Module Summary

The Shell Script File

- Naming shell scripts and file extensions.
- Shell script file permissions.

```
chmod 755 script
chmod +x script
```

The Basics

- Shebang

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

- Comments

```
#
```

- Variables

```
VAR="value"
```

Creating Standard Output and Quoting

- The `echo` builtin.
- Single versus double quotes

```
"${VAR} gets expanded."
'${VAR} does NOT get expanded.'
```

Getting Help for Shell Builtins

```
type: type [-afptP] name [name ...]
    For each NAME, indicate how it would be
interpreted if used as a command name.


help: help [-s] [pattern ...]
    Display helpful information about
builtin commands.
```

Getting Help for Linux Commands

```
man - format and display the
on-line manual pages
```

Shell Variables

- The shell sets several variables automatically.

```
HOSTNAME
RANDOM
UID
...
```

Command Substituion

```
VAR=$(command)

VAR=`command`    # Old style.

echo "Output of command: $(command)"
```

Pseudocode

```
# First, do this.

# Next, do this.

# Finally, do that.
```

The `if` statement

```
if [[ COMMANDS ]]
then
  COMMANDS
else
  COMMANDS
fi
```

Exit Statuses

- 0 = true / successful
- 1 = false / unsuccessful
- Any non-zero exit status represents a failure.

```
exit 1
echo ${?} # You can also use $?
```

Sanity Checking

- Don't assume; check!

```
if [[ "${UID}" -ne 0 ]]
then
   echo 'Please run as root.' >&2
   exit 1
fi
```

Command Line Conventions

```
id -u -n

id -un

id -nu

id --user --name
```

Obtaining Standard Input

- The `read` shell builtin.

```
read -p "A prompt: " VARIABLE
```

Generating Random Data

- The $RANDOM shell variable.

```
echo ${RANDOM}
```

- Seemingly random data using checksums.

```
date +%s%N | sha256sum | head -c8
```

Positional Parameters

Arguments vs Parameters

`$0` = Stores the script name.

`$1` = Stores the first argument.

`$2` = Stores the second argument.

`$*` = When used in quotes: `"$1 $2..."`

`$@` = When used in quotes: `"$1" "$2"...`

`$#` = The number of positional parameters.

The `for` Loop

```
for VARIABLE in LIST
do
  COMMANDS
done
```

The **`while`** Loop

```
while [[ COMMANDS ]]
do
  COMMANDS
done
```

I/O Redirection - Pipes

- Sending STDOUT as STDIN.

```
echo ${PWORD} | passwd --stdin ${NAME}
```

- String manipulation & data munging with pipes.

```
echo '!@#$%^' | fold -w1 | shuf | head -c1
```

File I/O Redirection

```
COMMAND > /path/to/file

COMMAND >> /path/to/file

COMMAND < /path/to/file

COMMAND 2> /path/to/file

COMMAND &> /path/to/file
```

I/O Redirection

```
COMMAND |& COMMAND
```

```
COMMAND >&2
```

```
COMMAND > /dev/null
```

You're doing great…
Keep it up!