Executing Commands on HPC

Manpreet S. Katari

BASH Scripts

- A script is simply a list of commands in a file.
- Once you execute the file, it executes all the commands in the file, one by one.
- Great for creating workflows
- Files often end with .sh

BASH environment

- Executing the command env provides all the environment variables

```
[msk8@log-3]$ env | grep msk8
VAST=/vast/msk8
ARCHIVE=/archive/msk8
USER=msk8
HOME=/home/msk8
SINGULARITY_CACHEDIR=/state/partition1/msk8-singularity-cache
SCRATCH=/scratch/msk8
MAIL=/var/spool/mail/msk8
LOGNAME=msk8
```

- You can always evaluate them to see their value by using the $ in front.

```
[msk8@log-3]$ echo $USER
msk8
```

Simple Text Editor (nano)

- Executing the command nano starts a simple text editor. The legend at the bottom shows how to perform special commands. The ^ represents the CTRL key. For example CTRL-X will exit nano.

```
^G Get Help        ^O Write Out
^W Where Is        ^K Cut Text
^J Justify         ^C Cur Pos
M-U Undo            ^X Exit
^R Read File        ^\ Replace
^U Uncut Text      ^T To Spell
^_ Go To Line      M-E Redo
```

Bash Script

Let’s create a file called myname.sh which simply echoes the USER variable.

```
[msk8@log-3]$ nano myname.sh
```

The first line of the file is often the path of the command needed to execute the commands in the file preceded by #! For example, if the file contains python code, the first line should be `#!/path/to/python`

```
#!/bin/bash

echo $USER
```

Executing the script

To execute the file, we must first give it executable privilege and then simply call the name of the file.

```
 [msk8@log-3]$ chmod 755 myname.sh
 [msk8@log-3]$ ./myname.sh
msk8
```

Alternatively you can simply call the application that is needed to execute the commands.

```
 [msk8@log-3]$ bash myname.sh
msk8
```

SLURM

If everyone executing their commands on the computer/node where they logged in, it would crash the computer.
SLURM is a job/queue manager that finds a computer that is available and has the resources you requested and executes the job there.
Let's edit the myname.sh script and add a line that tells us which computer the job runs on.

```
#!/bin/bash

echo $USER
echo $HOSTNAME
```

Executing the script

To execute the file, we must first give it executable privilege and then simply call the name of the file.

```
[msk8@log-3]$ ./myname.sh
msk8
log-3.hpc.nyu.edu
```

Now let's send this to another computer.

```
[msk8@log-3]$ sbatch myname.sh
Submitted batch job 14015883
```

Executing the script

You can check the status of your jobs by using the command squeue. Here I specify the username so i don't see everyone's jobs.

```
[msk8@log-3 ~]$ squeue -u msk8
            JOBID PARTITION     NAME     USER ST       TIME  N
         14015883 cs,cpu_gp myname.s     msk8 PD       0:00
```

Since the script executed on a different computer, I won't see the results printed on the screen. They get saved to a file.

```
[msk8@log-3 ~]$ cat slurm-14015883.out
msk8
cs039
```

SBATCH options

The best way to specify the resources that your script needs to run are by adding them to the script.

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1
#SBATCH --time=5:00:00
#SBATCH --mem=2GB
#SBATCH --job-name=myTest
#SBATCH --mail-type=END
#SBATCH --mail-user=bob.smith@nyu.edu
#SBATCH --output=slurm_%j.out
```

SBATCH Interactive mode

If you prefer to run your commands interactively instead of a script, you can request a bash shell on an available compute node

```
[msk8@log-3 ~]$  srun --cpus-per-task=4 --time=2:00:00 --mem=4000 --pty /bin/bash
srun: job 14016003 queued and waiting for resources
srun: job 14016003 has been allocated resources
[msk8@gr001 ~]$
```

Softwares and Environments

To help prevent conflicts in different versions of software dependencies, HPC uses module environment.

module unload <module-name> : unload a module

module purge : remove all loaded modules from your environment

module load <module-name> : load a module

module whatis <module-name>: Find out more about a software package

module list: check which modules are currently loaded in your environment

module avail: check what software packages are available

Fastqc

Fastqc is a popular software that performs quality control on fastq file. Let's run it on a sample that I used many years ago. First copy the file to your home directory.

```
[msk8@gr001 ~]$ cp /scratch/work/courses/HITS-2016/rnaseq/KCL_1.fastq .
```

make sure you are on a compute node when doing the next steps

```
[msk8@gr001 ~]$ which fastqc
/usr/bin/which: no fastqc in
(/home/msk8/.local/bin:/home/msk8/bin:/share/ap
[msk8@gr001 ~]$ module avail fastqc

---------------------------------------------
/share/apps/modulefiles
----------------------------------------------
   fastqc/0.11.9
[msk8@gr001 ~]$ module load fastqc/0.11.9
[msk8@gr001 ~]$ which fastqc
/share/apps/fastqc/0.11.9/bin/fastqc
[msk8@gr001 ~]$ fastqc KCL_1.fastq
```

Let’s put this in a script and use sbatch

fastqc.sh

```
#!/bin/bash
#SBATCH --cpus-per-task=1
#SBATCH --time=5:00:00
#SBATCH --mem=2GB
#SBATCH --job-name=fastqc
#SBATCH --output=fastqc_%j.out

# You can add comments in your code by starting with a hash

module purge # remove all module already loaded
module load fastqc/0.11.9  # load the module

fastqc KCL_1.fastq
```

```
[msk8@gr001 ~]$ sbatch fastqc.sh
Submitted batch job 14016174
```

Results

```
[msk8@gr001 ~]$ cat fastqc_14016174.out
Started analysis of KCL_1.fastq
Approx 5% complete for KCL_1.fastq
Approx 10% complete for KCL_1.fastq
Approx 15% complete for KCL_1.fastq
Approx 20% complete for KCL_1.fastq
Approx 25% complete for KCL_1.fastq
Approx 30% complete for KCL_1.fastq
Approx 35% complete for KCL_1.fastq
```

```
[msk8@gr001 ~]$ ls -l KCL_*
-rw-r--r--. 1 msk8 msk8 1086962248 Jan 19 01:04 KCL_1.fastq
-rw-rw-r--. 1 msk8 msk8     257243 Jan 19 01:18 KCL_1_fastqc.html
-rw-rw-r--. 1 msk8 msk8     342202 Jan 19 01:18 KCL_1_fastqc.zip
```

A script with arguments

fastqc2.sh

```
#!/bin/bash
#SBATCH --cpus-per-task=1
#SBATCH --time=5:00:00
#SBATCH --mem=2GB
#SBATCH --job-name=fastqc
#SBATCH --output=fastqc_%j.out

# The variable name should be flushed to the left of the line
# $1 means first argument, $2 would be the second.
FASTQ=$1

echo "Analyzing $FASTQ"

module purge # remove all module already loaded
module load fastqc/0.11.9  # load the module

fastqc $FASTQ
```

```
[msk8@gr001 ~]$ sbatch fastqc2.sh KCL_1.fastq
Submitted batch job 14016180
```

Results

```
[msk8@gr001 ~]$ cat fastqc_14016180.out
Analyzing KCL_1.fastq
Started analysis of KCL_1.fastq
Approx 5% complete for KCL_1.fastq
Approx 10% complete for KCL_1.fastq
Approx 15% complete for KCL_1.fastq
Approx 20% complete for KCL_1.fastq
Approx 25% complete for KCL_1.fastq
```

HPC tutorial site

https://sites.google.com/nyu.edu/nyu-hpc/training-support/tutorials?authuser=0