BASH Flow Control

SCRIPTING ESSENTIALS

DR. BURKMAN

BASH Flow Control

Recommended books if you are planning to use Linux and BASH:

- http://linuxcommand.org/tlcl.php
- https://nostarch.com/tlcl2

BASH Flow Control

Focus here is on elements required to complete our scripts.

Nano <filename.sh> to make the file and open the editor

#! /bin/sh first line (this is the shebang)

Ctrl x to exit, y to save, enter

Chmod 750 <filename.sh> to make the script executable

./<filename.sh> to execute the script

Tip: after entering some commands you can scroll through them with the up/down arrow keys on your keyboard

Tip: Precision is key with BASH. One errant space will end you.

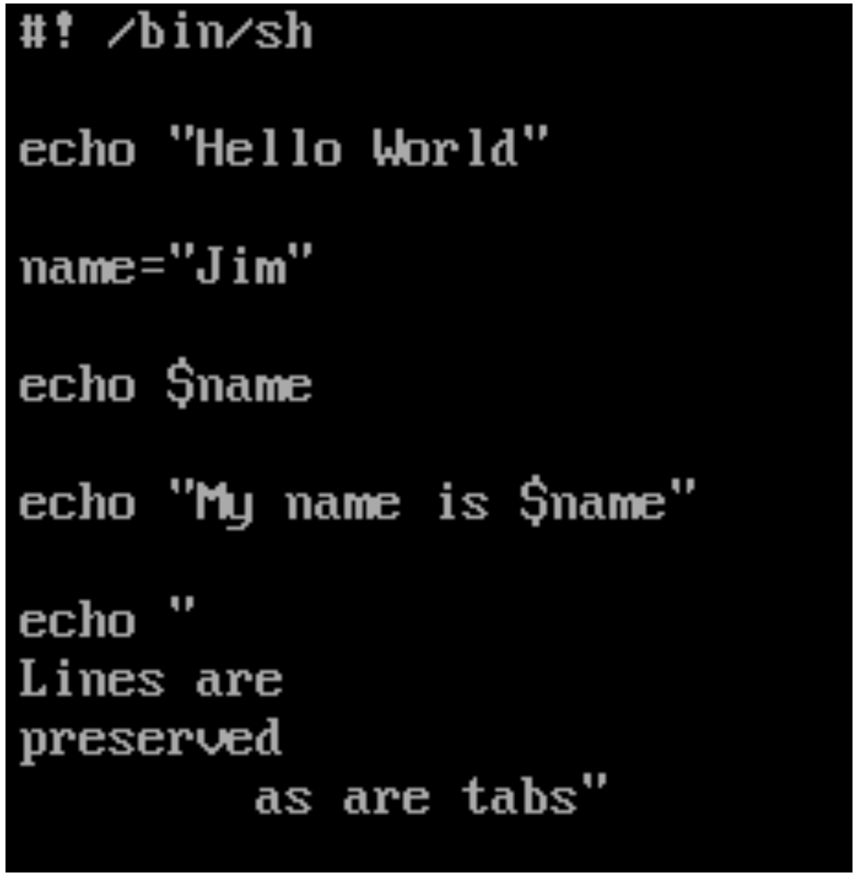
Writing Output

Write output to the screen with echo.

- Text goes in double quotes.
- Variables in the double quotes will show their values.
- Spaces, new lines etc. will print out (echo –e)
- echo by itself will make a blank line
- echo –e if you want to use \n

Comment lines by putting # in front of the line

Clear will clear the screen

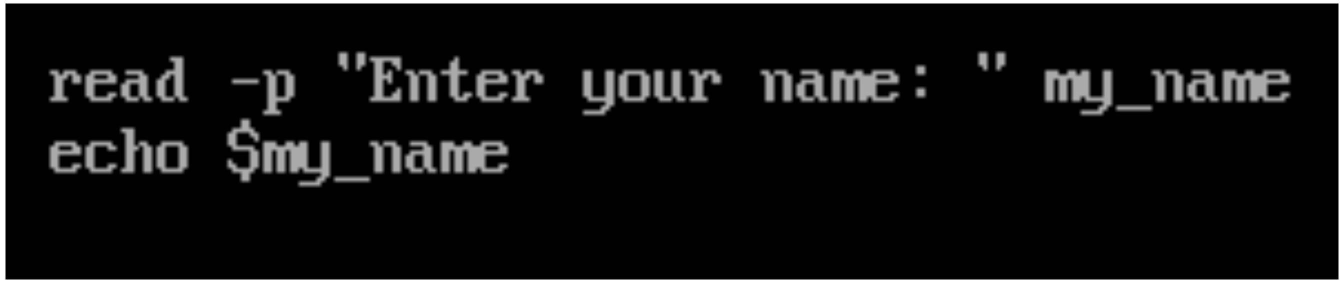


```
#! /bin/sh

echo "Hello World"

name="Jim"

echo $name

echo "My name is $name"

echo "
Lines are
preserved
        as are tabs"
```

Getting User Data

We get data from the user with the read command. We use –p for the prompt, and put the variable name at the end.

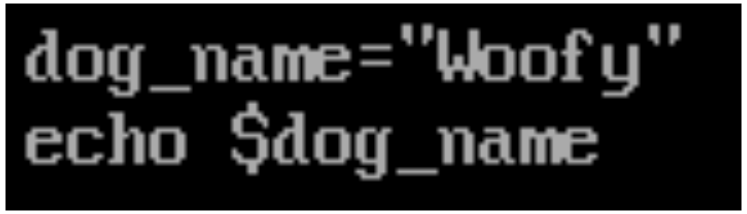
- Note: variables are assigned without using $. But we reference variables with $.



```
read -p "Enter your name: " my_name
echo $my_name
```

Assigning values to variables

There cannot be a space between the variable name and the value:



```
dog_name="Woofy"
echo $dog_name
```

Numeric Comparisons

Like PowerShell

- -eq, -ge, -gt, -le, -lt, -ne

Logical operators

- || or
- && and

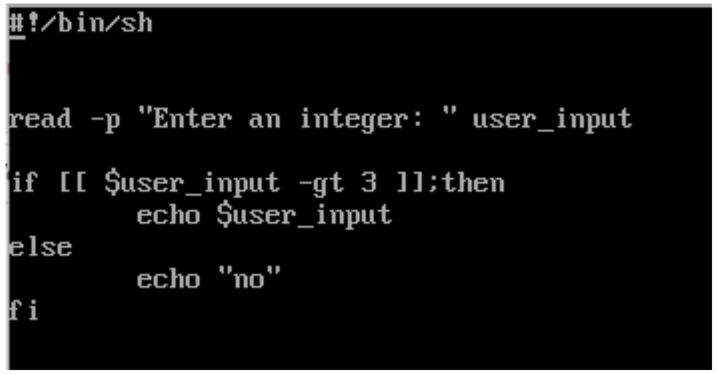
Conventions

Conventions

- For variables that have values that don’t change, put them in upper case
  - YEAR=2020
- All other variables just be consistent
  - my_name
  - myName
  - MyName

IF Statements

```
if [[ statement ]]; then
        do these things
else
        do these things
fi
```



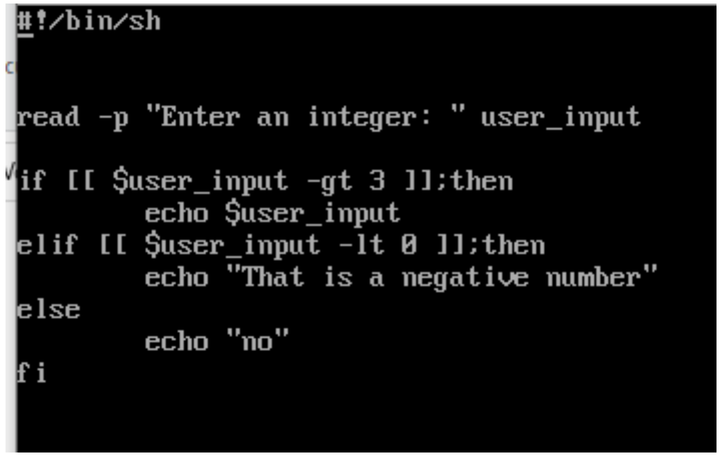
```
#!/bin/sh


read -p "Enter an integer: " user_input

if [[ $user_input -gt 3 ]];then
        echo $user_input
else
        echo "no"
fi
```

Note: there has to be spaces on either side of the condition being checked by the IF statement

http://tldp.org/LDP/Bash-Beginners-Guide/html/sect_07_01.html

IF Statements

```
if [[ statement ]]; then
        do these things
elif [[ statement ]];then
        do these things
else
        do these things
fi
```



```
#!/bin/sh


read -p "Enter an integer: " user_input

if [[ $user_input -gt 3 ]];then
        echo $user_input
elif [[ $user_input -lt 0 ]];then
        echo "That is a negative number"
else
        echo "no"
fi
```

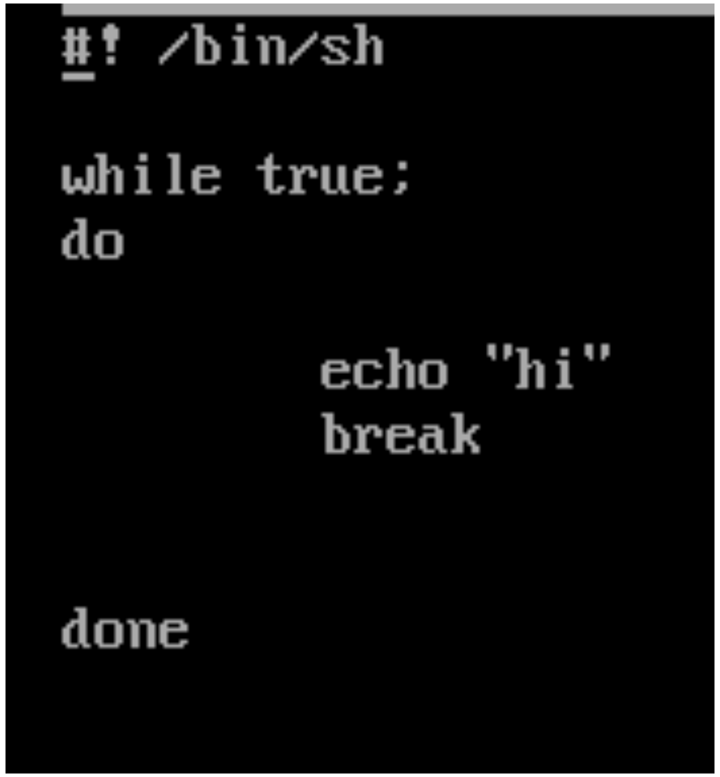
It works without indenting but that’s poor readability. Don’t be that person.

While

while true;
do

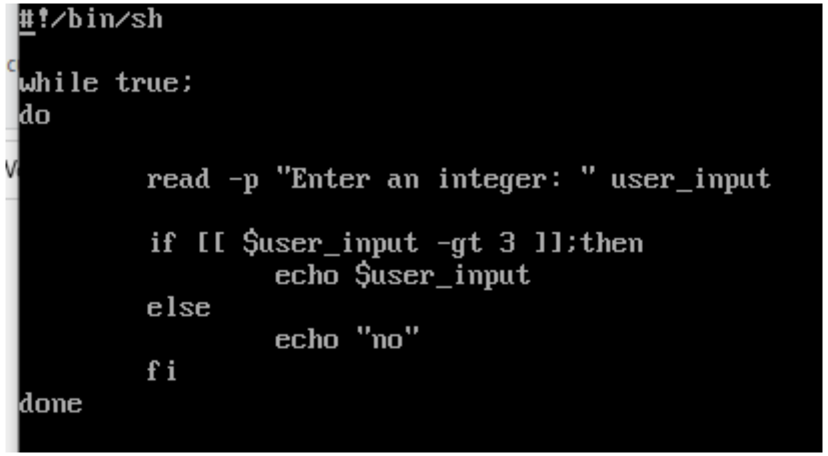
code stuff

done

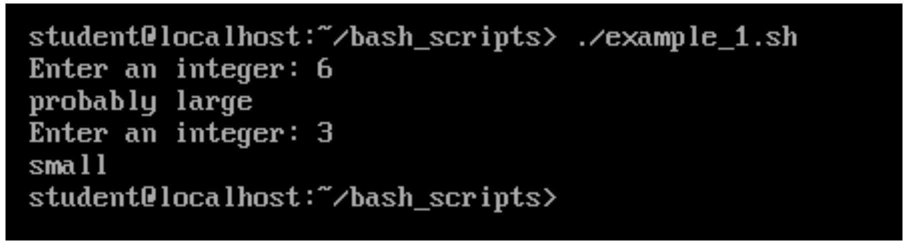


```
#! /bin/sh

while true;
do

        echo "hi"
        break


done
```

While

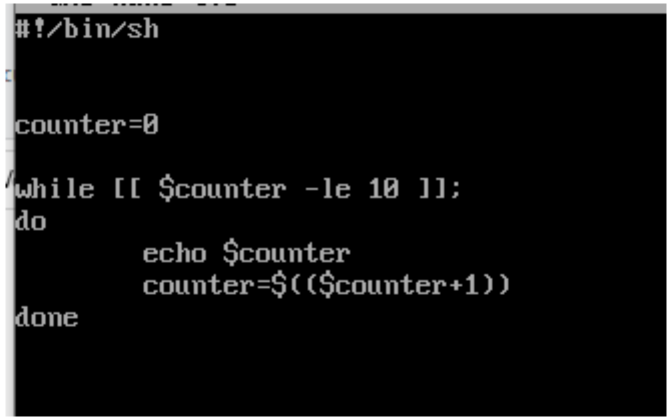


```
#!/bin/sh

while true;
do

        read -p "Enter an integer: " user_input

        if [[ $user_input -gt 3 ]];then
                echo $user_input
        else
                echo "no"
        fi
done
```

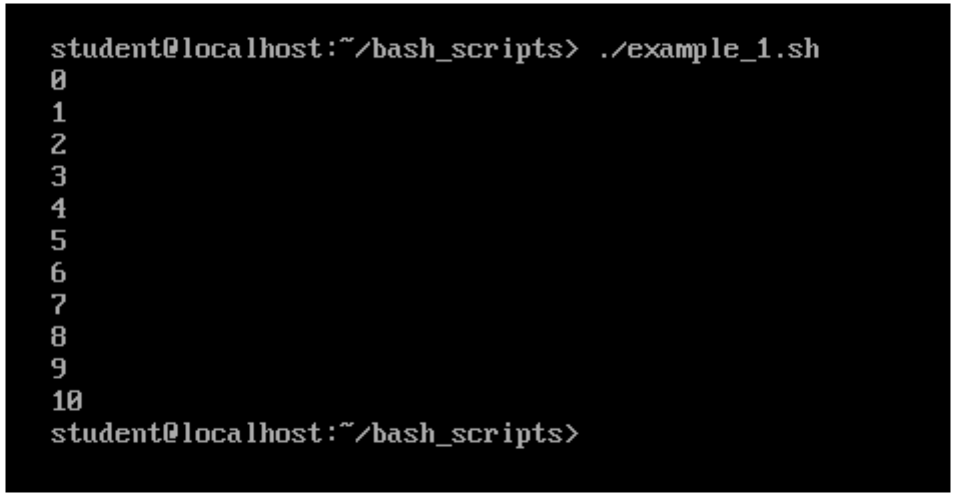


```
student@localhost:~/bash_scripts> ./example_1.sh
Enter an integer: 6
probably large
Enter an integer: 3
small
student@localhost:~/bash_scripts>
```

While



```
#!/bin/sh


counter=0

while [[ $counter -le 10 ]];
do
        echo $counter
        counter=$(($counter+1))
done
```



```
student@localhost:~/bash_scripts> ./example_1.sh
0
1
2
3
4
5
6
7
8
9
10
student@localhost:~/bash_scripts>
```

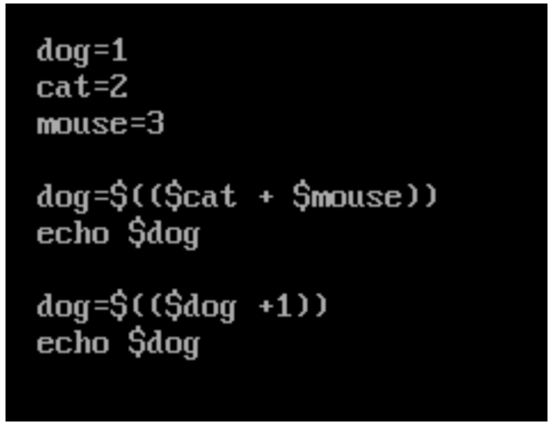
Adding the result to a variable

You cannot type dog=($cat + $mouse)

You have to type dog=$(($cat + $mouse))

You cannot type dog += 1

You have to type dog=$(($dog + 1))

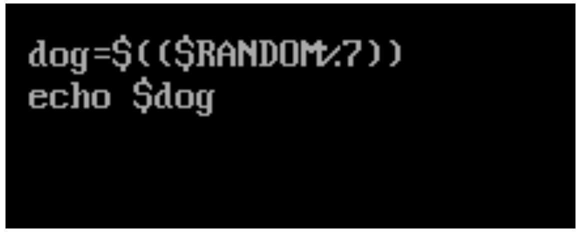


```
dog=1
cat=2
mouse=3

dog=$(($cat + $mouse))
echo $dog

dog=$(($dog +1))
echo $dog
```

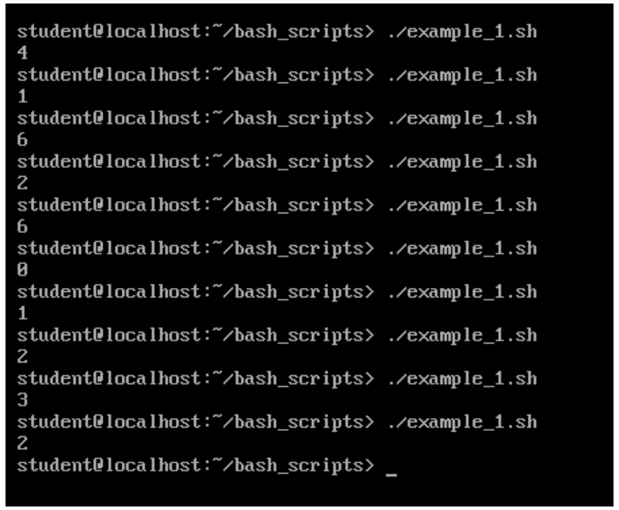
Getting random values

Linux will grab you a random integer. You can just type echo $RANDOM at a command prompt. You cannot specific a lower or upper bounds, though. So we have to take the modulus of that random number.

Mod 7 of a number will give us the integer amount remaining after dividing that number by 7. So values might be 0, 1, 2, 3, 4, 5 or 6. That's how we can get values between 0 and 6.



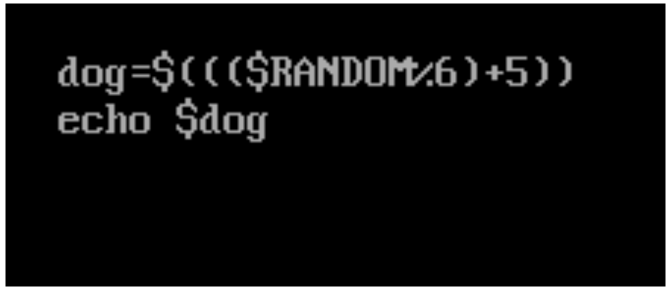
```
dog=$(($RANDOM%7))
echo $dog
```



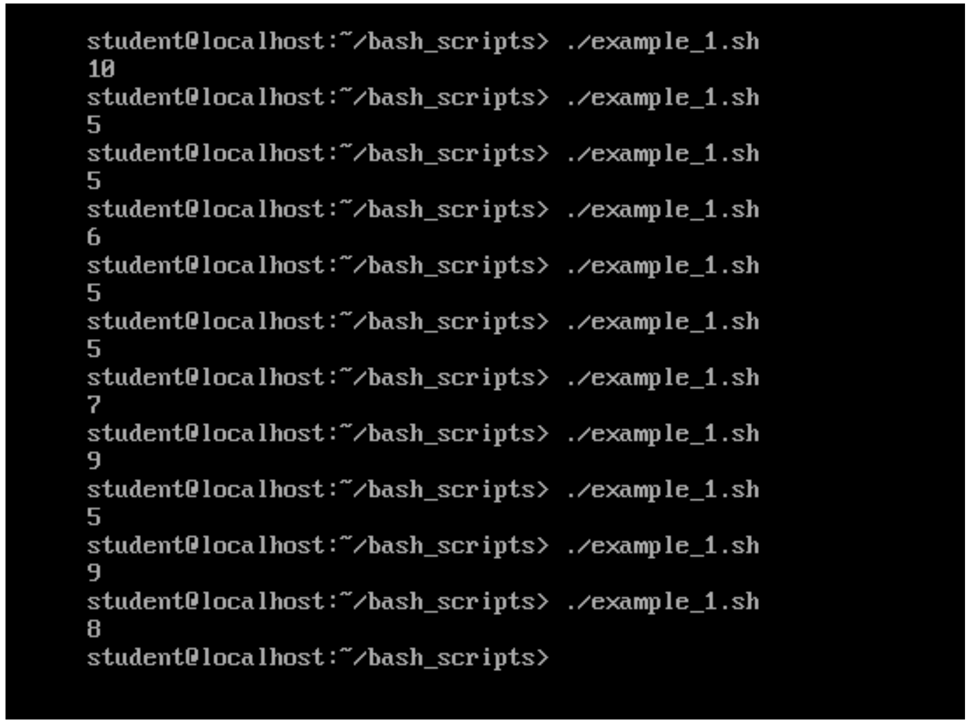
```
student@localhost:~/bash_scripts> ./example_1.sh
4
student@localhost:~/bash_scripts> ./example_1.sh
1
student@localhost:~/bash_scripts> ./example_1.sh
6
student@localhost:~/bash_scripts> ./example_1.sh
2
student@localhost:~/bash_scripts> ./example_1.sh
6
student@localhost:~/bash_scripts> ./example_1.sh
0
student@localhost:~/bash_scripts> ./example_1.sh
1
student@localhost:~/bash_scripts> ./example_1.sh
2
student@localhost:~/bash_scripts> ./example_1.sh
3
student@localhost:~/bash_scripts> ./example_1.sh
2
student@localhost:~/bash_scripts> _
```

Getting random values

If I wanted values between 5 and 10, I'd need to generate values 0 through 5 and add 5. So mod 6 will give me values 0 through 5. So..



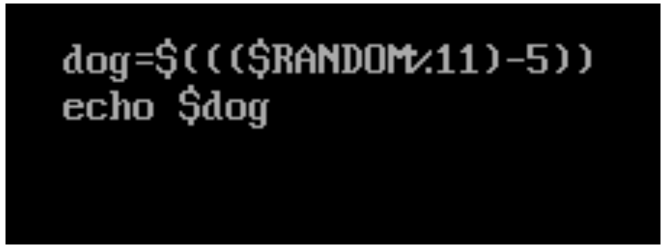
```
dog=$((($RANDOM%6)+5))
echo $dog
```



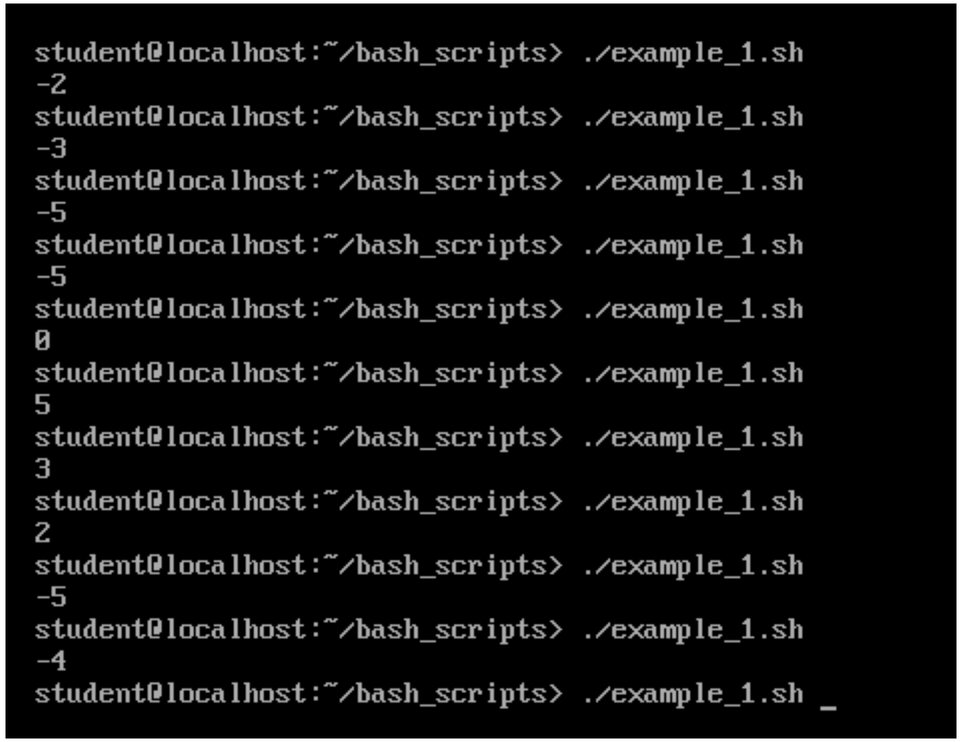
```
student@localhost:~/bash_scripts> ./example_1.sh
10
student@localhost:~/bash_scripts> ./example_1.sh
5
student@localhost:~/bash_scripts> ./example_1.sh
5
student@localhost:~/bash_scripts> ./example_1.sh
6
student@localhost:~/bash_scripts> ./example_1.sh
5
student@localhost:~/bash_scripts> ./example_1.sh
5
student@localhost:~/bash_scripts> ./example_1.sh
7
student@localhost:~/bash_scripts> ./example_1.sh
9
student@localhost:~/bash_scripts> ./example_1.sh
5
student@localhost:~/bash_scripts> ./example_1.sh
9
student@localhost:~/bash_scripts> ./example_1.sh
8
student@localhost:~/bash_scripts>
```

Getting random values

If I want numbers from -5 to 5 then I'll need to generate numbers from 0 to 10 and subtract 5. Mod 11 will get me numbers from 0 to 10, so..



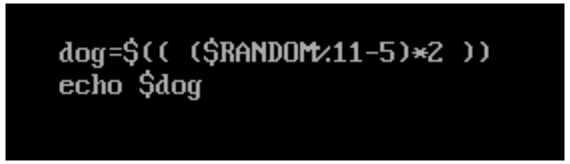
```
dog=$((($RANDOM%11)-5))
echo $dog
```



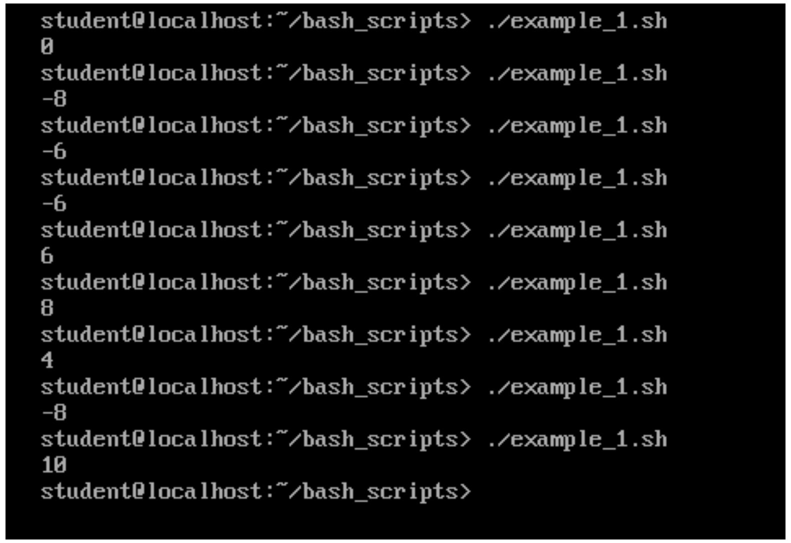
```
student@localhost:~/bash_scripts> ./example_1.sh
-2
student@localhost:~/bash_scripts> ./example_1.sh
-3
student@localhost:~/bash_scripts> ./example_1.sh
-5
student@localhost:~/bash_scripts> ./example_1.sh
-5
student@localhost:~/bash_scripts> ./example_1.sh
0
student@localhost:~/bash_scripts> ./example_1.sh
5
student@localhost:~/bash_scripts> ./example_1.sh
3
student@localhost:~/bash_scripts> ./example_1.sh
2
student@localhost:~/bash_scripts> ./example_1.sh
-5
student@localhost:~/bash_scripts> ./example_1.sh
-4
student@localhost:~/bash_scripts> ./example_1.sh _
```

Getting random values

And if I want to multiply that result by 2:



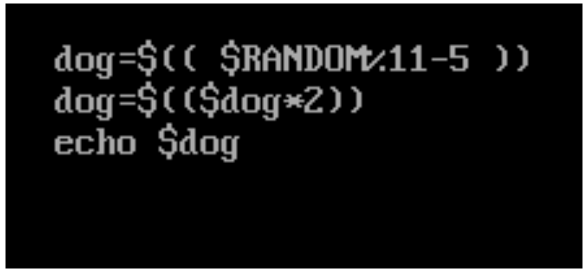
```
dog=$(( ($RANDOM%11-5)*2 ))
echo $dog
```



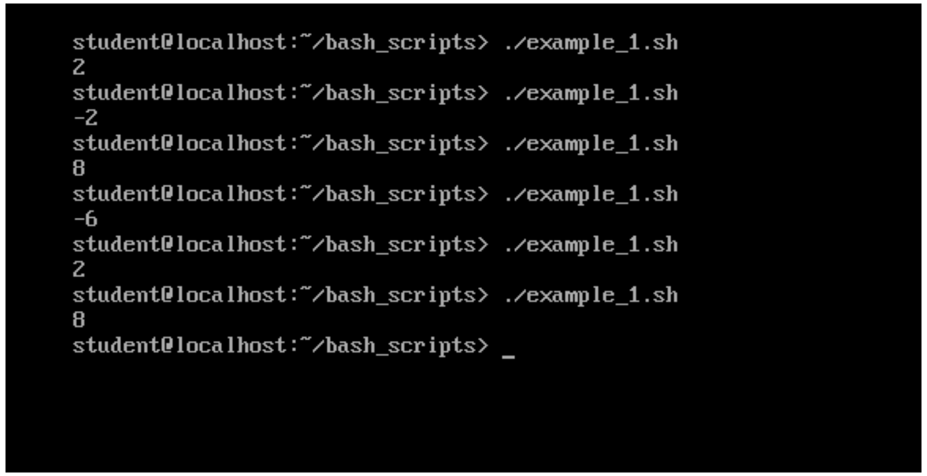
```
student@localhost:~/bash_scripts> ./example_1.sh
0
student@localhost:~/bash_scripts> ./example_1.sh
-8
student@localhost:~/bash_scripts> ./example_1.sh
-6
student@localhost:~/bash_scripts> ./example_1.sh
-6
student@localhost:~/bash_scripts> ./example_1.sh
6
student@localhost:~/bash_scripts> ./example_1.sh
8
student@localhost:~/bash_scripts> ./example_1.sh
4
student@localhost:~/bash_scripts> ./example_1.sh
-8
student@localhost:~/bash_scripts> ./example_1.sh
10
student@localhost:~/bash_scripts>
```

Getting random values

Or I could take it in two steps:



```
dog=$(( $RANDOM%11-5 ))
dog=$(($dog*2))
echo $dog
```



```
student@localhost:~/bash_scripts> ./example_1.sh
2
student@localhost:~/bash_scripts> ./example_1.sh
-2
student@localhost:~/bash_scripts> ./example_1.sh
8
student@localhost:~/bash_scripts> ./example_1.sh
-6
student@localhost:~/bash_scripts> ./example_1.sh
2
student@localhost:~/bash_scripts> ./example_1.sh
8
student@localhost:~/bash_scripts> _
```

More in Tutorial

I’ll cover a couple more concepts in the video tutorial this time so be sure to watch that and code along with me.

```
for i in {0..10..2}
  do
     echo "Welcome $i times"
 done
```

```
cat=15
for (( c=1; c<=$cat; c++ ))  or c--
do
   echo "Welcome $c times"
done
```

```
#int check
[[ $var =~ ^[+-]?[0-9] ]]
```

Nano Tips

Ctrl-6 to set mark

Alt-6 to end mark

Ctlr-U to paste

Alt U to undo

Alt E to redo

Hold shift, use arrows to highlight

- Then Alt ^ to copy
- Ctl-U paste