**
4. **/lib**, **/usr/lib**, and **/usr/local/lib**

The include search path is:

1. any directories you specified via the **-I** option
2. the "**include_dir**" directory specified in the **-dirs** message
3. **/usr/include**

3.4.5 Displaying Compilation Steps: The –dryrun Option

Specify **–dryrun** to cause **cs** to show, but not carry out, the commands to be executed in compiling and linking.

3.4.6 Putting .c Files through the C* Compiler: The –force Option

Specify **–force** to put **.c** files through the C* compiler. Otherwise, such files are passed unread to the C compiler. You might want to specify **–force** to take advantage of the C* compiler's type checking of prototyped function declarations.

3.4.7 Keeping an Intermediate File: The –keep Option

Use the **–keep** option to keep an intermediate file with the extension you specify. Choices are:

- **s**, to keep the assembly-language source file
- **o**, to keep the object file
- **dp**, to keep the DPEAC assembly-language file; the file is named *file***.pe.dp**, where *file* is the name of the C* source code, without the **.cs** extension

Using this option does not inhibit assembly or linking.

Note for users of CM-200 C*: **–keep c** is not allowed because CM-5 C* compiles directly to assembly code.

3.4.8 Using CMMD: The –node and –cmmd_root Options

To use the CMMD message-passing library from a C* program, you must link the program with the **–node** option; this specifies that the program is to be linked so that copies of it run separately on the individual nodes. This option is supported only if you also specify **–vu** to run on the vector units.

If the CMMD library has not been installed in the standard place at your site, use the **`-cmmd_root`** option, followed by a directory name, to specify its location. Check with your system administrator for the correct location at your site. To avoid having to specify this option every time you compile a node-level program, set the environment variable **`CMMD_ROOT`** to the correct pathname.

3.4.9 Displaying Names of Overloaded Functions: The –overload Function

Use the **`-overload`** option to cause the compiler to display informational messages listing the symbol names it uses internally for overloaded functions; it displays such a message once for every call to an overloaded function being compiled. Knowing these symbol names is necessary if you want to invoke such a function directly using Prism.

3.4.10 Creating Assembly Source Files: The –S Option

Use the **`-S`** compiler option to create assembly source files for each input source file. For example:

```
% cs -S foo.cs
```

produces the files **`foo.s`** (for the program that runs on the partition manager) and **`foo.pe.s`** (for the program that runs on the processors). See Section 3.5 for more information.

You cannot combine the **`-S`** option with either the **`-c`** option or the **`-o`** option.

3.4.11 Changing the Location of Temporary Files: The –temp Option

Use the **`-temp`** option, followed by the pathname of a directory, to cause temporary files used during C* compilation to be created in that directory. Issue the **`cs`** command with the **`-dirs`** option to find out the standard location in which they are created; see Section 3.4.4.

3.4.12 Turning On Verbose Compilation: The –v or –verbose Option

Specify **–verbose** or **–v** to display informational messages as the compilation proceeds. This can be useful if you want to see which part of the compilation process produced an error message.

3.4.13 Turning Off Warnings: The –warn Option

Specify **–warn** to suppress warnings produced during C* compilation.

3.4.14 Turning On Warnings about Undeclared Functions: The –Wimplicit Option

Specify **–Wimplicit** to tell the compiler to print a warning when you call a function that has not previously been declared or defined.

3.4.15 Specifying Options for Other Components: The –Z Option

Use the **–Z** option to specify a **cc**, **cmld**, **ld**, **dpas**, or **as** option that **cs** does not recognize. Use **–Zld** only when you specify the **–cmsim** option. Use **–Zdpas** only when compiling for the vector units.

These options are passed directly to the specified component without any interpretation by **cs**. Type **–Z**, followed by the component name, followed by the option. There is no space between **–Z** and the component name; leave at least one space between the component name and the option.

For example, specify

```
% cs –Zcc –w myfile.c
```

to suppress **cc** warning messages.

3.5 Linking

This section discusses issues in linking C* programs.

3.5.1 Names of C* Run-Time Libraries

You need to be aware of the names of the C* run-time libraries only if you are using a command other than **cs** to link — for example, when linking a C* program that calls CM Fortran; see Section 3.6.

The names of the run-time libraries depend on the target of the compilation:

- If you specify **-sparc**, the libraries are **libcs_cm5_sparc_sp.a** and **libcs_cm5_sparc_pn.a** (for the nodes).
- If you specify **-vu**, the libraries are **libcs_cm5_vu_sp.a** and **libcs_cm5_vu_pn.a** (for the nodes).
- If you specify **-cmsim**, the library is **libcs_cm5_cmsim.a**.

3.5.2 Intermediate Files

The C* compiler and the CM-5 linker, **cmld**, generate a single output file that combines a scalar and a parallel executable program. As intermediate output, however, the compiler generates separate scalar and parallel files. For example:

- With the **-S** option, the compiler generates two assembly files: for example, **myprogram.s** and **myprogram.pe.s**.
- With the **-c** option, the compiler generates two object files: for example, **myprogram.o** and **myprogram.pe.o**.

NOTE: The parallel files can have **.pn** extensions instead of **.pe**.

However, the linker generates only one executable file: for example, **a.out**. There is no file **a.out.pe** corresponding to the parallel intermediate files.

If you work with intermediate files — explicitly linking object files, for instance — you need only specify the scalar file. The corresponding parallel file is linked automatically.

If you wish to disable the automatic processing of the parallel (**.pe**) intermediate file when the corresponding scalar file is specified, set the environment variable **CS_AUTO_PE_FILES** to **0**. Any nonzero value for this variable leaves the feature enabled.

In either case, recall that the separate intermediate files exist. If you copy or move intermediate files to another directory, be sure to move both the scalar and the parallel file. See below for more information on creating libraries for **.pe** files.

3.5.3 Creating Libraries

If you put object files in a library, you must remember to put the **.pe** object files in a separate node library too; see the previous section. Note these points in creating the library for **.pe** files:

- The node library's name must begin with the same name as the library containing the corresponding scalar object files, and it must end in **_pe.a** (or **_pn.a**).
- It must be in the same directory as the one that contains the corresponding scalar object files.

When linking, you then need only specify the library containing the scalar object files.

For example, these commands create object files and store them in libraries in your current working directory:

```
% cs -c -cm5 foo.cs bar.cs
% ar q libmine.a foo.o bar.o; ranlib libmine.a
% ar q libmine_pe.a foo.pe.o bar.pe.o; \
  ranlib libmine_pe.a
```

You can then link the programs in these libraries with another program by issuing a command like this:

```
% cs -cm5 prog.cs libmine.a
```

NOTE: In the example shown above, the library containing the scalar object files can also be called **libmine_sp.a**. C* will still find the corresponding **_pe** or **_pn** library automatically.

3.6 Compiling and Linking a C* Program that Calls CM Fortran

If your program includes a call to a CM Fortran subroutine, as described in Chapter 2, follow the instructions in this section.

1. Compile the C* program, using the **-c** option. For example:

   ```
   % cs -cm5 -c testcs.cs
   ```

2. Compile the CM Fortran program, also using the **-c** option. For example:

   ```
   % cmf -cm5 -c testfcm.fcm
   ```

   NOTE: If you are using CM Fortran Version 2.1, you cannot compile with the **-noaxisorder** or **-nopadding** option.

3. Issue **cmf** again to link, using the applicable format:

   - For CM-5s with vector units:

     ```
     % cmf testcs.o testfcm.o -vu -Llib_dir \
       -lcs_cm5_vu_sp
     ```

   - For CM-5s without vector units:

     ```
     % cmf testcs.o testfcm.o -sparc -Llib_dir \
       -lcs_cm5_sparc_sp
     ```

   - For Sun-4s:

     ```
     % cmf testcs.o testfcm.o -cmsim -Llib_dir \
       -lcs_cm5_cmsim
     ```

   where *lib_dir* is the library directory listed by the **-dirs** option to the **cs** command.

   If your programs are compiled with **-g**, **-cmdebug**, or **-cmprofile**, you must also specify one of the following before **-lcs_cm5_***xxx*:

   - **-lprism5** (when the C* program is compiled with **-sparc**)
   - **-lprism5dp** (when the C* program is compiled with **-vu**)
   - **-lprism_sim** (when the C* program is compiled with **-cmsim**)

   For example:

   ```
   % cmf testcs.o testfcm.o -vu -Llib_dir -lprism5dp \
     -lcs_cm5_vu_sp
   ```

The result is an executable load module that you can execute as you normally would.

3.7 Compiling CM Fortran Programs that Call C*

If you are using Version 2.1 Beta or later of the CM Fortran compiler, you must compile the CM Fortran program with the **-padding** option. Otherwise, compile the CM Fortran as you normally would.

3.8 Symbols

The **cs** command predefines these preprocessor symbols to 1 where appropriate in C* code:

unix	Any UNIX system
sun	Sun only
sparc	Sun-4 only
__CM5__	CM-5 only
__CM5_SPARC__	CM-5 with SPARC processors only
__CM5_VECUNIT__	CM-5 with vector units
__CSTAR__	This is a C* compiler
__STDC__	This is an ANSI C compiler

3.9 Using make

The UNIX **make** utility can make object (**.o**) files from C* **.cs** files, just as it does with **.c** files. The only requirement is that this code must appear somewhere in the makefile:

```
CS = cs
CSFLAGS = $(CFLAGS)
.SUFFIXES: .cs
.cs.o:
        $(CS) -c $(CSFLAGS) $<
```