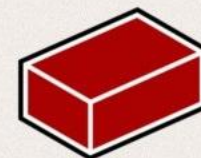


# Intro to Shell Scripting

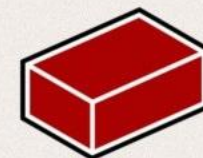
`#!/bin/bash`

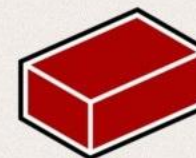


**Redbrick**  
DCU's Networking Society

# Shell? Like from the sea?

- Provides an interface between you and your OS
- Typically accepts commands from a terminal
- Can also read commands from a file
  - This is what we're looking at today
- Many shells exist:
  - bash, zsh, tcsh, ksh, csh
- We'll just be dealing with one today:





**Redbrick**

DCU's Networking Society

# Bash

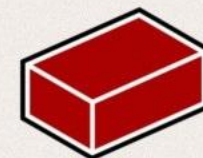
- Unix Shell & Command Language
- **Bourne Again SHell**
  - Replaced the Bourne Shell

# What You'll Need:

- A computer running **Linux** or **OSX**
  - This can also be done on Redbrick
- A Terminal Emulator
  - GNOME Terminal, Xterm, etc
- A Text Editor
  - Vim, nano, gedit
  - We recommend a terminal editor
- No prior knowledge needed!
  - We'll learn you a bash for great good

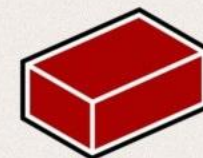
# Hello, World!

```
1 #!/bin/bash
2
3 FOO="Hello, World!"
4
5 echo $FOO
```



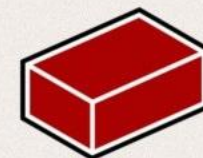
**Redbrick**  
DCU's Networking Society

- `#!/bin/bash`
  - This is where your interpreter is located
- `FOO="Hello, World!"`
  - Declares a variable 'FOO', assigns a string to it
- `echo $FOO`
  - Prints the contents of 'FOO'
  - '\$' used when calling variables



# Let's make it run!

- Move to the folder your script is in:
  - `cd path/to/yourscript.sh`
- Make it executable:
  - `chmod +x yourscript.sh`
- And run:
  - `./yourscript.sh`

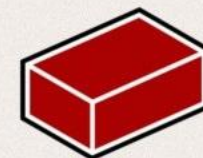




# Success! (Hopefully...)

```
[pints@meccano ~]$ cd Documents/introtobash/  
[pints@meccano introtobash]$ chmod +x helloworld.sh  
[pints@meccano introtobash]$ ./helloworld.sh  
Hello, World!  
[pints@meccano introtobash]$ █
```

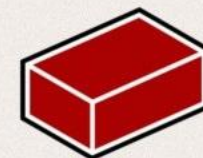
- You should be seeing something along these lines.
- If you don't see this and/or your machine is on fire, please raise your hand.



# A More Useful Example

# Check if a Number is Odd or Even

```
1 #!/bin/bash
2 echo "Enter a Number."
3 read n
4 num=$(expr $n % 2)
5 if [ $num -eq 0 ]
6     then
7         echo "Even Number."
8     else
9         echo "Odd Number."
10    fi
```

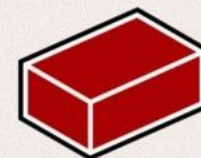


- `echo "Enter The Number"`
  - Prints the quoted text to the screen
- `read n`
  - Allows user input, stored in variable 'n'
- `num=$((expr $n % 2))`
  - Outputs the remainder after dividing by 2 the maximum number of times possible
- `if [$num -eq 0]`
  - Checks if this remainder is 0
- `then \ echo "Even Number."`
  - Tells us the number is even, if the remainder is 0.

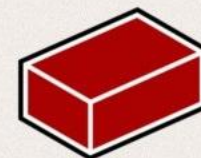
- `else`
  - If the remainder is not 0:
- `echo "Odd Number."`
  - Prints
  - A non-zero remainder indicated an odd number
- `fi`
  - Terminates the 'if' statement.

# And let's run it!

```
1 #!/bin/bash
2 echo "Enter a Number."
3 read n
4 num=$(expr $n % 2)
5 if [ $num -eq 0 ]
6     then
7         echo "Even Number."
8 else
9     echo "Odd Number."
10 fi
```



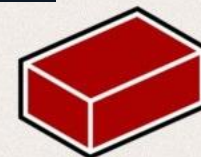
One More example!



**Redbrick**  
DCU's Networking Society

# Fibonacci Sequence

```
1 #!/bin/bash
2 echo "How many numbers do you want of Fibonacci series ?"
3   read total
4   x=0
5   y=1
6   i=2
7   echo "Fibonacci Series up to $total terms :: "
8   echo "$x"
9   echo "$y"
10  while [ $i -lt $total ]
11  do
12      i=`expr $i + 1 `
13      z=`expr $x + $y `
14      echo "$z"
15      x=$y
16      y=$z
17  done
```



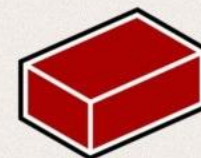


# Once again, let's run it!

- `chmod +x fibonacci.sh`
  - Makes the file executable
- `./fibonacci.sh`
  - Runs the script

# Bash can do a lot more...

All you need is imagination!



**Redbrick**  
DCU's Networking Society

# You can play Minesweeper!

```
board : L      size : 20*20  mine : 60  flag : 0  :(
* # # # # * # # * # # * # # # # # # # # #
# # # # # # # # # 1 1 1 1 1 # # # # # # # *
# # # # # # # # # 1 . . . 1 # # * * # # #
# # # # * * # * 1 . . . 1 * # # # # # # *
# # # # # # # * # 1 . . . 1 # # # # # # #
# # # # # # # * # 1 1 1 . 1 # # # # # # #
# # # # # # # # # # * 2 1 2 * # # # # # #
# # # * * # # 1 # # # # * # # # # # # * #
# * * # # # # * # * # * # # # # # # # #
# # # # [*] # # # # # # # # # # # # * * #
* * # * # # # # # # # # * # # # # # # #
* # # # # * # * # # # # # # # # # # #
# # # # # # # # # 1 # * # # # * # # # # *
# # # # # # # # # # # # # # * # # # # # *
* # * # # # # # * # # # # # * # # # # #
# # # # # # # # # # # # # 2 # # # # # * *
# # # # # * * * # # # # # # * # # # * # #
# * # # # * # # # # * # # # # # # # # #
# # # * # # # # # # # # # # # * # # # #
# # # # * # * * # * # # # # # # # # #
<h/j/k/l> Move <g> Step <f> Flag <n/N/m/M> New <q> Quit
```


# You can Edit Images!

...wait, what?

```
1 #!/bin/bash
2
3 icon="/home/pints/.i3/i3lock/icon.png"
4 tmpbg="/home/pints/tmp/screen.png"
5 tmp_lock="/home/pints/tmp/lock_screen.png"
6
7 xaxis=$(xdpyinfo | grep dimensions | uniq | awk '{print $2}' | cut -d 'x' -f1)
8 yaxis=$(xdpyinfo | grep dimensions | uniq | awk '{print $2}' | cut -d 'x' -f2)
9
10 #Grab current screen contents
11 scrot -z -q 100 "$tmpbg"
12
13 #Pixelate
14 convert "$tmpbg" -scale 10% -scale 1000% "$tmpbg"
15
16 #Tile a 10x10 circular cutout
17 convert -sample 10x10 xc: -draw 'circle 5,5 5,9' -negate \
18     -write mpr:spot +delete \
19 "$tmpbg" -scale 100% -size "$xaxis"x"$yaxis" tile:mpr:spot \
20 +swap -compose multiply -composite "$tmp_lock"
21
22 #Add a lock icon to the centre of the image
23 composite -gravity center "$icon" "$tmp_lock" "$tmp_lock"
24
25 #enable i3lock with colours modified image
26 i3lock --textcolor=ffffff00 --insidecolor=ffffff00 --ringcolor=ffffff00 --linecolor=ffffff00 --keyhlcolor=00FF00
    80 --ringvercolor=0000FF00 --insidevercolor=00000000 --ringwrongcolor=00000055 --insidewrongcolor=FF00001c -i "
    $tmp_lock"
27
28 #clean up
29 rm "$tmpbg"
30 rm "$tmp_lock"
```



### Event Details



**Event Location:** Opium Cafe, Wexford St, Dublin

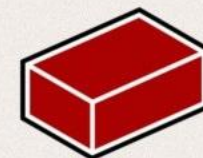
**Date:** 9th of April

**Tickets:** [Get Tickets From Tito](#)

Read maps.googleapis.com



# Questions?



**Redbrick**  
DCU's Networking Society



# More fun awaits:

[www.github.com/butlerx/bash-scripts](https://www.github.com/butlerx/bash-scripts)