Instructions to BASH Shell Scripts

Jingjing Yang

Department of Human Genetics

BASH Shell Script

- Why use BASH script?
  - Wrap Linux commands and tools together
  - Write a pipeline
  - Submit jobs
- Create a BASH script:
  - Use a text editor such as vi to create a text file containing Linux commands
  - First line contains the magic “shbang” sequence: #!/bin/bash
  - Comments start with “#” except for the first “shbang” line
  - Use “\” at the end of a line to break one command into multiple lines
  - Make the script executable: chmod 755
    - **7** is the combination of permissions **4**+**2**+**1** (read, write, and execute), **5** is **4**+**0**+**1**(read, no write, and execute)
    - Order of permission for: user, group, others
- Run a BASH script: ./example_bash.sh or bash -x example_bash.sh

BASH Shell Script

- Cons: have very little built-in math (consider using other Tools: R, Python)
- Back quotes ` ` and $( ) mean executing the command inside the quotes or parenthesis first and then assign the output as values for the variable on the left-hand-side
  - some_variable=`some Unix command`
  - some_variable=$(some Unix command)
- Each source code line is printed prior to its execution when specify option -x
  - Either in the header (first line, i.e., shebang, in the script): #!/bin/bash -x
  - Or on the command line: bash -x example_bash.sh

Common Syntax in BASH Script

- if/else (Here [ ] is part of the command, and **the space is important around [ ]**)

```
if [ condition ] ; then
        commands
fi

if [ condition ] ; then
        commands
else
        commands
fi

if [ condition ] ; then
        commands
elif [ condition ] ; then
        commands
fi
```

Logic Syntax

- Numeric comparison: -eq, -ne, -gt, -ge, -lt, -le
- String comparison: =, !=, <, >, -z, -n
- Directory exist: if [–d $dir] ; then …
- File exist: if [-f $myfile]; then …
- File exist and nonempty: if [-s $myfile] ; then …
- Executable file: if [-x $myfile]; then …
- || and && operands inside if [condition] (i.e. between round parentheses) are logical operands (or/and)
- || and && operands outside if [condition] mean then/else

  ```
  ( [ $a -eq 1 ] || [ $b -eq 2 ] ) && echo "ok" || echo "nok"
  ```
- Practically the statement says: if (a=1 or b=2) then; print "ok”; else; print "nok”;

Logic Syntax

- Loop:

```
for var in bash-list ; do
    commands
done

while [ condition ] ; do
    commands
done
```

AWK: Useful Tool in BASH

- The word awk is derived from the names of its inventors!!!
- awk is actually Aho Weinberger and Kernighan.
- From the original awk paper published by Bell Labs, awk is
  - “ Awk is a programming language designed to make many common information retrieval and text manipulation tasks easy to state and to perform.”
- Simply put, awk is a programming language designed to search for, match patterns, and perform actions on files.

AWK: Useful Tool in BASH for handling text files

awk options program file

- Options:
  - To specify a file separator: -F fs
  - To declare a variable: -v var=value
- Program:
  - To define an awk script, use braces surrounded by single quotation marks like this:

    awk ’{print "Welcome to awk command tutorial "}’
  - pattern { action }
    - awk –F”\t” ‘NR==1{print $0}’ file
    - BEGIN {…} pattern {…} pattern{…}END{…}
    - Commands in {…} are separated by semicolons “;”

AWK

- Built-in Variables: $0, $1, NR, FNR, NF
- Built-in Math Functions: sin(x), cos(x), sqrt(x), exp(x), log(x)
- C operators like: ++, --, +=, -=
- More information:
  - https://likegeeks.com/awk-command/
  - https://www.ibm.com/developerworks/library/l-awk1/

Example 1: run FastQC on a single file

- Step 1: Create a folder to hold all files related to the task/project
  - Recommended folder structure
    - ${HOME}/project
    - ${HOME}/project/scripts
    - ${HOME}/project/data
    - ${HOME}/project/refs
    - ${HOME}/project/logs
    - ${HOME}/project/output
- Step 2: Create the job submission script in ${HOME}/project/scripts
  - Recommend to create one script per step, e.g. FastQC, mapping, calling variants, etc.
  - Give a descriptive name to your scripts e.g. step01_fastqc.sh

Example bash script: run FastQC on a single file

```
1.   #!/bin/sh

2.   # This script requires a single parameter when
3.   # called – the portion of the file name
4.   # preceding .fastq.gz or .bam. This is usually
5.   # the <sample_name>
6.   #
7.   # The output directory (OUTDIR) needs to exist

8.   module load FastQC

9.   PRJDIR=“${HOME}/project”
10.  DATADIR=“${PRJDIR}/data”
11.  OUTDIR=“${PRJDIR}/output/FastQC”

12.  if [ -e /bin/mktemp ]; then
13.         TMPDIR=`/bin/mktemp –d /scratch/XXXXXX`
14.  elif [ -e /usr/bin/mktemp ]; then
15.         TMPDIR=`/usr/bin/mktemp –d /scratch/XXXXXX`
16.  else
17.         echo “Error. Cannot find program to create tmp directory”
18.         exit
19.  fi
```

```
18.  cp ${DATADIR}/$1.fastq.gz ${TMPDIR}

19.  fastqc –o ${TMPDIR} --no-extract ${TMPDIR}/$1.fastq.gz

20.  /bin/rm ${TMPDIR}/$1.fastq.gz

21.  rsync –av ${TMPDIR}/ ${OUTDIR}/$1

22.  /bin/rm –fr ${TMPDIR}

23.  module unload FastQC
```

Line numbers are not part of the script!

- Line 8 load the FastQC module
- Lines 9-11 defines some variables to use in the script
- Lines 12-19 create the unique folder in /scratch
- Line 18 copies data to the unique folder
- Line 19 runs the fastqc program
- Line 20 deletes the data copied in line 18
- Line 21 copies results back to the project folder
- Line 22 removes the unique scratch folder
- Line 23 unload the FastQC module