Introduction to Bash

The Shell and Shebang

- Shell – program that interfaces with operating system
- Bash – is the default shell, or interpreter, for most Linux systems (such as Proteus)
- Bash uses a wide array of commands to interface with the interpreter

- These commands can also be used in a script: my_script.sh
- A shebang is the first line of a script that tells the OS what interpreter to use.
  - Bash shebang - #!/bin/bash

Common Commands

- cd [directory]
- ls [directory]
- mkdir [directory]
- rm [file]
    - rm –rf [directory]
- mv [file] [location]
- cp [file] [location]
- less [file]
- man [command]

Writing Bash Scripts

- On Proteus
  - nano
  - vim
  - emacs
- Use your personal computer
  - Notepad++
  - Sublime
  - Visual Studio
- May need to use dos2unix if coming from Windows

Absolute vs Relative Pathing

- An absolute path is the full path from the root directory (/) to the current directory
  - pwd will display your current directory path
  - /mnt/HA/groups/testGrp

- A relative path is the path from your directory to another directory
  - . is a reference to the current directory
  - .. is a reference to the parent directory
  - ../../test/foo

Variables

- VAR=test
  - No space next to =
  - Case sensitive
- $VAR
  - Access with $
- Environment Variables are system wide variables
  - $SHELL, $HOST, $USER
  - Type “env” or “printenv” in terminal to see all
    - “printenv VAR” will display the value of $VAR
- export VAR
- Variables you create exist only for the session, set them in .bashrc in order for them to persist between sessions

Array Variables

- VAR[index]=test
- VAR=(val1 val2 val3)
- Zero based index (i.e. VAR[1] -> val2)
- Use curly braces { } to reference more than the first index
  - echo ${VAR[*]} -> val1 val2 val3
- To delete array variables use the “unset” command
  - unset VAR
  - unset VAR[1]

Command Redirection

- >
  - Sends output from a command to file
- <
  - Sends input into a command
- |
  - “pipe”
  - Sends command output to another command
- ( )
  - Subshell
- $( )
  - Command substitution

Flow Control

- if

```
if [ condition ]
then
        command
else
        command
fi
```

- for

```
for VAR in {1..5}
do
        command
done
```

Flow Control

- while

```
while [ condition ]
do
        command
done
```

- seq FIRST INCREMENT LAST
  - seq 2 5 20
    - 2
    - 7
    - 12
    - 17
  - seq –f “FORMAT” -> display in format
  - seq –s “ ” -> display as a string
  - seq –w -> pad with leading zeros

Functions

```
function test_func{
        command
}
test_func
```

- For input parameters:
- $0, $1, $2, etc. for each argument
- $0 – name of shell script

Testing

- test
- [ ]
- [[ ]]
  - More functionality than [ ]
    - No need to quote variables
    - &&
    - | |
    - =~ (match)

Arithmetic

- Integer based
  - 3 / 2 == 1
- $(( )) - arithmetic expansion
  - $((5 * 4))
- expr
  - Expressions may need escape characters
  - expr 2 + 3
  - expr 5 * 4

Let's Try It!