Bash Scripting

Scripting the command line

Outline

1. Using variables in Bash
2. Reading files
3. Conditionals & Loops
4. Functions

Variables

- **Variables are created by using an equal sign (=)**
  - No spaces between the variable, the equal sign, and the value
  - `VAR=VALUE`
  - To refer to a variable use a dollar sign ($) e.g. $VAR
- **Command line arguments**
  - Arguments passed to the script are assigned to positional variables from $1 to $n
  - The script itself is assigned to $0

Special Variables

- $# - Total number of arguments
- $@ - Values of all the arguments
- $* - Values of all arguments double quoted
- $? - Exit status of the last command
- $! - Process ID of the last command
- $IFS - Internal Field Separator
- $USER - Username of the person executing the script
- $HOSTNAME - The machine the script is running

Reading Files

- To read the contents of a file use a while loop with input redirection

```
f = open('filename', 'r')

for line in f.readlines():
    print( line.strip() )
```

```
while read line
do
    echo $line
done < filename
```

If - Else

- If/Else code uses the keywords:
  - if
  - then
  - else
  - fi
- Conditionals use square brackets ([])
- Tests use comparison operators

```
if [[ -e $1 ]]
then
  echo "It exists"
else
  echo "it does not exist"
fi
```

Comparison Operators

- -eq = Equal
- -ne = Not equal
- -gt = Greater than
- -lt = Less than
- -z = String is null
- -n = String is not null

File Test Operators

- -e = File exists
- -f = File is a file not a directory or device
- -d = File is a directory
- -s = File size is not zero
- -r = User running script has read permission to file
- -w = User running script has read write to file
- -x = User running script has read execute to file

"For" Loops

```
for VAR in LIST
do
    echo $VAR
done
```

Types of lists

- String - “apple banana cherry”
- Command - $(cat filename)
- Ranges - {1..5}

Functions

```python
#!/usr/bin/env python3

def function_name(arg):
    print(arg)
```

```bash
#!/bin/bash

function_name() {
    echo $1
}
```

Summary

1. Shell scripting allows you to automate sets of commands
2. Shell scripting uses the same commands as the shell