

Shell and Compiler options on RCC machines

Shan-Ho Tsai

**Research Computing Center
University of Georgia**

<http://www.rcc.uga.edu>

08/28/2007

Computing Resources at RCC

- **IBM pcluster** – AIX 5.2
 - 32 Power4 8-way nodes (p655 series)
 - 16GB RAM per node
- **SGI altix** – 64-bit Suse Linux
 - 16 Intel Itanium2 1.3GHz processors (IA-64)
 - 32GB shared memory
- **Rackable rcluster** – 64-bit Red Hat Linux
 - 120 (2p) single core AMD Opterons 2.2GHz (IOB)
 - 24 (2p) dual core AMD Opterons 2.2GHz
 - 15 (2p) dual core AMD Opterons 2.2GHz (Statistics Dept.)
 - 2GB RAM per core
- Shared home directory (Netapp filer)

What is a UNIX SHELL?

A program that defines the environment of a terminal and interprets the commands typed at the prompt. The basic shells come in three main language forms:

- **Bourne Shell** (sh and extension bash): used by many OS scripts, compact syntax.
- **C Shell** (csh and extension tcsh): has similar syntax to C language.
- **Korn Shell** (ksh): a superset of sh, most functional shell, made famous by IBM's AIX.

Which SHELL am I using?

At the command prompt type:

```
pcluster$ echo $SHELL  
/bin/tcsh
```

To change the SHELL for current terminal:

```
pcluster$ bash
```

To change default SHELL:

```
pcluster$ chsh
```

Some differences between the shells

- Keyboard functionality (**bash** and **tcsh**)
 - Arrow up and down to recover previous commands.
 - Tab key to autocomplete file and directory names.

Possible but more tricky to set these up in **ksh** and **csch**.
- **history** command has time stamp (**tcsh**)
- Input and Output file redirection.
- Behavior of **nohup**

Startup “Dot Files”

Startup files are processed in the following order:

- **ksh**: /etc/profile then ~/.profile (can set latter to source ~/.kshrc)
- **bash**: /etc/profile, then the first of ~/.bash_profile, ~/.bash_login, or ~/.profile (can set one of these 3 to source ~/.bashrc). Also processes ~/.bash_logout when a login shell exits.
- **csh, tcsh**: /etc/csh.cshrc; /etc/csh.login (if a login shell); ~/.cshrc; ~/.login (if a login shell). Also processes ~/.logout (if present) when login shell exits.

User can create system specific dot files. E.g.

- ~/.profile.pcluster, ~/.profile.rcluster
- ~/.cshrc.pcluster, ~/.cshrc.rcluster

Sample contents of dot files

Define aliases:

ksh,bash

- `alias rm='rm -i'`
- `alias mv='mv -i'`
- `alias emacs='emacs -bg DarkSlateGray -fg Wheat -geometry 80x40 -fn 10x20'`

csh,tcsh

- `alias rm 'rm -i'`
- `alias mv 'mv -i'`
- `alias emacs 'emacs -bg DarkSlateGray -fg Wheat -geometry 80x40 -fn 10x20'`

To use GNU commands on pcluster:

- `alias ls='/usr/linux/bin/ls --color=tty'`
- `alias ls '/usr/linux/bin/ls --color=tty'`

Sample contents of dot files

Define path:

Examples:

1. On pcluster, to use GNU commands (like `ls`, `cp`, `du`, etc) instead of AIX versions of the same:

`ksh,bash:`

```
export PATH=/usr/linux/bin:${PATH}
```

`csch,tcsh:`

```
setenv PATH "/usr/linux/bin:${PATH}"
```

2. Add path to non-default version of apps.

```
export PATH=/usr/local/app-3.6/bin:${PATH}
```

```
setenv PATH "/usr/local/app-3.6/bin:${PATH}"
```

Sample contents of dot files

Define environment variables:

Example: Define a source code directory in `~/myenv`

ksh,bash:

```
export SRCDIR=${HOME}/MC/src
```

csh,tcsh:

```
setenv SRCDIR "${HOME}/MC/src"
```

To use this definition in non-interactive shell, source this file in `~/kshrc`, `~/bashrc`, or `~/cshrc` with

- `~/myenv`

```
source ~/myenv
```

Sample contents of dot files

Define prompt:

The default prompt on RCC machines for bash is

```
[username@hostname dir]$
```

Example:

```
[shtsai@rcluster MC]$ cd src
```

```
[shtsai@rcluster src]$
```

To include the full path in the prompt, add in `~/.profile`

```
export PS1="[ \"'whoami'\"@\"'hostname'':$PWD' \"$ "
```

Then prompt above becomes:

```
[shtsai@rcluster.rcc.uga.edu:/home/rccstaff/shtsai/MC/src]$
```

Input and Output Redirection - csh

- Redirect stdout to a file
`./a.out > out.txt`
- Redirect both stdout and stderr to a file
`./a.out >& outerr.txt`
- Redirect stdin from a file
`./a.out < input.txt`

No way to separately redirect stderr and stdout, you either redirect stdout to a file and let stderr go, or redirect both stderr and stdout to the same file.

Input and Output Redirection - bash

Uses file descriptors: fd0=stdin, fd1=stdout, fd2=stderr

- Redirect stdout to a file

```
./a.out > out.txt
```

- Redirect stderr to a file

```
./a.out 2 > err.txt
```

- Redirect both stdout and stderr to a file

```
./a.out > outerr.txt 2>&1
```

- Redirect stdout to one file and stderr to another

```
./a.out > out.txt 2> err.txt
```

- Redirect stdin from a file

```
./a.out < input.txt
```

Interactive jobs with NOHUP

bash

```
rcluster$ nohup ./a.out &  
nohup: appending output to 'nohup.out'
```

```
rcluster$ nohup time ./a.out &  
nohup: appending output to 'nohup.out'
```

(stdout, stderr and time appended to file nohup.out)

csh/tcsh

```
rcluster$ nohup ./a.out &  
(stdout and stderr printed on the screen)
```

```
rcluster$ nohup ./a.out >& nohup.out &  
(stdout and stderr goes to file nohup.out, use >> to append)
```

```
rcluster$ nohup time ./a.out >& nohup.out &  
(stdout, stderr of a.out goes to file nohup.out, time shown on screen)
```

SHELL Script

A collection of SHELL commands executed in sequence.

Example: script to submit jobs on pcluster

```
#!/bin/csh
cd ~ /MC/run1
poe ./a.out < $1 > $2
rm -f *.temp
```

```
ugsub T8-i8-t1-12h sub.sh in.txt out.txt
```

Example: script to automatically resubmit a job when it reaches the maximum runtime and is killed by the batch queue (pcluster)

```
#!/bin/ksh
# Script run.ksh
cd /home/tsai
export qname=T1-i1-t1-1h
trap "ugsub $qname /home/tsai/run.ksh;
      exit 0" 24
time /home/tsai/prog.exe
```

No equivalent trap signal with csh

Example: script to submit an MPI job on rcluster

```
#!/bin/csh
cd working_directory
echo $LSB_HOSTS
cat /dev/null > mlist.$$
foreach variable ($LSB_HOSTS)
    echo $variable >> mlist.$$
end
mpirun -np 4 -machinefile mlist.$$
                               ./a.out
rm -f mlist.$$

bsub -n 4 -q d4-24h -o out.%J -e err.%J ./sub.sh
```

Compiler options - pcluster

IBM compilers: xlf, xlf90, xlf95, xlc, cc, x1C

Debugging:

- **-g** : Produces symbolic debug information in the object files
- **-C** : Array bound check (Fortran)
- **-qcheck=bounds**: Array bound check (C/C++)
- **-qflttrap=<option>**: Traps runtime floating-point exceptions

Optimization:

- **-O** : low order optimization (same as -O2)
- **-O3** : more aggressive optimization
- **-qstrict**: used with -O3 to ensure semantics not changed
- **-qarch=pwr4** : generates binaries tuned for Power4 processor
- **-qtune=pwr4** : generates binaries tuned for Power4 processor

IBM compilers: xlf, xlf90, xlf95, xlc, cc, xlc

Other:

- **-c**: compiles only, does not link
- **-o**: specifies executable name (default a.out)
- **-bmaxdata:0x80000000** : For 32-bit addressing, allows use of 2GB RAM (default is 256MB)
- **-q64 -qwarn64**: uses 64-bit memory addressing (default 32-bit)
- **-qcpluscmt**: permits "//" to introduce a comment line that lasts until the end of the source line, as in C++. (C only)
- **-qsuffix=f=f90**: allows F90 code to have .f90 ending (default .f)
- **-qlanglvl=f77std -qfixed**: Allows using xlf90 for Fortran77.
- **-qfloat=nomaf**: Uses 64-bit arithmetics (default is 80-bit)

Compiler options - rcluster

PGI compilers: pgf77, pgf90, pgf95, pgcc, pgCC

Debugging:

- **-g** : Produces symbolic debug information in the object files
- **-C or -Mbounds** : Array bound check (on by default with F77)
- **-Mnobounds** : Turns off array bound check (C/C++ default)
- **-Mfptrap**: Traps certain runtime floating-point exceptions

Optimization:

- **-O** : Low order optimization (included by default when -g absent)
- **-O3** : More aggressive optimization (use with care)
- **-fastsse**: Chooses generally optimal flags for a processor that supports SSE instructions (such as Opteron). Type **pgf77 -fastsse -help** to see the equivalent switches (caution: may slow code down)

PGI compilers: pgf77, pgf90, pgf95, pgcc, pgCC

Other:

- **-c**: compiles only, does not link
- **-o**: specifies executable name (default a.out)
- **-Kieee**: strict IEEE compliance
- **-B**: permits "//" to introduce a comment line that lasts until the end of the source line, as in C++. (C only)
- **-tp amd64**: specify target processor as Opteron in 64-bit mode
- **-tp k8-32**: specify target processor as Opteron in 32-bit mode

Compiler options - altix

Intel compilers: ifort, icc

- **-g** : Produces symbolic debug information in the object files
- **-CB or -check bounds** : Runtime array bound check
- **-fpe<n>** : Specifies floating-point exception handling at runtime
- **-fpe1** : All floating-point exceptions are disabled
- **-ftz** : Flushes underflowing numbers to zero (**important**)
- **-c**: compiles only, does not link
- **-o**: specifies executable name (default a.out)
- **-O** : low order optimization (same as -O2)

Compiler options - help!

On pcluster:

```
pcluster$ xlf
```

```
pcluster$ xlc
```

On rcluster:

```
rcluster$ man pgf77
```

```
rcluster$ man pgCC
```

On altix:

```
altix$ man ifort
```

```
altix$ man icc
```

Questions or Problems:

<http://www.rcc.uga.edu/contact/support.html>