Introduction to Linux with Advanced Hands-on Practice

Xin Yang
Course Outline

1:30 - 1:45 Introduction, Accessing the system, Basic Shell Scripting
1:45 - 2:10 Hands-on Session 1

2:10 - 2:25 Basic Bash Scripting Constructs
2:25 - 2:55 Hands-on session 2

2:55 - 3:05 Break

3:05 - 3:20 Sed, Awk, pipes, redirects
3:20 - 4:00 Hands-on session 3
Directives used in this Lecture

Commands to type in will use the following:

- **Bold** words should be entered explicitly
- *Italicized* words are variable depending on the information that the utility needs
- commands for you to type in
- command output in
Accessing the system

- **HPRC Portal:**
  - [https://portal.hprc.tamu.edu/](https://portal.hprc.tamu.edu/)
  - login with your HPRC account

- **SSH (secure shell):**
  - Encrypted communication
  - Windows:
    - [https://hprc.tamu.edu/wiki/HPRC:MobaXterm](https://hprc.tamu.edu/wiki/HPRC:MobaXterm)
  - MacOS:
Select “Grace OnDemand Portal”
Using the **Portal**

OnDemand provides an integrated, single access point for all of your HPC resources.

**Message of the Day**

- **Files** > copy and edit files on the cluster’s filesystems
- **Jobs** > submit and monitor cluster jobs
- **Clusters** > open a shell terminal (command line) on a login node
- **Interactive Apps** > start graphical software on a compute node
- **Dashboard** > view file quotas and computing account allocations
Basic Shell Scripting

A shell script is a text file that contains one or more Linux commands that can be run as a single batch of commands.

Ideal for automating tasks.

It is good practice to name shell scripts with: .sh
Hands-on Session 1

- logon to the portal
- navigate to the file menu
- turn on hidden files
- open a terminal
Hands-on Session 1

Create a bash script using the portal text editor or your favorite text editor.

Name it my_script.sh

make your bash script executable

```bash
chmod u+x my_script.sh
```

run your bash script

```bash
./my_script.sh name
```

```
#!/bin/bash

# HPRC shell script exercise

my_name=$1

input="

You have written a simple shell script.

"

echo "Howdy $my_name" > output.txt

echo "$input" >> output.txt

mkdir script_output

mv output.txt script_output

cat script_output/output.txt
```
Basic Constructs for Bash Scripting

**Conditionals:** If something is true do something and if it is false, do something else

```bash
#!/bin/bash
#
i=1
if [ $i -eq 1 ] ; then
    echo i is equal to 1
else
    echo i does not equal 1
    echo i equals $i
fi
```
Basic Constructs for Bash Scripting

**Case Constructs**

case var in
case1)  
  <commands>
  ;;
  case2)  
  <commands>
  ;;
esac

#!/bin/bash
#
month='June'
case $month in
  Jan)
    mnum='01'
    ;;
  Feb)
    mnum='02'
    ;;
  ...
  Dec)
    mnum = '12'
    ;;
esac
**Basic Constructs for Bash Scripting**

**Loops:** Do something over and over until a specific condition changes and then stop

```
#!/bin/bash
#
i=1
while [ $i -le 100 ] ; do
    echo i equals $i
    ((i++))
done
```

```
for var in <list> ; do
    <commands>
done
```

```
for file in *.log ; do
    head -n1 $file
done
```
Hands-on Session 2

1. Create a shell script that checks if a variable is set to 1. Print to stdout if it is 1. If is not 1, print that it is not 1 and it's actual value. (hint: if then construct, echo)

2. Create a shell script that checks a variable named colors for the values red, green, and blue and echoes its color or ‘not primary’ if it is not red, green, or blue (hint: case, echo)

3. Create a shell script with a loop that echoes the value of a variable i from 1 to 10 (hint: while loop)
GNU `sed` - Stream editor

- **Useful one-liner scripts for `sed`**
  - common uses:
    - `sed 's/pattern1/pattern2/g' filename`
      - output is set to stdout
    - `sed 's/pattern1/pattern2/g' filename > filename2`
      - output is set to `filename2`
    - `sed -i 's/pattern1/pattern2/g' filename`
      - Modifies the file in-place. Changes the original file.

- **man `sed`**
  - Manual page for `sed` on Linux systems
GNU Awk

awk is used to search files for lines (or other units of text) that contain certain patterns and then do something (print, manipulate, etc).

- Delimiters
  - Default is white space
- Search patterns
  - awk '/pattern/' filename
- Variables
  - fields are stored in variables based on the delimiter
  - $0 the entire line
  - $1 1st field
  - $2 2nd field
- Print statement
  - awk '/pattern/ {print $0}' filename
  - awk '/pattern/ {print "$1","$2"}' filename>outputfilename.txt
- printf Statement for more control over the print format
- BEGIN/END
  - perform a task at the beginning or end
  - BEGIN {print '============='}
  - END {print '-------------'}
Pipes

- Pipes take the output of one command and sends it to another
  - `ls | more`
  - `ls | less`
    - List the files one page at a time
  - `grep Energy run1.out | grep HF`
  - `grep Energy run1.out | grep HF > HF_output.txt`
    - Searches a file named run1.out for the word Energy and then searches for the word HF in the lines that have the word Energy. The resulting information is then sent to a file named HF_output.txt
Redirecting Output

- Redirects output
  - `command>outputfilename`
  - `ls -al>list-of-files.txt`
  - `>` symbol appends to the end of the file instead of overwriting it.
  - `ls -al>>list-of-files.txt`
Exercise 3

1. Using the portal text editor or your favorite editor, create a file that contains the sample data from a mail-list: 
   https://www.gnu.org/software/qawk/manual/qawk.html#Sample-Data-Files
2. Use awk to print column 3 only: email addresses
3. Use awk to print the email addresses on one line separated by commas
4. Use sed to replace the @ with a space and send the result to awk using a pipe ( | ) and print the name and username of the email address only. (ie Amilia amelia.zodiacusque)
5. Bonus: send the results to a file instead of stdout (hint: >)
Need Help? Contact the HPRC Helpdesk

Website:  hprc.tamu.edu
Email:  help@hprc.tamu.edu
Phone:  (979) 845-0219

Help us, help you -- we need more info

• Which Cluster (Terra, Grace)
• NetID (NOT your UIN)
• Job id(s) if any
• Location of your jobfile, input/output files
• Application used if any
• Module(s) loaded if any
• Error messages
• Steps you have taken, so we can reproduce the problem