IBS574 – P45 Grace Crum Rollins Building

Linux shell & shell scripting - II

Ashok Dinasarapu Ph.D
Zwick Group, Dept. of Human Genetics
Emory University, Atlanta

Shell script

- A script is a collection of commands stored in a file.
- Shell is a command-line interpreter.
- To execute a shell script, ex. “hello.sh”

  Usage:

```
./hello.sh
. ./hello.sh
/home/user_name/script/hello.sh
bash hello.sh (or sh hello.sh)
```

Easiest way to do this is …

Lets start our second lab!

Login at Terminal

```
ssh user_name@blnx1.emory.edu
```

```
user_name@blnx1:~$
```

~ means your home dir, /home/user_name

user_name = your user name

SSH allows you to connect to your server securely and perform Linux command-line operations.

“mkdir”

- Create directories from your home directory (i.e /home/user_name)

Usage:
```
mkdir -p project/{data, script, out}
```

project/data

project/script

project/out

Usage:
```
cd project/script
```

Create/Edit text files

Choose a text editor: emacs, Vim

Usage: `vi hello.sh`

INSERT mode:
press keys like `i` OR `a` & start typing.

"i" will let you insert text just before the cursor.
"I" inserts text at the beginning of the current line.
"a" will let you insert text just after the cursor, and
"A" will let you type at the end of the current line.

Create/Edit text files

Type the following text:

```
#!/bin/sh
# My first script
echo "Hello World!"
```

“**#!/bin/sh**” a special clue given to the shell indicating what program is used to interpret the script.

Create/Edit text files

Type the following text:

```
#!/bin/sh
# My first script
echo "Hello World!"
```

```
#!/bin/bash
#!/usr/bin/perl
#!/usr/bin/Rscript
#!/usr/bin/env python
```

"**#!/bin/sh**" a special clue given to the shell indicating what program is used to interpret the script.

Create/Edit text files

SAVE mode:

press `esc` key AND

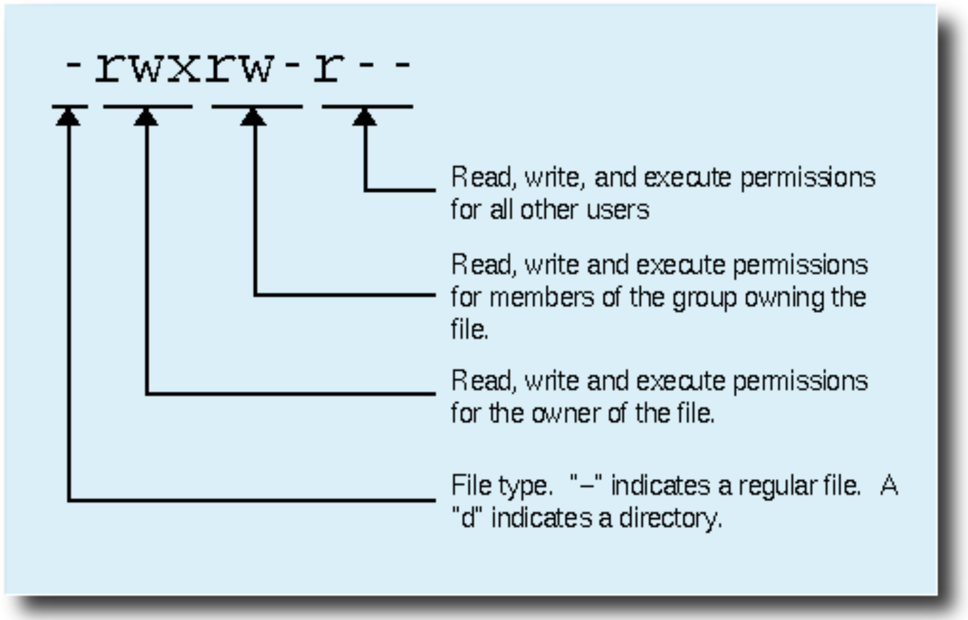
`:q!` for not to save OR

`:x` to save all typed content.

"chmod"

- Change the permissions of files
  - **R**ead (r), **w**rite (w), and e**x**ecute (x)
  - 3 types of users (user, group & other)

Usage: ls -l



-rwxrw-r--
Read, write, and execute permissions for all other users
Read, write and execute permissions for members of the group owning the file.
Read, write and execute permissions for the owner of the file.
File type. "–" indicates a regular file. A "d" indicates a directory.

“make file executable”

- **To make it executable**

  Usage: chmod +**rwx**r-x**r-x** hello.sh

  Usage: `chmod +x hello.sh`

  Usage: `sh hello.sh`

- **To make it un-executable**

  Usage: `chmod -x hello.sh`

“make file executable”

- **To make it executable** +755

  Usage: chmod +**rwx**r-x**r-x** hello.sh

  Usage: `chmod +x hello.sh`

  Usage: `sh hello.sh`

- **To make it un-executable**

  Usage: `chmod -x hello.sh`

Variables in Linux/Shell

Usage: `vi hello.sh`

```
#!/bin/sh
# My first script
i=22
echo “Hello, I am “ $i
# echo “Hello, I am  $i”
```

Usage: `sh hello.sh`

Variables in Linux/Shell

Usage: `vi hello.sh`

```
#!/bin/sh
# My first script
i=22
j="Hello, I am ”
echo $j $i
```

Usage: `sh hello.sh`

Conditions in Linux/Shell

Usage: `vi hello.sh`

```
#!/bin/sh
# My first script

PASS="test1234"

if [ $PASS == "test1234" ]; then
    echo "Correct pass word!!"
fi
```

Usage: `sh hello.sh`

Remember that the spacing is very important in the if statement.

Conditions in Linux/Shell

Usage: `vi hello.sh`

```
#!/bin/sh
# My first script

PASS="test123"

if [ $PASS=="test1234" ]; then
    echo "Correct pass word!!"
else
    echo "enter correct pass word!!"
fi
```

Usage: `sh hello.sh`

String Comparison Operators

Operator	Description	Example
= or ==	Is Equal To	if ["$1" == "$2"]
!=	Is Not Equal To	if ["$1" != "$2"]
>	Is Greater Than (ASCII comparison)	if ["$1" > "$2"]
>=	Is Greater Than Or Equal To	if ["$1" >= "$2"]
<	Is Less Than	if ["$1" < "$2"]
<=	Is Less Than Or Equal To	if ["$1" <= "$2"]
-n	Is Not Null	if [-n "$1"]
-z	Is Null (Zero Length String)	if [-z "$1"]

Integer Comparison Operators

Operator	Description	Example
-eq	Is Equal To	if [$1 -eq 200]
-ne	Is Not Equal To	if [$1 -ne 1]
-gt	Is Greater Than	if [$1 -gt 15]
-ge	Is Greater Than Or Equal To	if [$1 -ge 10]
-lt	Is Less Than	if [$1 -lt 5]
-le	Is Less Than Or Equal To	if [$1 -le 0]
==	Is Equal To	if (($1 == $2))
!=	Is Not Equal To	if (($1 != $2))
<	Is Less Than	if (($1 < $2))
<=	Is Less Than Or Equal To	if (($1 <= $2))
>	Is Greater Than	if (($1 > $2))
>=	Is Greater Than Or Equal To	if (($1 >= $2))

"for" loop in Linux/Shell

Usage: `vi hello.sh`

```
#!/bin/bash
# My first script
for i in {1..10}
do
    echo $i
done
```

Usage: `bash hello.sh`

“for” loop in Linux/Shell

Usage: `vi hello.sh`

```
#!/bin/bash
arr=(‘A’ ‘B’ ‘C’ ‘D’ ‘E’)
for i in {0..4}
do
    echo $i
    echo ${arr[$i]}
done
```

Usage: `./hello.sh`

"**Array**" in Linux/Shell

Usage: `vi hello.sh`

```
#!/bin/bash
arr=('A' 'B' 'C' 'D' 'E')
for i in {0..4}
do
    echo $i
    echo ${arr[$i]}
done
```

chmod +x hello.sh

Usage: `./hello.sh`

“**for**” **loop** in Linux/Shell

Usage: `vi hello.sh`

```
#!/bin/sh
# My first script
for i in 1 2 3 x y z
do
    echo $i
done
```

loop indices doesn't have to be just numbers

{1..3} x y z

Use bash

Usage: `sh hello.sh`

“while” loop in Linux/Shell

Usage: vi hello.sh

```
#!/bin/sh
# My first script

i=0
while [ $i -le 5 ]; do
    # echo “before $i”
    i=$(( $i+1 ))
    echo “after $i”
done
```

Usage: sh hello.sh

Functions in Linux/Shell

Usage: `vi hello.sh`

```
#!/bin/sh
# function definition
add_a_user()
{
	USER=$1
	PASS=$2
	echo “Passwd $PASS created for $USER on $(date)”
}
# function call
echo $(add_a_user bob letmein)
```

Usage: `sh hello.sh`

Functions in Linux/Shell

Usage: `vi hello.sh`

```
#!/bin/sh
# function definition
add_a_user()
{
      #USER=$1
      # PASS=$2
      echo “Passwd $1 created for $2 on $3”
}
# function call
echo $(add_a_user bob letmein “$(date)”)
```

Usage: `sh hello.sh`

download a fastq file

Usage: `cd project/data`

wget https://github.com/CGATOxford/UMI-tools/releases/download/v0.2.3/example.fastq.gz

- **View**

  Usage: `zcat example.fastq.gz|head`
- **Check file size**

  Usage: `ls -lh example.fastq.gz`
- **Count number of sequences in a fastq file**

  Usage: `grep -c "^>" example.fastq.gz`

“Calculate the length of reads”

Create the following file at **project/script**

Usage:
```
vi fastq.sh
```

```
#!/bin/sh
# using awk

zcat ../data/example.fastq.gz | \
awk '{if(NR%4==2) print length($1)}’ ../out/length.txt

# Rscript /home/user_name/script/hist.R
```

Usage:
```
sh fastq.sh
```

“Plot - Histogram”

Create the following file at **project/script**

Usage: `vi hist.R`

```
t.dat <- read.table('/home/user_name/out/length.txt')
jpeg('/home/user_name/out/rplot.jpg')
hist(t.dat[,1])
dev.off()
```

Usage: `sh fastq.sh`

“Plot - Histogram”

Create the following file at **project/script**

Usage:
```
vi hist.R
```

```
#!/usr/bin/Rscript
t.dat <- read.table('/home/user_name/out/length.txt’)
jpeg('/home/user_name/out/rplot.jpg’)
hist(t.dat[,1])
dev.off()
```

Usage:
```
chmod +x hist.R
```

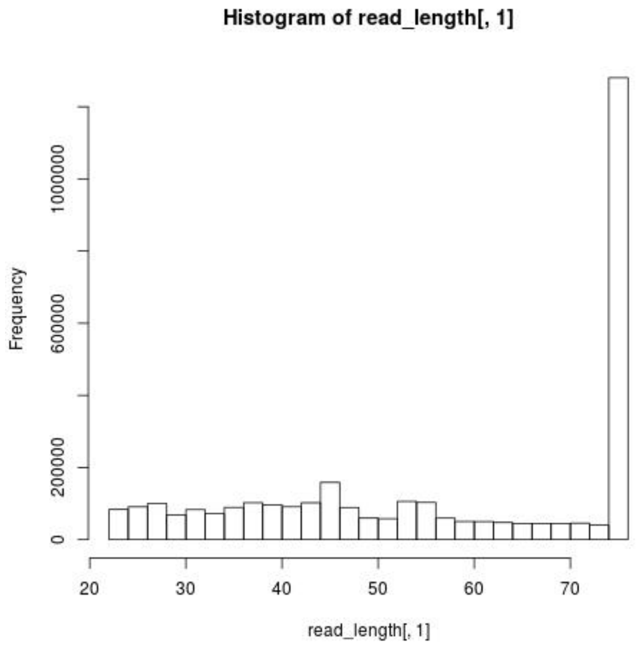
Usage:
```
./hist.R
```

Open/View Image

- Login as interactive mode

Usage:
```
ssh -Y user_name@blnx1.emory.edu
```

Usage:
```
xdg-open rplot.jpg
```



Practice Makes Perfect

Thank you