Bash Scripting and Jobs

Research Computing
UNIVERSITY OF COLORADO **BOULDER**

Be Boulder.

Bash Scripting and Job Submission

- Daniel Trahan & Andrew Monaghan
- *Email: Daniel.Trahan@Colorado.edu, Andrew.Monaghan@Colorado.edu*

- *RC Homepage: https://www.colorado.edu/rc*

Sign in! http://tinyurl.com/curc-names

- Slides available for download at: https://github.com/ResearchComputing/CHANGE_2019

Adapted from presentations by RC members Andrew Monaghan, Aaron Holt and John Blaas: 1, 2, 3, 4.

Outline

- Bash Scripting Basics
- Jobs Basics
- Running Jobs
  - Batch Jobs
  - Interactive Jobs
- Example Jobs

Bash?

- A shell is the environment in which commands are interpreted in Linux
- GNU/Linux provides various shells; bash most popular
  - sh Bourne-again Shell (Bash)
  - csh C-Shell
  - tcsh Tc-Shell
  - ksh Korn-shell
- Shell scripts are files containing collections of commands for Linux systems that can be executed as programs

Bash Script

- To create a bash shell script file, the first line must be:

```
#!/bin/bash
```

- Program loader recognizes the `#!` as an interpreter directive. This is followed by `/bin/bash` which tells the OS which shell should be used.
- Example:

```
#!/bin/bash

cd /home/user

hostname
echo “Hello!” > file.out
```

Permissions

- Before you can run a script you need to make sure the script has the appropriate permissions
- At the command line, type, `ls -l`
- Column 1: Permissions
  - d, r, w, x
  - Owner, group, global
- Chmod changes permissions
  - `chmod a+x filename.sh`
  - Makes the file executable for everyone
- Run it, using `./filename.sh`
- Could also have done bash filename.sh and avoided permissions

Bash Variables

- Local variables are usable within the current shell
- Environment variables are usable to any child process of the shell
- Several pre-defined environment variables in your container
  - Type `env` in the terminal window
- Variables can be stored in arrays can be used too!
- Examples (Try these out!)

Local Variables

```
NAME=CUBOULDER
echo $NAME

ARRAY=(Hello World!)
echo ${ARRAY[0]}
```

Environment Variables

```
export ENVVAR=Persists!
echo $ENVVAR

/bin/bash
echo $ENVVAR
```

Bash Conditionals

- 3 varieties of conditionals exist: if then, elif then, else
- Comparative operators include `-lt -le -eq -gt -ge -ne`
- Syntax is a bit odd:

`if then, else` loop

```
if [[ $x -gt 10 ]]; then
    echo $x
else
    hostname
fi
```

`if then, elif then, else` loop

```
if [[ $x -gt 10 ]]; then
    echo $x
elif [[ $x -lt 4]]; then
    let x++
else
    hostname
fi
```

Bash Iteration

- Two standard loops: `for` and `while`:
  - For loops iterate over a set count
  - While loops iterate until a certain condition is met
- `for` syntax:

```
for i in {0..10}; do
    echo $i;
done;
```

```
abc = (a b c)
for i in ${abc[@]}; do
    echo $i;
done;
```

`while` syntax:

```
i=0
while [ $i -gt 10 ]; do
    echo $i;
    let i++;
done;
```

Example: File Creation

- Generate 6 files named 0, 1, 2, 3, 4, 5
- Utilize the touch command! Creates an empty file. Syntax:

```
touch <file-name>
```

- Answer:

```
for i in {0..5}; do
    touch $i
done
```

Jobs and Job Scripting

- Job – a system allotment of resources that run a particular application.
- Slurm resource manager
- Because Summit and Blanca are shared resource amongst a variety of groups on campus, Users must run their applications through jobs.
  - Ensures everyone can utilize the system
  - No one person taking up too much of the system!
- Blanca is a bit different, ICS owns a portion of the system that is solely shared between ICS researchers.
  - Substantially shorter wait times!

Job Types

- Batch Jobs
  - Non-interactive allotment of resources that run applications in the background
  - Think “baking a batch of cookies”
  - Useful for:
    - Applications that take a substantial amount of time
    - Non-interactive Applications
- Interactive Jobs
  - Interactive allotments or resources where user run their applications manually
  - Useful for:
    - Debugging Applications
    - Running GUI Applications

Submitting Batch Jobs

- `sbatch` command: submit a batch job to Slurm
- The sbatch command usually takes in 1 parameter: A job script.
  - Job scripts provide all information on what is needed for the job.
  - Parameters can be overwritten or added externally by specifying the parameter as a flag.
- Example:

```
sbatch test.sh
```

or

```
sbatch test.sh --time=02:00:00
```

Job Parameters

- Allocation: `--account=<account-name>`
- Partition: `--partition=<partition-name>`
- Number of nodes: `--nodes=<nodes>`
- Number of Tasks: `--ntasks=<number-of-tasks>`
- Quality of service: `--qos=<qos>`
- Reservation: `--reservation=<name>`
- Wall time: `--time=<wall-time>`
- Job Name: `--job-name=<jobname>`
- Output File: `--output=<outputname>`

More on slurm commands: https://slurm.schedmd.com/quickstart.html

FYI: You do NOT actually type <> above – this designates something specific you as a user must enter about your job

ICS QoS, Account, and Partition

- When using the ICS Blanca nodes, job scripts must specify the following information:

```
--account=blanca-ics
--partition=blanca-ics
--qos=blanca-ics
```

- Users of Blanca also have access to a low-priority preemptible QoS that will attempt to run your job on any available node on Blanca
  - Preemptable jobs will be booted off the node and restarted at a later time if a high priority user runs a job on the node.

```
--qos=preemptable
```

Submitting Your First Job!

Submit a slurm job with the following instructions:

1. The job should run the Linux “hostname” command
2. The job script will be named: `submit_hostname.sh`
3. The job will run on 1 core of 1 node
4. We will request a 1 minute wall time
5. Run from the blanca-ics QOS
6. Run on the blanca-ics partition
7. Use the ‘change’ reservation
   - *This is only for this workshop*

submit_hostname.sh

```
#!/bin/bash
#SBATCH --nodes=1                       # Number of requested nodes
#SBATCH --ntasks=1                      # Number of requested cores
#SBATCH --time=0:01:00                  # Max wall time
#SBATCH --qos=blanca-ics                # Specify QOS
#SBATCH --partition=blanca-ics          # Specify Summit Haswell nodes
#SBATCH --output=hostname_%j.out        # Rename standard output file
#SBATCH --job-name=hostname             # Job submission name
#SBATCH --reservation=change            # Reservation (workshop only)

# Written by:         Shelley Knuth, 15 July 2016
# Updated by:         Andy Monaghan, 8 March 2018
# Purpose:            Demonstrate how to run a batch job on RC resources

# purge all existing modules
module purge

hostname
```

Running the job script

Submit the job:

```
$ sbatch submit_hostname.sh
```

Check the status of the job:

```
$ squeue / $ squeue -u <user> /
$ squeue -q <qos>
```

…or

```
$ sacct / $ sacct --format=<options>
```

…or

```
$ scontrol show job <job number>
```

Look at the job output:

```
$ cat hostname_<job-id>.out
```

*(*note that <job-id> is your job number)*

[More on slurm commands: https://slurm.schedmd.com/quickstart.html](https://slurm.schedmd.com/quickstart.html)

Your turn

- Create a Slurm job script and submit it as a job, with the following instructions:

1. Name it 'submit_sleep.sh'
2. The job should run first the whoami command, then the Linux "sleep" command for 30 seconds, then the hostname command
   - Syntax for these Linux commands is:

```
echo "I am" `whoami`
echo "Running on host" `hostname`
echo "Starting Sleep"
sleep 30
echo "Ending Sleep. Exiting Job!"
```

Submit_sleep.sh

```
#!/bin/bash
#SBATCH --nodes=1                       # Number of requested nodes
#SBATCH --ntasks=1                      # Number of requested tasks
#SBATCH --time=00:01:00                 # Max wall time
#SBATCH --qos=blanca-ics                # Specify QOS
#SBATCH --partition=blanca-ics          # Specify Summit Haswell nodes
#SBATCH --output=sleep_%j.out           # Rename standard output file
#SBATCH --job-name=sleep                # Job submission name
#SBATCH --reservation=change            # Reservation (workshop only)

# purge all existing modules
module purge

echo "I am" `whoami`
echo "Running on host" `hostname`
echo "Starting Sleep"
sleep 30
echo "Ending Sleep. Exiting Job!"
```

Interactive jobs

- Sometimes we want our job to run in the background
- Sometimes we want to work on program in real time
    - Great for testing, debugging
- For example, let's run an interactive Python job…

Running an interactive job

- To work with R interactively, we request time from Summit
- When the resources become available the job starts

- Commands to run:

```
sinteractive –-time=00:10:00 --qos=blanca-ics
```

- Once we receive a prompt, then:

```
module load python
python
```

- Once we finish we must exit! (job will time out eventually)

Thanks!

- Please fill out the survey: http://tinyurl.com/curc-survey18
- Sign in! http://tinyurl.com/curc-names
- Contact information:

  rc-help@Colorado.edu

  Daniel.Trahan@Colorado.edu
- Slides and Examples from this course: https://github.com/ResearchComputing/CHANGE_2019
- Slurm Commands: https://slurm.schedmd.com/quickstart.html