Linux Classroom Series – 28/Aug/2020

Variable Scopes in Shell scripts

- Lets create two shell scripts
  - script1.sh

```
#!/bin/bash

name="Learning Thoughts"
echo "This is from script 1 name= ${name}"
./script2.sh
```

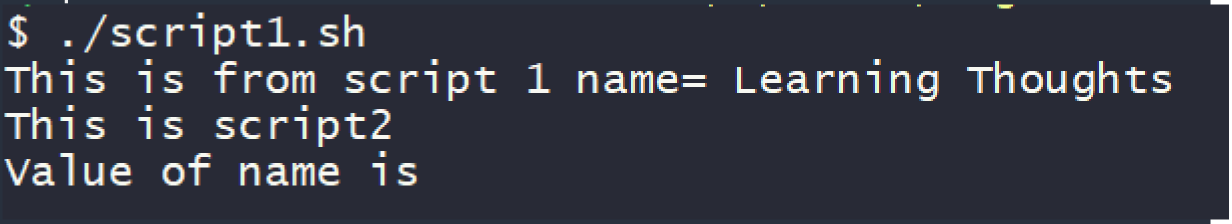
  - script2.sh

```
#!/bin/bash

echo "This is script2"
echo "Value of name is ${name}"
```

- From script1 lets call script2 and try to see if the variable defined in script1 is available for use in script2

- lets execute script1.sh

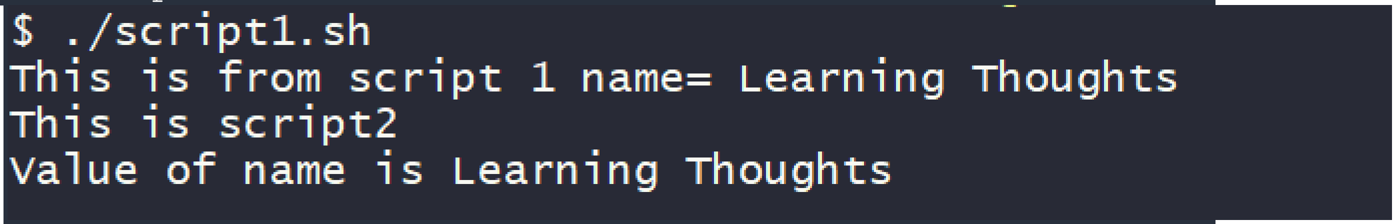


```
$ ./script1.sh
This is from script 1 name= Learning Thoughts
This is script2
Value of name is
```

- The value of name is not available in script2. The default scope of variable is the same script file.
- So lets find out if there is any other way to pass the variable from script1 to external scripts,

1. add Arguments to script2

- `./script2.sh`
- `# replace this with`
- `./script2.sh $name`
  - export the variable using export statement in script1. so lets change script1.sh
- `#!/bin/bash`
-
- `name="Learning Thoughts"`
- `echo "This is from script 1 name= ${name}"`
- `export name`
- `./script2.sh`



```
$ ./script1.sh
This is from script 1 name= Learning Thoughts
This is script2
Value of name is Learning Thoughts
```

Debugging your scripts

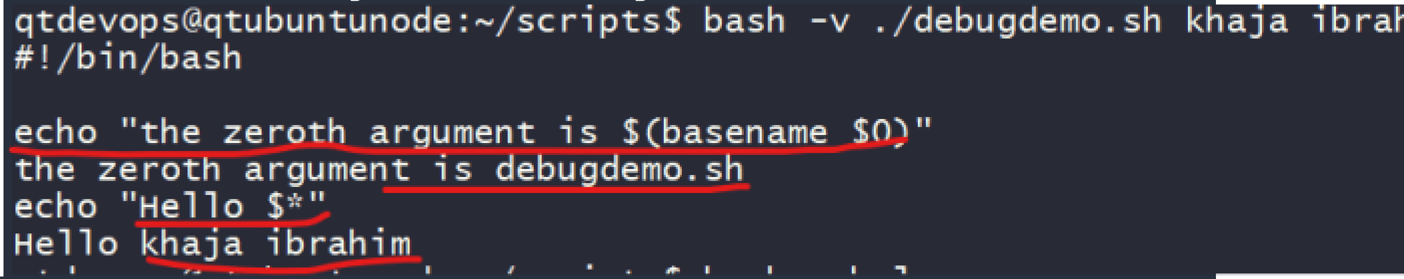
- As the script grows and decision paths are included with conditional statements, we start using looping structures etc, we may need some level of debugging to analyse the scripts.
- This can be done as bas provides two options for us
  - -v option
  - -x option
- Create a bash script debugdemo.sh with the following content

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

```
echo "the zeroth argument is $(basename $0)"
echo "Hello $*"
```

- Now execute this script with the following command

```
bash -v ./debugdemo.sh khaja ibrahim
```

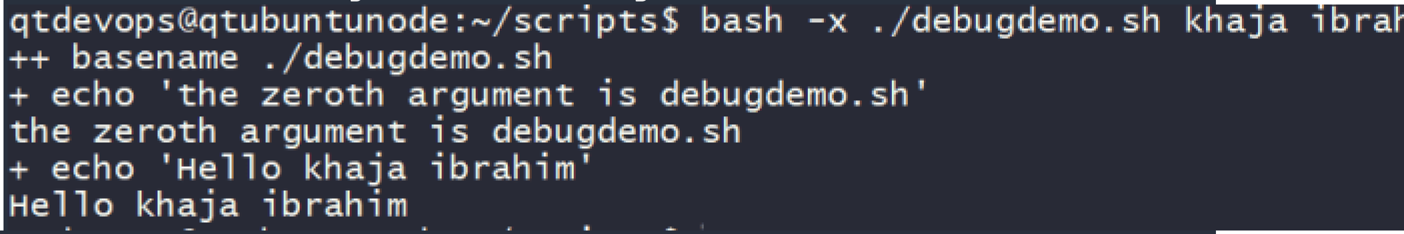


```
qtdevops@qtubuntunode:~/scripts$ bash -v ./debugdemo.sh khaja ibrah
#!/bin/bash

echo "the zeroth argument is $(basename $0)"
the zeroth argument is debugdemo.sh
echo "Hello $*"
Hello khaja ibrahim
```

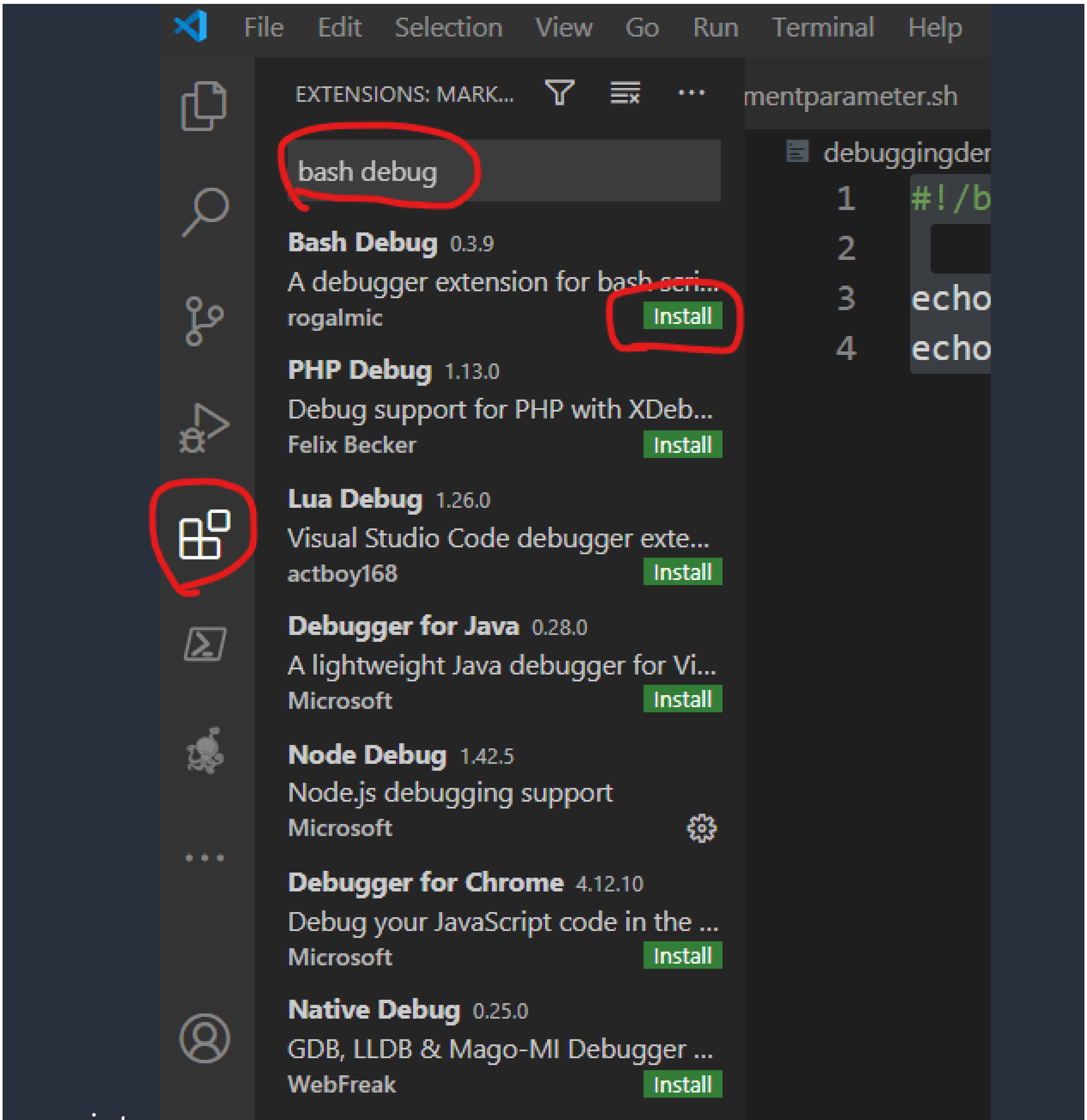
- -v option displays the verbose output from bash
- -x option, which displays the commands as they are executed and is most commonly used.
- Now lets run the script

```
bash -x ./debugdemo.sh khaja ibrahim
```



```
qtdevops@qtubuntunode:~/scripts$ bash -x ./debugdemo.sh khaja ibrah
++ basename ./debugdemo.sh
+ echo 'the zeroth argument is debugdemo.sh'
the zeroth argument is debugdemo.sh
+ echo 'Hello khaja ibrahim'
Hello khaja ibrahim
```

- This way shows how the evaluated and is very helpful to know the decision branch that has been chosen by the script.
- Install bash *debug plugin* in visual studio code and we can debug the shell



bash debug

Bash Debug 0.3.9
A debugger extension for bash scri...
rogalmic Install

PHP Debug 1.13.0
Debug support for PHP with XDeb...
Felix Becker Install

Lua Debug 1.26.0
Visual Studio Code debugger exte...
actboy168 Install

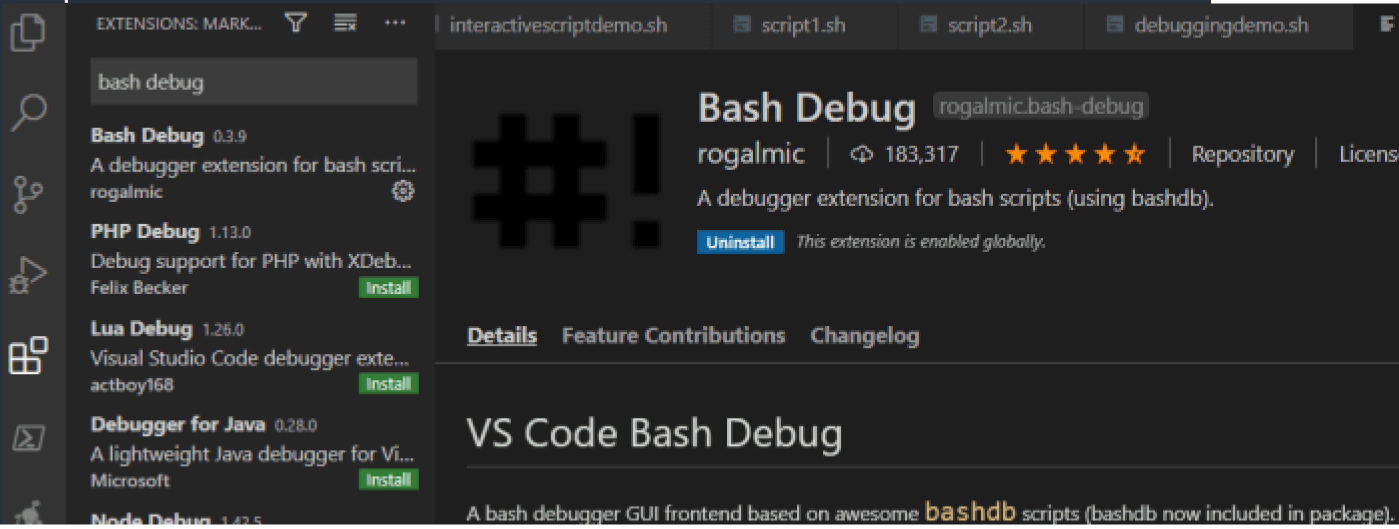
Debugger for Java 0.28.0
A lightweight Java debugger for Vi...
Microsoft Install

Node Debug 1.42.5
Node.js debugging support
Microsoft

Debugger for Chrome 4.12.10
Debug your JavaScript code in the ...
Microsoft Install

Native Debug 0.25.0
GDB, LLDB & Mago-MI Debugger ...
WebFreak Install

script.



bash debug

Bash Debug 0.3.9
A debugger extension for bash scri...
rogalmic

PHP Debug 1.13.0
Debug support for PHP with XDeb...
Felix Becker Install

Lua Debug 1.26.0
Visual Studio Code debugger exte...
actboy168 Install

Debugger for Java 0.28.0
A lightweight Java debugger for Vi...
Microsoft Install

Node Debug 1.42.5

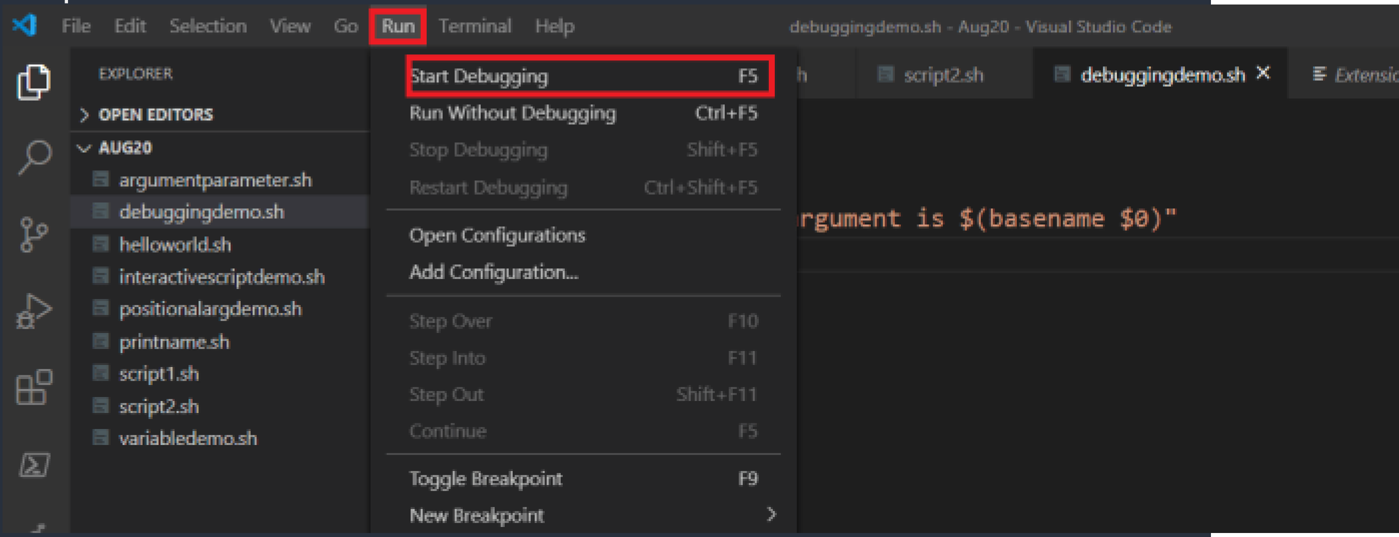
Bash Debug rogalmic.bash-debug
rogalmic | 183,317 | ★★★★★ | Repository | Licens
A debugger extension for bash scripts (using bashdb).
Uninstall *This extension is enabled globally.*

Details Feature Contributions Changelog

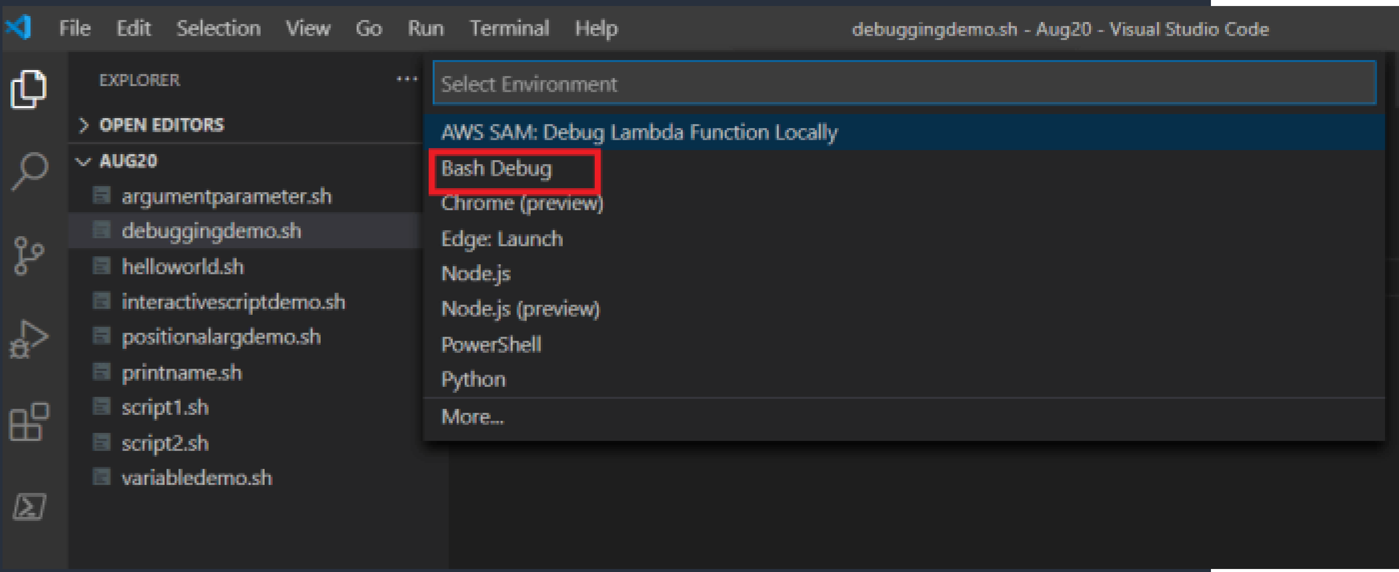
VS Code Bash Debug

A bash debugger GUI frontend based on awesome bashdb scripts (bashdb now included in package).

- Now we can debug the shell script



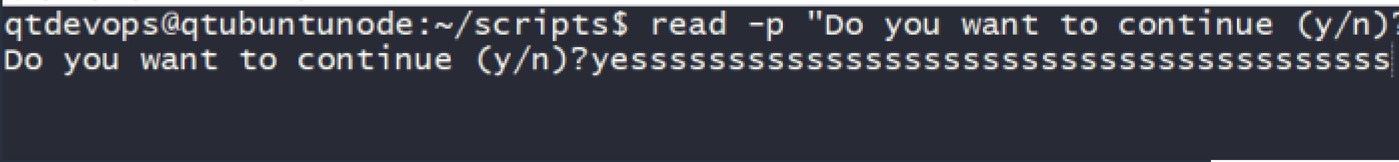
```
rgument is $(basename $0)"
```



- This approach works with visual studio code installed on the linux desktops or mac.

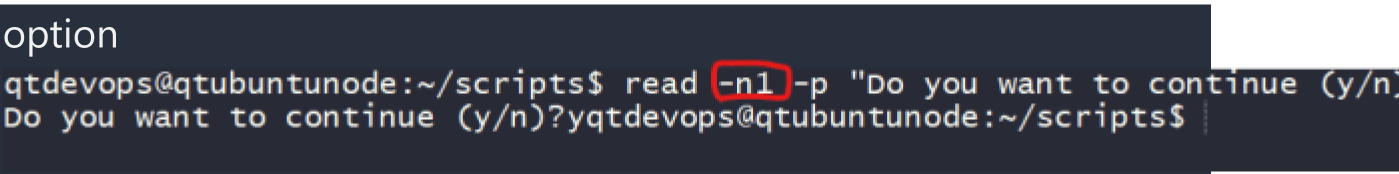
Enhancing interactive scripts

- Lets understand how to limit number of characters entered



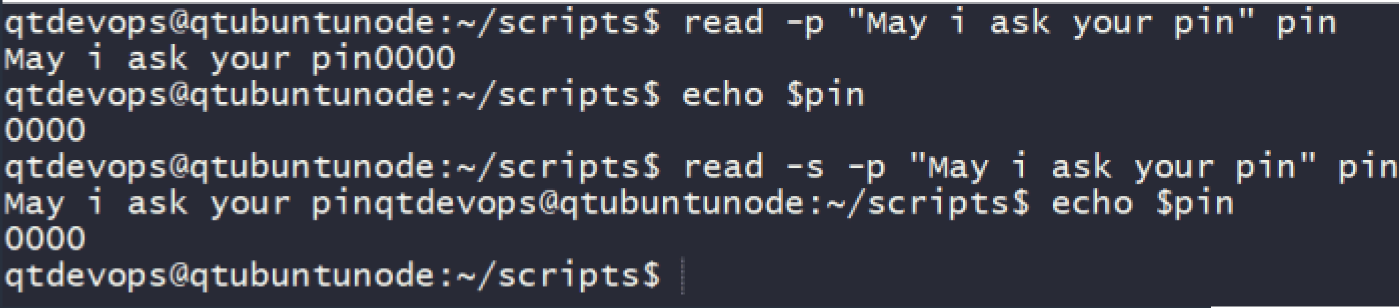
```
qtdevops@qtubuntunode:~/scripts$ read -p "Do you want to continue (y/n)?
Do you want to continue (y/n)?yessssssssssssssssssssssssssssssssssss
```

- so limit number of characters to be entered, lets run the same command with one more option



```
qtdevops@qtubuntunode:~/scripts$ read -n1 -p "Do you want to continue (y/n)
Do you want to continue (y/n)?yqtdevops@qtubuntunode:~/scripts$
```

- Lets understand how to control visibility of entered text, If we request for sensitive content like pin/password etc, its not a good approach to show the text while user is typing, Shell gives an option of hiding text when the user is typing. This can be achieved by adding a -s option to read command



```
qtdevops@qtubuntunode:~/scripts$ read -p "May i ask your pin" pin
May i ask your pin0000
qtdevops@qtubuntunode:~/scripts$ echo $pin
0000
qtdevops@qtubuntunode:~/scripts$ read -s -p "May i ask your pin" pin
May i ask your pinqtdevops@qtubuntunode:~/scripts$ echo $pin
0000
qtdevops@qtubuntunode:~/scripts$
```

Next Steps:

- How to make my shell scripts understand options (named parameters)

```
./downloadfile.sh --url <>  --location-to-save
/home/ubuntu/test.txt
```