Creating a custom modulefile

Introduction

NCCS uses the Environment Modules Project (http://modules.sourceforge.net/), better known as the `module` command, to manage multiple versions of programs and libraries. Users use it to manage their environment. Users are not restricted to only the system level modulefiles provided by NCCS. Users may create their own. One specific use of custom modulefiles is to manage various LIS-related environments; e.g., to manage compilers and environment variables for compiling and running LIS 6 vs LIS 7 or for compiling and running with the Intel compilers vs the GCC compilers. This post explains how to create a custom modulefile.

Please refer to the man pages for *module(1)* and *modulefile(4)* for more information.

Set up

First create a sub-directory named *privatemodules* in your home directory.

Then add this command to your *.profile* (or equivalent) login file.

```
module use --append $HOME/privatemodules
```

This will make your custom modulefiles available to the `module` program every time that you log on.

NOTE

Please note that on-line resources suggest using `module load use.own` to set up your custom modulefiles. This command will create, if necessary, a *privatemodules* sub-directory in your home directory along with providing a sample modulefile named *null*. Then this command will add your *privatemodules* sub-directory to the `$MODULEPATH` environment variable, making your custom modulefiles available to the `module` program.

However, that command does not work on all systems. On some systems, like *discover*, you have to run `module load $MODULESHOME/modulefiles/use.own`.

Also note that if you subsequently run the `module purge` command, then you must rerun the `module load use.own` command to make your custom modulefiles available again.

Thus, instead of using `module load use.own`, this post suggests that you add `module use --append $HOME/privatemodules` to your *.profile* file. This command neither relies on the system to find the *use.own* module nor is affected by a `module purge` command.

Creating a custom modulefile

Below is a sample custom modulefile. It loads the modules and sets the environment variables needed to set up an Intel-18-based development environment. To use it, copy the contents of the listing into *$HOME/privatemodules/intel_18_0_3_222*. Then run the command:

```
discover07$ module load intel_18_0_3_222
```

After loading the custom modulefile, a `module list` command will report something similar to:

```
discover07$ module list
Currently Loaded Modulefiles:
  1) comp/intel-18.0.3.222   5) other/svn-1.9.5
  2) mpi/impi-18.0.3.222     6) other/vim-8.0
  3) tool/tview-2017.1.21    7) intel_18_0_3_222
  4) other/git-2.9.3
```

Listing of sample custom modulefile intel_18_0_3_222

```
#%Module1.0#####################################################################

proc ModulesHelp { } {
    puts stderr "\t[module-info name] - loads the INTEL_18_0_3_222 env"
    puts stderr ""
    puts stderr "\tThe following env variables are set:"
    puts stderr "\t\tDEV_ENV"
    puts stderr ""
    puts stderr "\tThe following modules are loaded:"
    puts stderr "\t\tcomp/intel-18.0.3.222"
    puts stderr "\t\tmpi/impi-18.0.3.222"
    puts stderr "\t\ttool/tview-2017.1.21"
    puts stderr "\t\tother/git-2.9.3"
    puts stderr "\t\tother/svn-1.9.5"
    puts stderr "\t\tother/vim-8.0"
    puts stderr ""
    puts stderr "\tPython 2.7 is added to the search PATH."
}


conflict comp mpi



module-whatis   "loads the [module-info name] environment"



set modname     [module-info name]
set modmode     [module-info mode]



module load comp/intel-18.0.3.222
module load mpi/impi-18.0.3.222

module load tool/tview-2017.1.21
module load other/git-2.9.3
module load other/svn-1.9.5
module load other/vim-8.0



setenv   DEV_ENV         INTEL_18_0_3_222

prepend-path   PATH   /usr/local/other/SLES11.3/python/2.7.11/gcc-4.3.4/bin
```

Additional custom modulefiles

It is suggested that you keep your custom modulefiles as lean as possible. Do not load more modules than are needed for a particular task. The above sample custom modulefile loads only the modules needed to perform general software development with the Intel 18 compilers.

But Jim, I need Matlab, R, IDL, etc., what should I do? Create additional custom modulefiles targeting the other tools and tasks that you need. For example, create a custom modulefile for Matlab. It should load only the modules needed to run Matlab, and it should set only environment variables required and related to running Matlab.

WARNING When you load too many unrelated modules into your environment, you run the risk of creating undetected incompatibilites between tools and libraries. For example, you may end up loading three different NetCDF libraries. And one day LIS runs, but the next day LIS cannot read its own restart file.

When dealing with multiple tasks using multiple custom modulefiles, you may either load and change the custom modulefiles, or you may launch multiple `xterm` s (one for each task) on discover, and load your different custom modulefiles in each `xterm`.

Example of loading and changing custom modulefiles

```
discover07$ module load lis_7_intel_14_0_3_174_sp3

# Work, work, work

discover07$ module purge
discover07$ module load my_matlab_mod

# Plot, plot, plot
```

Sample *.profile* and *.bashrc* files

Please refer to the man page for *bash(1)* for more information regarding the use of the *.profile* and *.bashrc* files.

Sample .profile file

```
# This file is read each time a login shell is started.

#
# Set LD_LIBRARY_PATH
#
# non LDT_*, non LIS_*, non LVT_* paths go here
#export LD_LIBRARY_PATH=$ADDITIONAL_PATHS:$LD_LIBRARY_PATH


#
# Set environment
#
export EDITOR=vim
export SVN_EDITOR=vim
export TVDSVRLAUNCHCMD=ssh


#
# Set PATH
#
# non LDT_*, non LIS_*, non LVT_* paths go here
#export PATH=$ADDITIONAL_PATHS:$PATH


#
# Set modules
#
# load modules manually from the command line
# to keep your login environment clean
module use --append $HOME/privatemodules


ulimit -s unlimited
```

Sample .bashrc file

```
# This file is for interactive shells.
# On some systems, it is loaded in some non-interactive cases.
# Return if this is not an interactive shell.
[ "$PS1" ] || return


#
# Set aliases
#
unalias -a
alias ls="ls --color=auto"


#
# Set prompt
#
export PS1="\[\e[36m\]\h\[\e[00m\][\!]\$ "


#
# Set environment
#
export
LS_COLORS='no=00:fi=00:di=33:ln=36:pi=40;33:so=01;35:bd=40;33;01:cd=40;33;01:or=01;05;
37;41:mi=01;05;37;41:ex=32:*.cmd=32:*.exe=32:*.com=32:*.btm=32:*.bat=32:*.sh=32:*.csh=
32:*.tar=01;31:*.tgz=01;31:*.arj=01;31:*.taz=01;31:*.lzh=01;31:*.zip=01;31:*.z=01;31:*
.Z=01;31:*.gz=01;31:*.bz2=01;31:*.bz=01;31:*.tz=01;31:*.rpm=01;31:*.cpio=01;31:*.jpg=0
1;35:*.gif=01;35:*.bmp=01;35:*.xbm=01;35:*.xpm=01;35:*.png=01;35:*.tif=01;35'

export MANPATH=$MANPATH:/usr/slurm/share/man
```

LISF development environments

Custom modulefiles used to compile LISF are found in the *env* directory at the top of the source code.