Shell Scripting for Computer Programmers

Andrew Thorp

Linux@App

What is shell scripting?

- BASH commands
- BASH pipes
- Environment variables
- File manipulation

Why use shell scripting

- Learning the language
- Not having to learn the language
- Avoiding large overhead/libraries
- Very easy to write
- Interact natively with OS environment

Documentation / References

Documentation can be found my typing:

```
$man <program_name>
```

Review

Pipes

- `cmd1 | cmd2` takes output of `cmd1` and feeds it as input to `cmd2`
- `cmd1 >> file` takes the output of `cmd1` and appends `file` with the contents
- `cmd2 > file` takes the output of `cmd1` and overwrites `file` with the contents
- `cmd3 < file` takes the contents of `file` and extracts them as input to `cmd3`

Basic program

- HelloMe.sh

```
#!/bin/bash
# This is a comment!
echo "Hello $USER!" # This is also a comment
```

Anatomy

```
echo "Hello $USER!"
```

The echo command

Anatomy (aside)

What is the difference here? `echo "Hello $USER!"`

vs

```
echo 'Hello $USER!'
```

Anatomy

```
# This is a comment
```

Golly Gee I wonder what this is

Usage?

Running the script

```
$ ./helloMe.sh
```

Usage

Running the script:

run `$chmod +x /path/to/script`

run `$./helloMe.sh`

Fundamentals: Variables

Variables

- prequalMeme.sh

```
# variable definition
message="hello there.\n"
name="genreal kenobi..."
#variable usage
echo "$message $name"
```

Fundamentals: Arguments

Arguments

- arguments.sh

```
# Called with argScript.sh <firstName> <lastName>
FNAME=$1
LNAME;$2
STR1="Hello $1 $2. "
STR2="You called this script with $# arguments"
STR3="The arguments were: $@"
echo $STR1 $STR2
echo $STR3
```

Fundamentals: Loops

FOR loop

- friendlyList.sh

```
for i in $( ls ); do
    echo "item: $i"
done
```

Fundamentals: Loops

FOR loop

```
for i in *; do
    echo "item: $i"
done
```

Fundamentals: Loops

FOR loop

```
for i in `seq 1 10; do
    echo "item: $i"
done
```

Fundamentals: Loops

WHILE loop

- repeater.sh

```
echo "Press Ctrl-C to escape"
while read f; do
    echo $f
done
```

Fundamentals: Conditionals

Testing

```
if [ "myString" = "myString" ]; then
    echo true
elif test 5 -lt 7
    echo "This should not be reached"
else
    echo "If you're here something is very wrong"
fi
```

Notice the spaces around the bracket

Practice

Write a quick program that takes in any number of arguments, then prints them all out in the format:

arg1: <argument 1> arg2: <argument 2>

Practice

One solution:

```
for i in 1 $#; do
    echo "arg$i: $$i"
done
```

Fundamentals: Tools

- `grep` - find
- `tr` - find and replace characters
- `echo` - print a value
- `cat` - print the contents of a file
- `sleep` - sleeps for a given number of seconds
- `bc` - basic calculator

grep

- Used to pattern match
- Accepts regular expressions
- Should not be used like `cat myFile | grep "pattern"`.
- General syntax: `grep "pattern / expression" <file/files>`

tr

- Replaces characters with other characters (called “sets”)
- Usefull for manipulating text files
- Basic syntax: (replaces each space witha newline) `cat myFile | tr ' ' '\n'`

echo

- Prints input to standard out
- Used for piping variables into commands
- Basic syntax: `echo "Hello World!"`

cat

- Stands for concatinate
- Outputs contents of file to stnadard out.
- Notoriously overused
- Basic Syntax: `cat myFile`