Introduction to BASH: Part II

By Michael Stobb

University of California, Merced

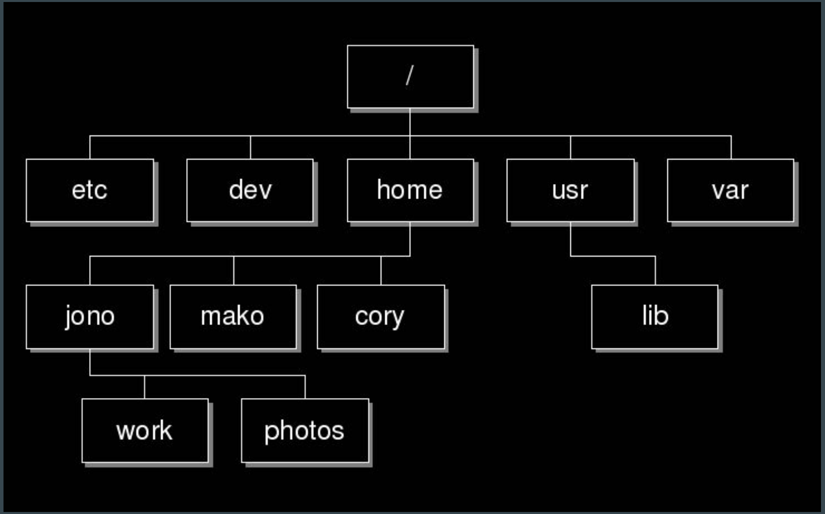
February 17th, 2017

Quick Review

- Linux is a very popular operating system for scientific computing
- The command line interface (CLI) is ubiquitous and efficient
- A “shell” is a program that interprets and executes a user's commands
  - BASH: Bourne Again SHell (by far the most popular)
  - CSH: C SHell
  - ZSH: Z SHell
- Does everyone have access to a shell?

Quick Review: Basic Commands

- pwd
  - ‘**p**rint **w**orking **d**irectory’, or where are you currently
- cd
  - ‘**c**hange **d**irectory’ in the filesystem, or where you want to go
- ls
  - ‘**l**i**s**t’ the contents of the directory, or look at what is inside
- mkdir
  - ‘**m**a**k**e **dir**ectory’, or make a new folder
- cp
  - ‘**c**o**p**y’ a file
- mv
  - ‘**m**o**v**e’ a file
- rm
  - ‘**r**e**m**ove’ a file (be careful, usually no undos!)
- echo
  - Return (like an echo) the input to the screen
- Autocomplete!



Download Some Example Files

1) Make a new folder, perhaps ‘bash_examples’, then cd into it.

2) Type the following command:

Capital ‘o’

```
wget "goo.gl/oBFKrL" -O tutorial.tar
```

3) Extract the tar file with:

```
tar -xf tutorial.tar
```

4) Delete the old tar file with

```
rm tutorial.tar
```

5) cd into the new director ‘tutorial’

Input/Output Redirection

- Typically we give input to a command as follows:
  - cat file.txt

- Make the input explicit by using “<”
  - cat < file.txt

- Change the output by using “>”
  - cat < file.txt > output.txt

- Use the **output** of one function as the **input** of another
  - cat < file.txt | less

BASH Utilities

- BASH has some awesome utilities
  - External commands not directly affiliated with BASH
  - Common on almost *all* Linux systems

BASH Utilities: bc

- bc

- bc - A **b**asic **c**alculator
- Doesn’t support floating points by default
- Use -l option to load standard math libraries

BASH Utilities: du

- bc
- du

- du - check **d**isk **u**sage
- Tells you how much hard disk space you are using
- Use -h option for ‘**h**uman’ readable units

BASH Utilities: ps

- bc
- du
- ps

- ps - List all the running **p**rocesse**s**
- Tells you what programs are running and who is running them
- Use -aux options to list all programs with more information

BASH Utilities: sleep

- bc
- du
- ps
- sleep

- sleep - Do nothing for a defined length of time
- Will cause the computer to just wait
- Use suffix s, m, h, or d to define units (seconds, minutes, etc.)

BASH Utilities: sort

- sort - Sort the given lines in ascending order
- Can sort either alphabetically or by number
- Use -r to **r**everse direction, and -kn to sort by column n

- bc
- du
- ps
- sleep
- sort

BASH Utilities: time

- time - Time how long it takes to execute a command
- Reports back in seconds by default
- Typical use is ‘time command’
- Try it with sleep!

- bc
- du
- ps
- sleep
- sort
- time

BASH Utilities: tr

- tr - **Tr**anslate or delete a character in a file
- Fast and easy way to remove all of a character from a file
- Use -d option to delete
- Use tr ‘a’ ‘b’ < input.txt > output.txt to replace all a with b

- bc
- du
- ps
- sleep
- sort
- time
- tr

BASH Utilities: grep

- grep - **G**lobally search for a **Re**gular expression and **p**rint
- Search through files to find matching text
- Uses regular expressions (a whole different discussion!)
- Use as ‘grep pattern < input.txt’

- bc
- du
- ps
- sleep
- sort
- time
- tr
- grep

BASH Utilities: awk

- awk - A full programming language itself, typically used for extracting data from files
- Can write full programs in Awk!
- Most often used for ‘one-liner’ functions
- Examples:
  - Print out third column: awk ‘{print $3}’ < input.txt
  - Sum column 6 in file: awk '{sum += $6} END {print sum}' < input.txt
  - Print any line where column 6 > 30: awk '$6 > 30' < input.txt
  - Sum column 6, but only if column 6 > 30: awk '$6 > 30 {sum += $6} END {print sum}' < input.txt

- bc
- du
- ps
- sleep
- sort
- time
- tr
- grep
- awk

BASH Utilities: awk

You Try it! Use the people_table.txt and awk to answer the questions:

1) What is the total amount of money made by people over 30?

2) By only modifying your last command, what is the average per person?

BASH Utilities: awk

You Try it! Use the people_table.txt and awk to answer the questions:

1) What is the total amount of money made by people over 30?

```
cat people_table.txt | tr -d '$' | tr -d ',' | awk '$6 > 30 {sum += $8} END {print sum}'
```

2) By only modifying your last command, what is the average per person?

```
cat people_table.txt | tr -d '$' | tr -d ',' | awk '$6 > 30 {sum += $8; count++} END {print sum/count}'
```

BASH Programming: Variables

- Variables are stored as

varName=value

  - Note: Cannot have spaces!

- Stored values are accessed with

$varName

- Special parameters:
  - $? - Contains exit status of last command
  - $0 - Name of the current running command
  - $1 - First argument of the current running command
  - env - List all current ***environment variables*** for the session

BASH Programming: Looping

- Lots of different ways to loop over commands

1)
```
for i in LIST
do
        commands;
done
```

2)
```
while CONDITION
do
        commands;
done
```

LIST examples
- {1..100..1}
- $(ls)
- 1 2 3 4 5 6
- File1 File2 File3

COND examples
- [$x -le 5]
- [$count -gt 4]
- read line
- *Many more!*

BASH Programming: Conditionals

- If/then/else statements allow branching in BASH:

```
if [ condition* ]
then
    command1;
elif [ condition* ]
then
    command2
else
    command3
fi
```

*Conditions are same as for the while loop!

BASH Programming: Functions

- User defined functions are also possible
- Input parameters are passed as space separated words:

FuncName arg1 arg2 arg3

Define Function

```
Function_name ()
{
    Commands;
}
```

Use Function

Just type: Function_name

Example

```
sayHello ()
{
    for i in {1..$1}
    do
        echo Hello $i;
    done
}
```

BASH Programming: Scripts

- BASH Script: A plain-text file of commands for BASH to run

- Can contain:
    - Bash commands (cd, ls, cat, ...)
    - Variable definitions
    - Logical statements (loops, conditionals, etc.)
    - External function calls (e.g. python calls)
    - Function definitions
    - Comments!
    - Anything else that the command line can understand

- File must start with the SheBang

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

BASH Programming: Scripts

- BASH Script: A plain-text file of commands for BASH to run

- Can contain:
  - Bash commands (cd, ls, cat, ...)
  - Variable definitions
  - Logical statements (loops, conditionals, etc.)
  - External function calls (e.g. python calls)
  - Function definitions
  - Comments!
  - Anything else that the command line can understand

- File must start with the SheBang

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