CLAS-NOTE:95-006-a
September 19, 1995
A. Freyberger

Templates for CLAS

Abstract

A simple **perl** script has been written to provide standard templates for Fortran source code and include files, C code, c-shell scripts, born shell scripts, perl scripts and HTML files. Another **perl** script is also available to “insert” **rcs** variables into existing Fortran code.

1 Why templates?

Standardization of a code template is important for large software efforts for several reasons. Use of a common template for **all** code, makes all code look familiar and information about the code is easy to locate. With the use of the code management system, **rcs**, a template provides automatic declaration of **rcs** variables. These variables allow for identification of the source versions used to create the executable, via the **ident** command. The template also automatically declares character strings that are used in the standard message logging package.

Without the template and/or script to produce the template the originator of the source code would have to write code to declare these variables. This would yield a large amount of code with typographical errors and a large amount of code with no **rcs** variables declared. With this in mind two simple scripts have been written to:

- write a Fortran template file
- insert **rcs** variables into existing Fortran files.

These scripts are flexible and easy to use. Both of these scripts now reside in the official CLAS software domain and by including:
source /clas/cebafb/u1/etc/profile/.clascshrc
in your **.cshrc** file, you will be able to execute these scripts on the HP cluster here at CEBAF.

2 Fortran template script: *mktmpl*

The perl script **mktmpl** will create a complete Fortran source or include file in the directory that it is executed. For example typing
mktmpl dummy
will create a Fortran subroutine, *dummy*, in the file *dummy.F* in your directory. Look inside *dummy.F* and you will see some **rcs** variables, standard comment fields and message logging characters. The file also contains an example of how to use include files and the message logging routine **RECMES**. The **rcs** variables will be expanded when the code is inserted into **rcs**, this will keep track of modifications and code versions.

You can also type,
mktmpl dummy i
and the include file *dummy.inc* will be created in your present directory.

The complete command string is: **mktmpl name flags** where **name** is the routine name and **flags** is an optional string of flags. Type **mktmpl h** for a complete list of options. But the typical user will execute **mktmpl name** and **mktmpl name i** most of the time.

The list below shows the flags recognized by mktmpl and the output these flags create:

- f (default) Fortran subroutine template
- b Bourne shell script template
- c C-code template
- C c-shell script template

- i Fortran include file template
- p perl script template
- w HTML document format

3 Inserting rcs variables into existing Fortran code: *insert_rcs*

Suppose you are already in possession of Fortran source code and include files, and you would like to have this part of the CLAS software effort. This implies that you insert the **rcs** character strings into all your Fortran and include files, a tedious job. If **all** the code has a identifiable string or "keyword", for example **integer**, and the best place for the **rcs** variables is the line above the line that contains the keyword **integer**. Then the command

insert_rcs integer -1 .F

will insert the **rcs** variables one line above the *first* occurrence of a line that contains the keyword **integer** in **all** your *.F* files in that directory. Another likely "keyword" could be **IMPLICIT** and the command

insert_rcs IMPLICIT 2 name.F

would insert the **rcs** variables 2 lines *below* the first occurrence of **IMPLICIT** in the routine *name.F*. Note that the "keyword" is case sensitive.

The result of executing the **insert_rcs** script is a Fortran file with the same name as the original Fortran file. The original Fortran file can be found in, *name.F.old*.

The same script will insert a **rcs** character string into include files (files with *.inc* extensions), by executing:

insert_rcs keyword offset name.inc

or

insert_rcs keyword offset .inc

This should be done for all include files, so that we can identify in the executable the version the include file used to compile the code.

4 Getting these scripts to work on other machines

First you will need the scripting language *perl* on your home machines. Then create or find a suitable subdirectory for these scripts and copy all the files in:

/clas/cebafb/u1/cmobj/script/

into this subdirectory. You will need to define an environment variable **CMH_SCRIPT** to point to this subdirectory. Then you will need to make sure this subdirectory is in your path or add it to your path.

That should be all you need to do.

5 Hard copies of the scripts

Copies of the **perl** scripts, **mktmpl insert_rcs** are included for those who would like to see an example of the scripting language **perl**. These are just examples; perhaps bad examples. For those wishing to learn **perl**, the book *Learning Perl*, by Randal L. Swartz (published by O'Reilly & Associates, Inc.) is recommended.