Managing Many Jobs:
A hands-on workshop

Dr. Ross Dickson, Computational Research Consultant

Overview

- Problem: Lots of jobs, lots of files.
  How can I do less typing?
  - tedious *and* error-prone!
- Solution: Shell programming
- Examples:
  1) Create input files from a template
  2) Submit multiple jobs
  3) Use an *array job*

Setup

- Log in to an ACEnet cluster
- Create and get in to a directory called **`demo`** for this workshop

```
$ mkdir demo; cd demo
```

➔ Get all files from /home/rdickson/demo/ManyJobs

```
$ cp /home/rdickson/demo/ManyJobs/* .
```

The 'for' loop

```
$ for n in 10 20 30
> do
>     echo $n
> done
```

- 'for', 'in', 'do' & 'done' required
- indentation optional
- 'n' is a *variable name*
- '$n' is its *value*

The 'for' loop: Exercise

```
$ for n in 10 20 30; do
> echo $n; done
```

- Create three subdirs run10, run20, run30

```
$ for n in 10 20 30
> do
>     mkdir run$n
> done
```

Longer lists

```
$ seq 4
```

- Creates sequences

➔ Exercise: Print 0 10 20 30 40

```
$ man seq
$ seq 0 10 40
```

➔ Exercise: Print 08 09 10 11 12

```
$ seq -w 8 12
```

Command Substitution

```
$ list=$(seq 4)
$ echo $list
```

- **`$(cmd)`** runs cmd and substitutes output. Same as **`` `cmd` ``** (backticks)

→ Exercise: Print
`run08 run09 run10 run11`

```
$ man seq
$ for n in $(seq -w 8 11); do
>     echo "run$n"
> done
```

Stream editor 'sed'

- Reads a file or stdin
- Edited (transformed) data to stdout
- Many subcmds, **`s/x/y/`** is substitution

```
$ cat hello
Hello there
$ sed "s/there/everyone/" hello
Hello everyone
$ cat hello
Hello there
```

- "**`>filename`**" directs output to a new file

Input from a template

```
$ sed "s/there/everyone/" hello
```

- Set a variable **x=99**
- From **template.in**, create input with **T=99** instead of **T=1**

```
$ x=99
$ sed "s/T=1/T=$x/" template.in
##################
# Temperature
T=99   ...
```

Inputs from a template

```
$ sed "s/there/everyone/" hello
```

- From **`template.in`** create 3 files called **`input.in`**, each with a different T value.
- Leave them in the subdirs created earlier

```
$ for x in 10 20 30; do
> sed "s/T=1/T=$x/" template.in \
>                   >run$x/input.in
> done
```

Submit multiple jobs

- '**`qsub file`**' submits a job to Grid Engine
- → Submit the job **`job.sh`** from each subdir **`run10 run20 run30`**

```
$ cd ~/demo
$ for t in 10 20 30; do
>   cd ~/demo/run$t
>   qsub ~/demo/job.sh
> done
```

Array job

```
#$ -t 10-30:10
sed "/T=1/T=$SGE_TASK_ID/" \
    template.in >input.$SGE_TASK_ID
./run.exe input.$SGE_TASK_ID
```

- **`-t` *`low-hi:step`***
  defines range for **`SGE_TASK_ID`**
- Grid Engine sets different **`SGE_TASK_ID`** for each *task* in *job*
- Must be positive integers --- no zero or negatives

Array job

```
#$ -cwd
#$ -j y
#$ -l h_rt=00:05:00,test=true
#$ -t 10-30:10
sed "/T=1/T=$SGE_TASK_ID/" \
    template.in >input.$SGE_TASK_ID
./run.exe input.$SGE_TASK_ID
```

- Submit **`array-job`** and observe
  - Where did the standard output go?

Multiple parameters

- How to handle irregular lists of params? e.g. (T=10,P=15; T=19,P=13; …) ?
- File of parameter pairs + 'read':

```
$ cat params
10 15
19 13
23 10
$ read t p <params
$ echo $t $p
10 15
```

Multiple parameters

- **`read`** only works if the file stays open
- Combine **`read`** with a **`while`** loop like this:

```
$ cat params | while read t p
> do
>     echo $t $p
> done
10 15
19 13
23 10
```

Multiple parameters

- Array variables:

```
$ tarray=(10 19 21)
$ parray=(15 13 9)
$ echo ${tarray[0]}, ${parray[0]}
10, 15
```

Limitations

- Arithmetic
  - integer arithmetic possible but ugly
  - no floating point arithmetic
- No multi-dimensional arrays
- No associative arrays
- Not fast
    - but who cares?

Alternatives

- Python
- Perl
- Tcl/Tk, Ruby, …

- Cannot use any directly for job script
- but trivially invoked
- All can read environment
- All provide system interface functions

Where next?

- Online manual, 'man'
  - **`man bash; man qsub; man ...`**
- Tutorials on the Web
  - Google 'unix shell script tutorial'
- Books
  - *The Unix Programming Environment* by Kernighan & Pike is the classic
  - ...but there are many others!
- ACEnet Wiki
  - http://www.ace-net.ca/
- ACEnet Tech Support email
  - support@ace-net.ca