Introduction to BASH scripting

Day10_FirstBash.md

Recalling

LAST TIME TOPIC

Today's Topics

- what is bash,
- use of bash for hackers
- Output,
- variables & data type
- Input
- comment and indentations
- Arithmetic operation
- if-else

What is Bash Script?

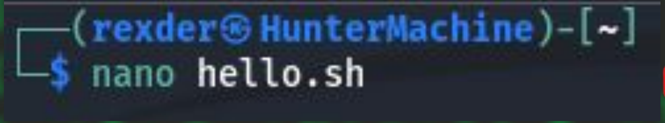
- Bash = Bourne Again Shell
- It is a shell, that used to interact with your kernel.
- What is Script?
    - Script is a file that contains shell commands in a simple and clear algorithm.
- The original is sh - Bourne shell

Uses of bash

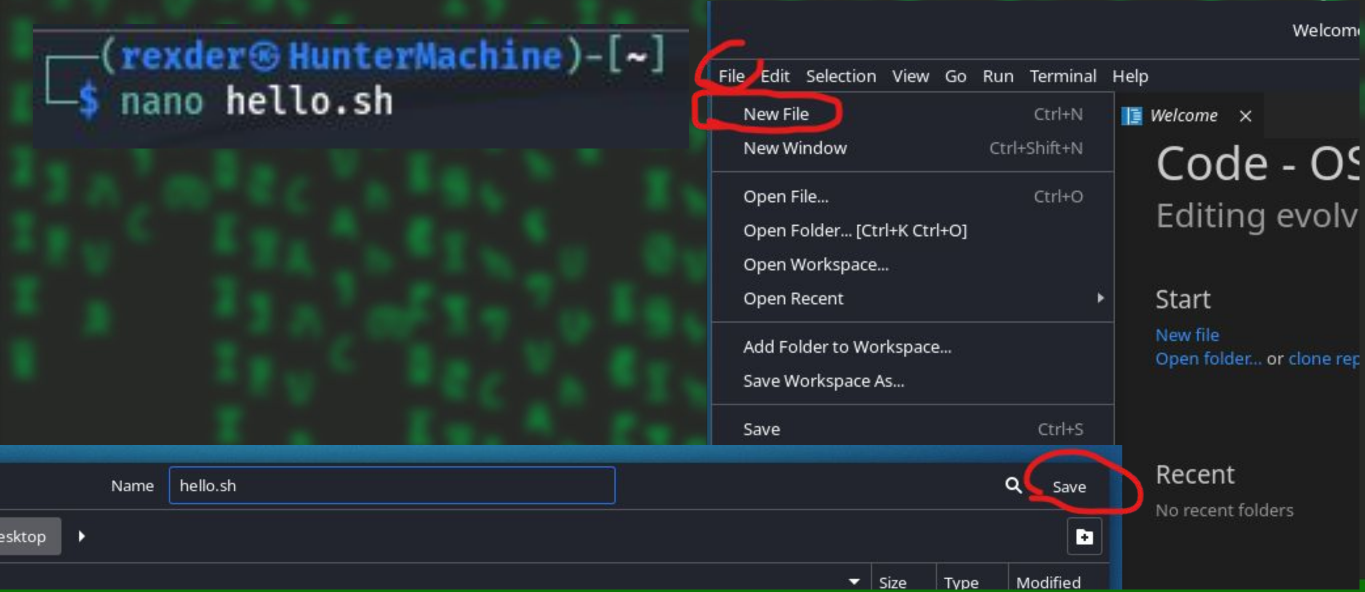
- Script development
- Automating tasks
- Simplifying your linux ability
- Developing hacking scripts.

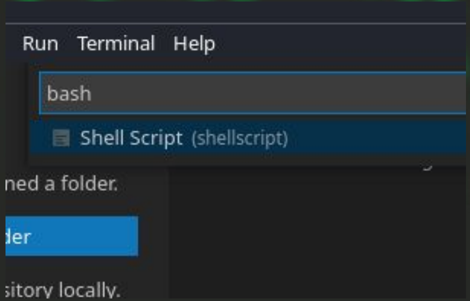
Starting with Bash

- Bash files can have ".sh" extension but you can have it without '.sh' too
- The file have to have executable Permissions.
- You can use any text editors u need: VIM,nano,VScode,gedit,cherrytree...

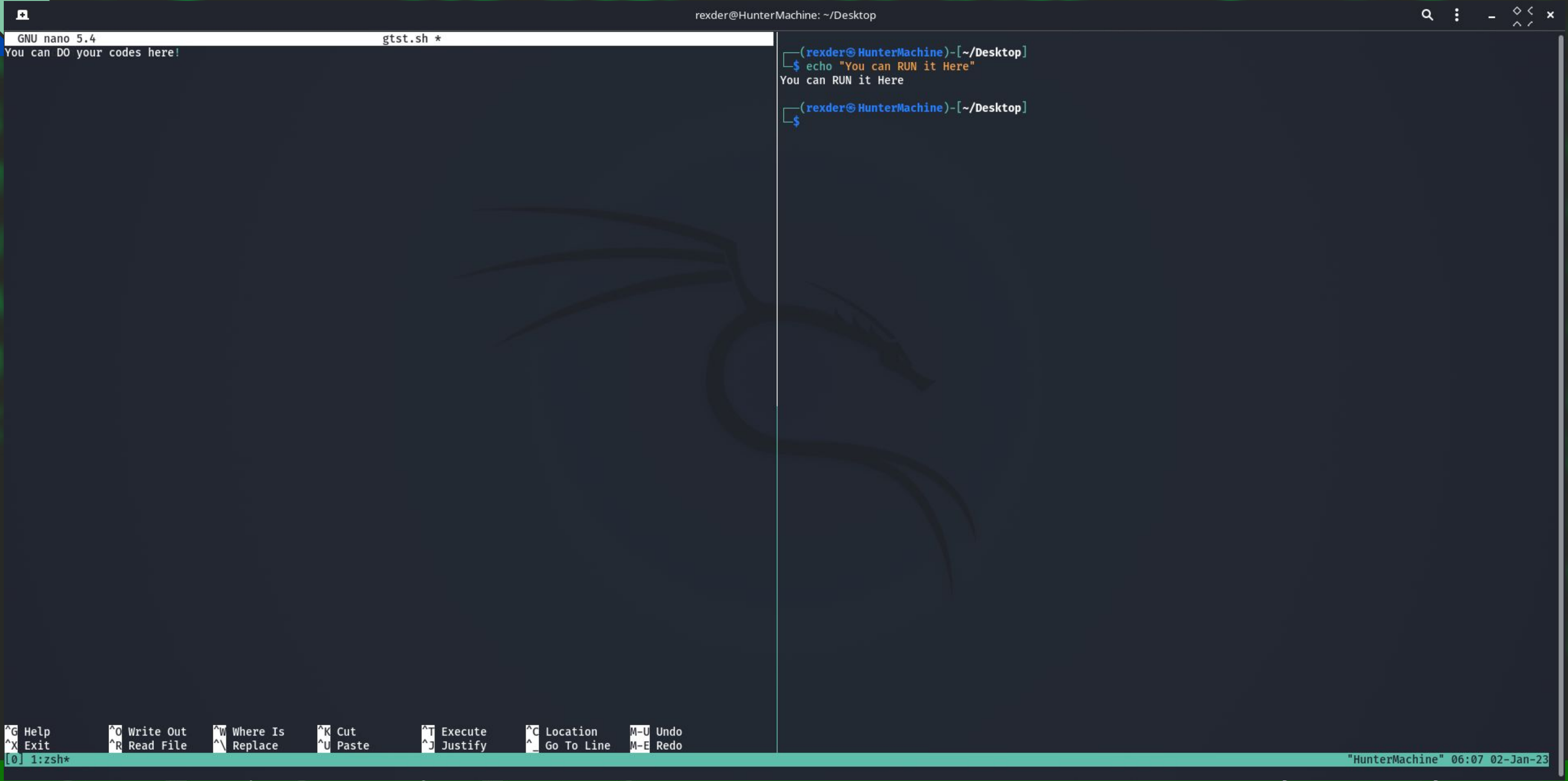


```
┌──(rexder㉿HunterMachine)-[~]
└─$ nano hello.sh
```





Tmux can help



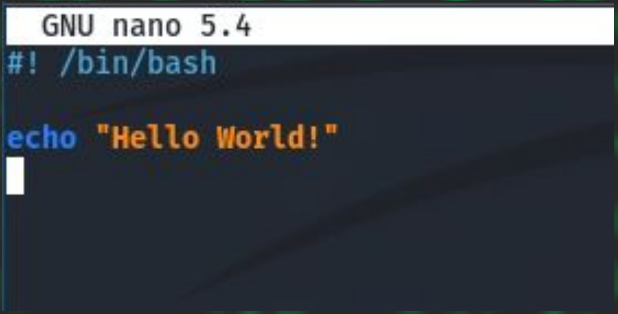
```
rexder@HunterMachine: ~/Desktop

  GNU nano 5.4                      gtst.sh *
You can DO your codes here!

^G Help      ^O Write Out  ^W Where Is  ^K Cut    ^T Execute  ^C Location    M-U Undo
^X Exit      ^R Read File  ^\ Replace   ^U Paste  ^J Justify  ^_ Go To Line  M-E Redo
[0] 1:zsh*                                          "HunterMachine" 06:07 02-Jan-23
```

```
┌──(rexder㉿HunterMachine)-[~/Desktop]
└─$ echo "You can RUN it Here"
You can RUN it Here

┌──(rexder㉿HunterMachine)-[~/Desktop]
└─$
```

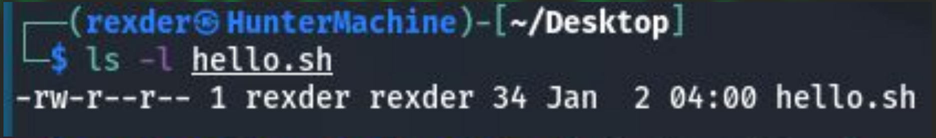
Displaying output

Shebang's used to tell the shell which interpreter is used to execute the script

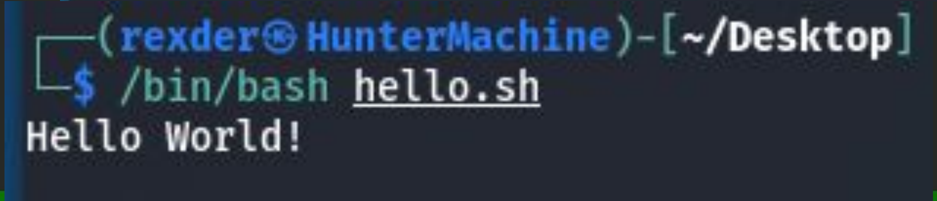
- To start Every bash scripts use shebang.
  - #! /bin/bash
  - #! /bin/sh
- To display outputs on bash you just do
  - echo "YOUR TEXT HERE"
- To run your project you can do:
  - /bin/bash hello.sh
  - ./hello.sh -> need x
  - hello -> need x



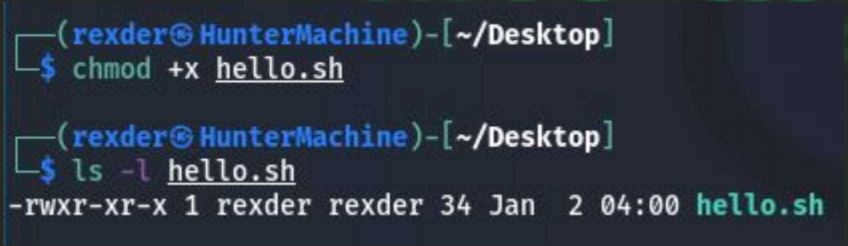
```
  GNU nano 5.4
#! /bin/bash

echo "Hello World!"
```



```
┌──(rexder㉿HunterMachine)-[~/Desktop]
└─$ ls -l hello.sh
-rw-r--r-- 1 rexder rexder 34 Jan  2 04:00 hello.sh
```

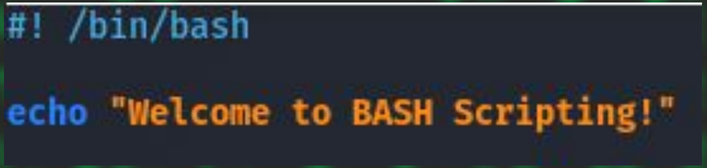


```
┌──(rexder㉿HunterMachine)-[~/Desktop]
└─$ /bin/bash hello.sh
Hello World!
```

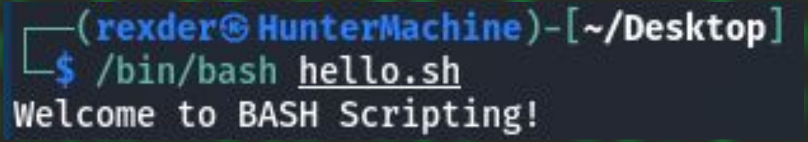


```
┌──(rexder㉿HunterMachine)-[~/Desktop]
└─$ chmod +x hello.sh

┌──(rexder㉿HunterMachine)-[~/Desktop]
└─$ ls -l hello.sh
-rwxr-xr-x 1 rexder rexder 34 Jan  2 04:00 hello.sh
```

Examples

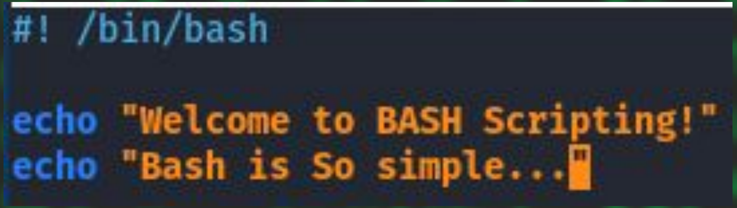


```
#! /bin/bash

echo "Welcome to BASH Scripting!"
```

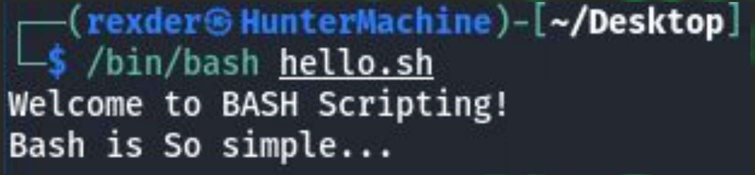


```
┌──(rexder㉿HunterMachine)-[~/Desktop]
└─$ /bin/bash hello.sh
Welcome to BASH Scripting!
```

If you need to add new lines on your code just add echo



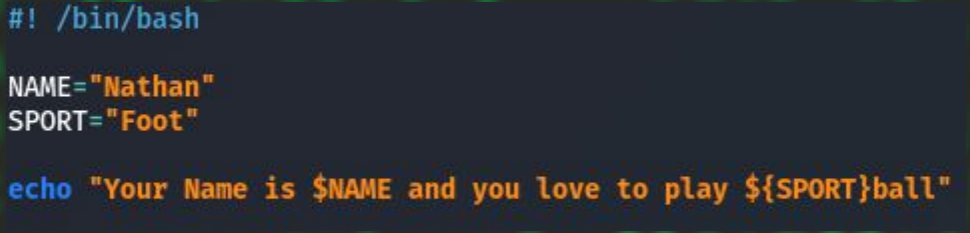
```
#! /bin/bash

echo "Welcome to BASH Scripting!"
echo "Bash is So simple..."
```



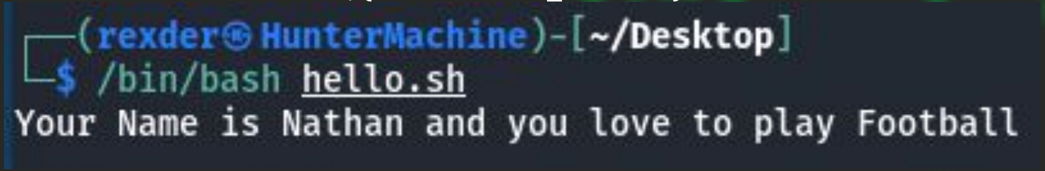
```
┌──(rexder㉿HunterMachine)-[~/Desktop]
└─$ /bin/bash hello.sh
Welcome to BASH Scripting!
Bash is So simple...
```

Variables

- Bash Variables are same with python variables, with some exceptions.
- Syntax:
  - VARIABLE_NAME=value
- Exceptions:
  - NO Space between the equal sign (=)
    - NAME = “Nathan”   => ERROR
    - NAME=”Nathan” => Correct.
    - Never Start with Numbers
    - USE double quotes only.
- To use the variable we will use dollar sign($) before the Variable name
- If you want to display the variable sticked with other text use ${VARIABLE_NAME}
- Bash Variables are string by default.

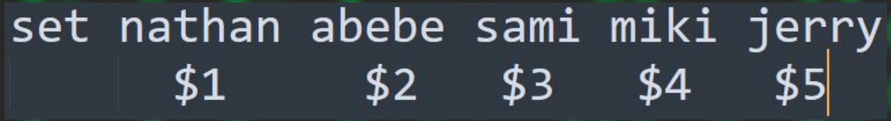


```bash
#! /bin/bash

NAME="Nathan"
SPORT="Foot"

echo "Your Name is $NAME and you love to play ${SPORT}ball"
```



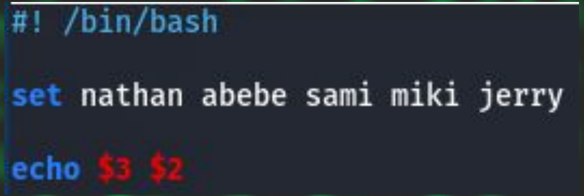
```
┌──(rexder㉿HunterMachine)-[~/Desktop]
└─$ /bin/bash hello.sh
Your Name is Nathan and you love to play Football
```

Cont...

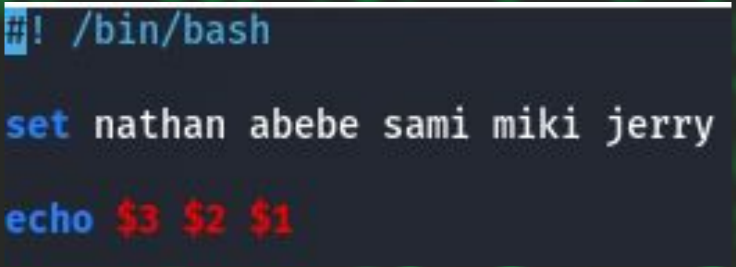


```
set nathan abebe sami miki jerry
     $1      $2    $3   $4   $5
```

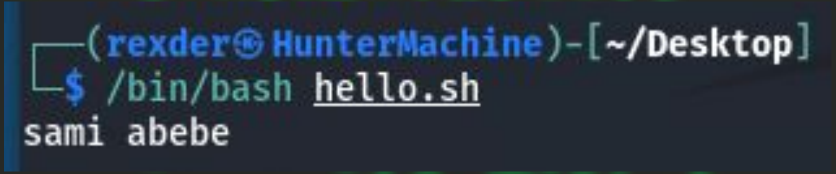
- The set command can be used to assign values to positional parameters.
- Syntax:
    - set value1 value2 value3 value4 value5



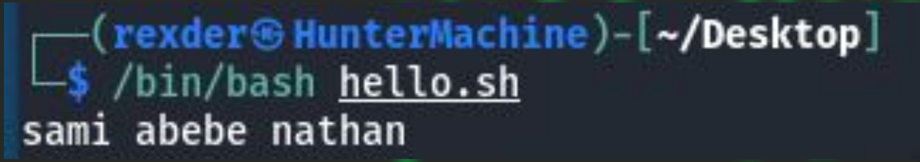
```
#! /bin/bash

set nathan abebe sami miki jerry

echo $3 $2
```



```
#! /bin/bash

set nathan abebe sami miki jerry

echo $3 $2 $1
```



```
┌──(rexder㉿HunterMachine)-[~/Desktop]
└─$ /bin/bash hello.sh
sami abebe
```



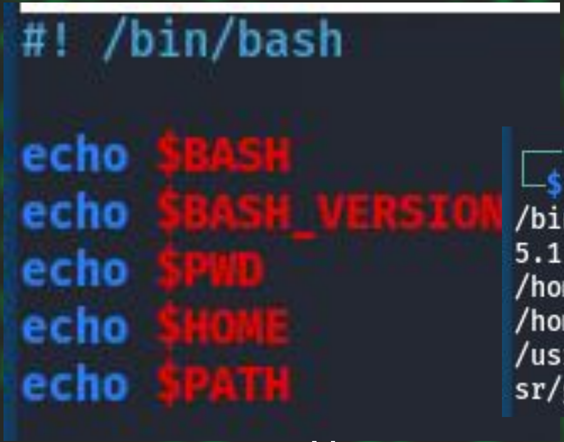
```
┌──(rexder㉿HunterMachine)-[~/Desktop]
└─$ /bin/bash hello.sh
sami abebe nathan
```

Exercise 1 10min

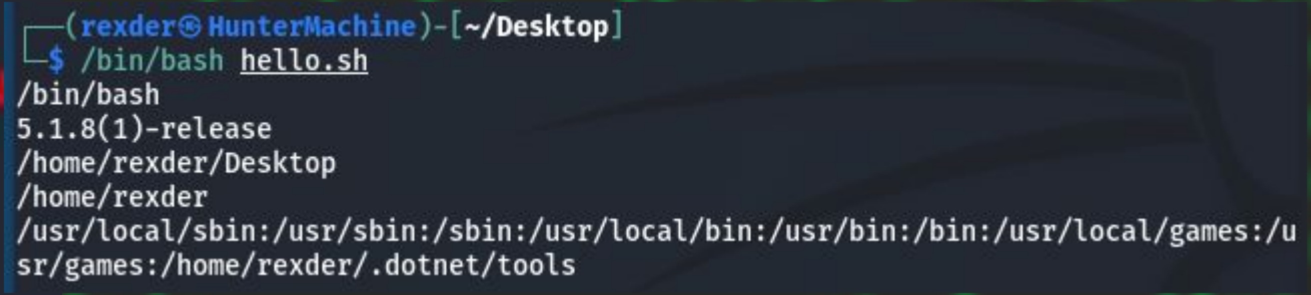
1) Display, “This is Day10,” and on new line “Introduction to Bash Scripting”
2) Create a Variable Called fname and lname with value of your first name and last name ,
   a) DIsplay: “ (Your name) is firstname and your Father name is (your father name).:
3) Create a Variable called Num and give it value of 10 and
   a) Display, “This is Day10 class” the num 10 from the variable

System Variables

- Are variables those are declared by the system.



```
#! /bin/bash

echo $BASH
echo $BASH_VERSION
echo $PWD
echo $HOME
echo $PATH
```



```
┌──(rexder㉿HunterMachine)-[~/Desktop]
└─$ /bin/bash hello.sh
/bin/bash
5.1.8(1)-release
/home/rexder/Desktop
/home/rexder
/usr/local/sbin:/usr/sbin:/sbin:/usr/local/bin:/usr/bin:/bin:/usr/local/games:/u
sr/games:/home/rexder/.dotnet/tools
```

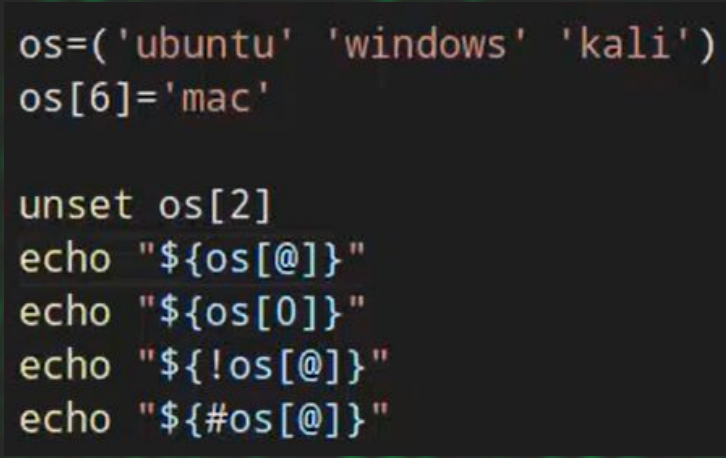
- There are so many: LANG, TERM,MAIL,EDITOR,USER,SHELL….
- USER displays Computer owner(host)

Variables & Data Types

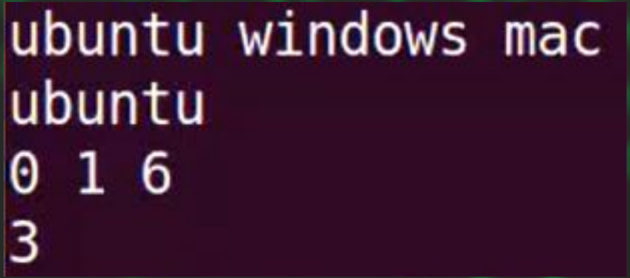
- As we saw, the previous method they create strings only.
- So to create other data types we use declare.
- **<u>Arrays</u>**
  a) Arrays are lists or tuples on python.
  b) Syntax:
     i) var=("list1" "list2" "list3" "list4)
     ii) TO display echo
        (1) ${var[0]}
     iii) To get all the elements
        (1) ${var[@]}

cont...

i) To get the indexes
   (1) ${!var[@]}
ii) To get the length
   (1) ${#var[@]}
iii) To add element to the array
   (1) var[4]="list5"
iv) To remove from the array
   (1) unset var[3]



```
os=('ubuntu' 'windows' 'kali')
os[6]='mac'

unset os[2]
echo "${os[@]}"
echo "${os[0]}"
echo "${!os[@]}"
echo "${#os[@]}"
```



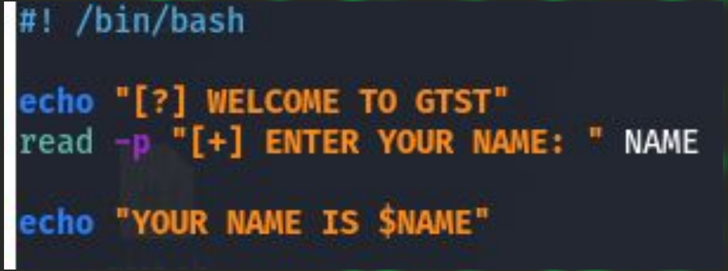
```
ubuntu windows mac
ubuntu
0 1 6
3
```

Bash Input

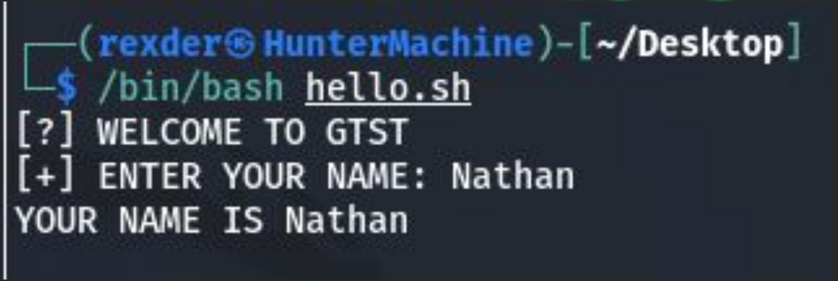
- On bash we have 2 methods to accept input
1. Read function
2. Arguments

1) Bash read

- Read used to accept inputs while the script is running.
- Syntax:
  - read -p "Text To Display" var
  - read -sp "Password: " var => used to accept hidden texts like password.
  - read -a var => for accepting arrays(lists)

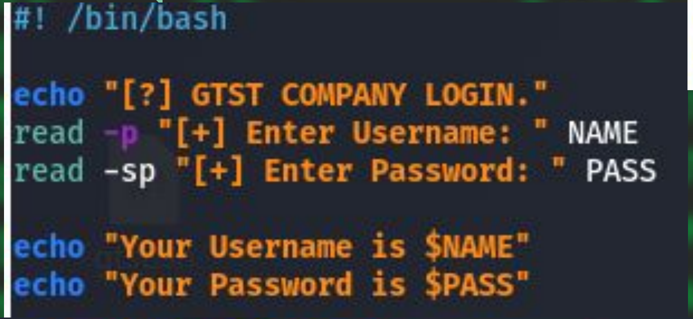


```
#! /bin/bash

echo "[?] WELCOME TO GTST"
read -p "[+] ENTER YOUR NAME: " NAME

echo "YOUR NAME IS $NAME"
```

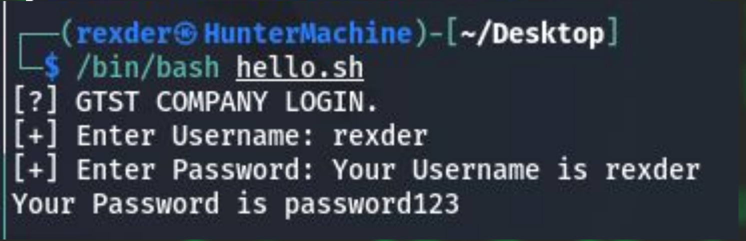


```
┌──(rexder㉿HunterMachine)-[~/Desktop]
└─$ /bin/bash hello.sh
[?] WELCOME TO GTST
[+] ENTER YOUR NAME: Nathan
YOUR NAME IS Nathan
```

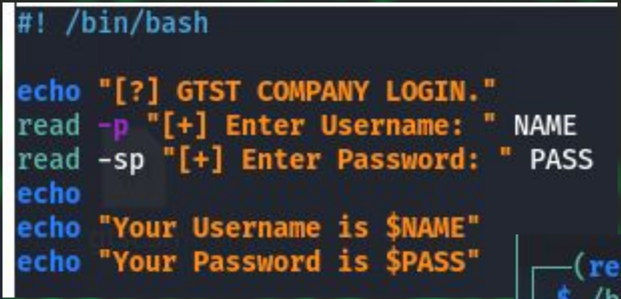
Cont...



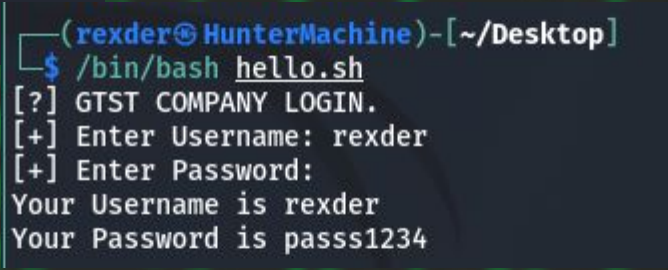
```
#! /bin/bash

echo "[?] GTST COMPANY LOGIN."
read -p "[+] Enter Username: " NAME
read -sp "[+] Enter Password: " PASS

echo "Your Username is $NAME"
echo "Your Password is $PASS"
```



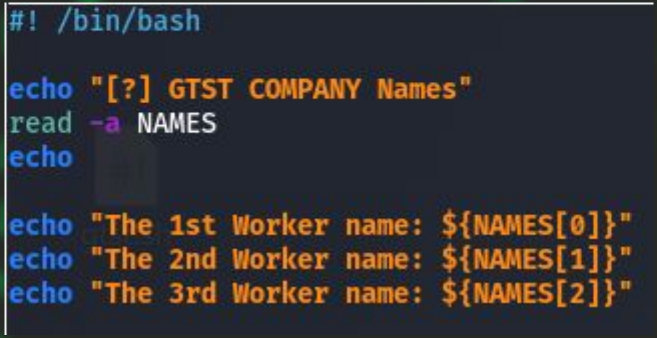
```
┌──(rexder㉿HunterMachine)-[~/Desktop]
└─$ /bin/bash hello.sh
[?] GTST COMPANY LOGIN.
[+] Enter Username: rexder
[+] Enter Password: Your Username is rexder
Your Password is password123
```



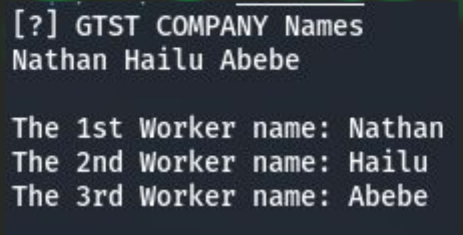
```
#! /bin/bash

echo "[?] GTST COMPANY LOGIN."
read -p "[+] Enter Username: " NAME
read -sp "[+] Enter Password: " PASS
echo
echo "Your Username is $NAME"
echo "Your Password is $PASS"
```



```
┌──(rexder㉿HunterMachine)-[~/Desktop]
└─$ /bin/bash hello.sh
[?] GTST COMPANY LOGIN.
[+] Enter Username: rexder
[+] Enter Password:
Your Username is rexder
Your Password is passs1234
```



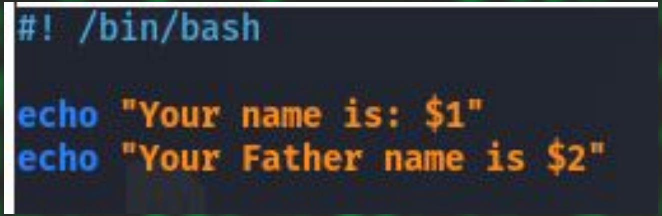
```
#! /bin/bash

echo "[?] GTST COMPANY Names"
read -a NAMES
echo

echo "The 1st Worker name: ${NAMES[0]}"
echo "The 2nd Worker name: ${NAMES[1]}"
echo "The 3rd Worker name: ${NAMES[2]}"
```



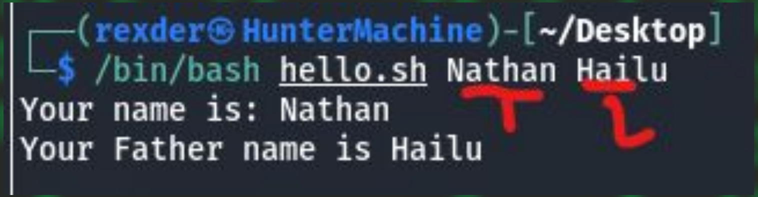
```
[?] GTST COMPANY Names
Nathan Hailu Abebe

The 1st Worker name: Nathan
The 2nd Worker name: Hailu
The 3rd Worker name: Abebe
```

2) Arguments

- These helps to get input before the script starts
- Syntax:
  - Just use $0-$9 while you want to work with the input



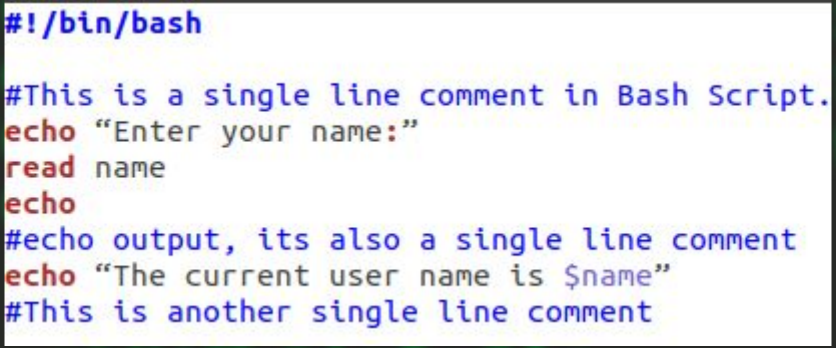
```
#! /bin/bash

echo "Your name is: $1"
echo "Your Father name is $2"
```



```
┌──(rexder㉿HunterMachine)-[~/Desktop]
└─$ /bin/bash hello.sh Nathan Hailu
Your name is: Nathan
Your Father name is Hailu
```

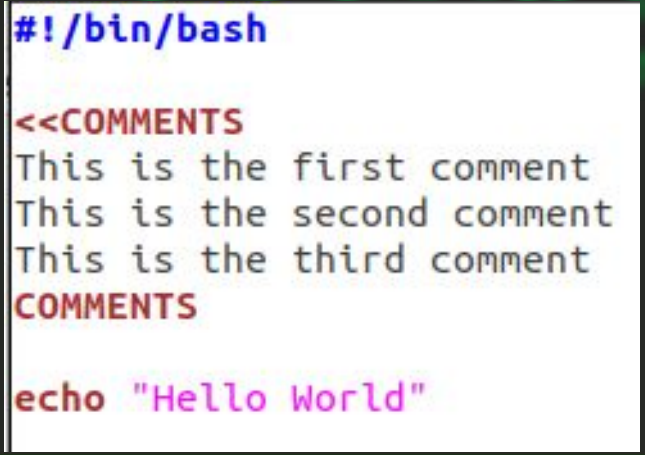
Comments

On bash the comments are



```
#!/bin/bash

#This is a single line comment in Bash Script.
echo “Enter your name:”
read name
echo
#echo output, its also a single line comment
echo “The current user name is $name”
#This is another single line comment
```

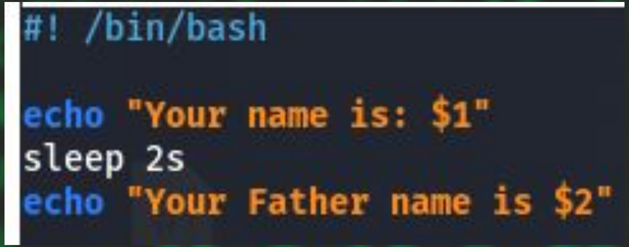
For multi line comment we start with
<<COMMENTS
Sfasf
Sfa
COMMENTS , we close with this



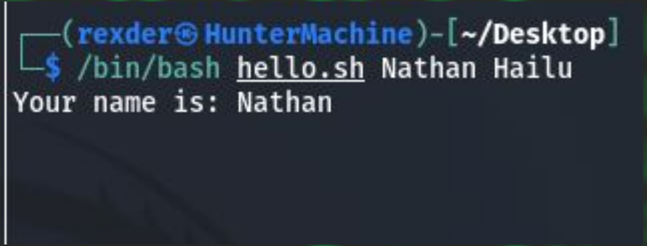
```
#!/bin/bash

<<COMMENTS
This is the first comment
This is the second comment
This is the third comment
COMMENTS

echo "Hello World"
```

Bash sleep

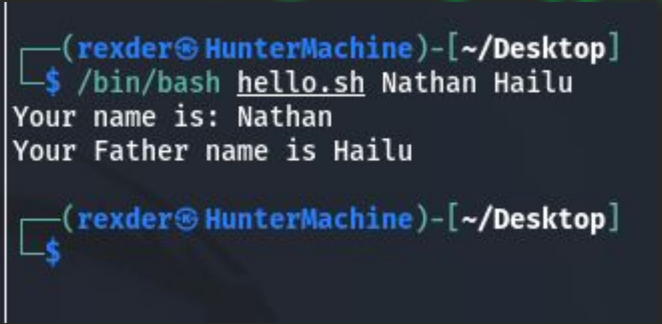
- Sleep used to make a good waiting on our script.
- Syntax:
  - sleep <number>s



```
#! /bin/bash

echo "Your name is: $1"
sleep 2s
echo "Your Father name is $2"
```



```
┌──(rexder㉿HunterMachine)-[~/Desktop]
└─$ /bin/bash hello.sh Nathan Hailu
Your name is: Nathan
```



```
┌──(rexder㉿HunterMachine)-[~/Desktop]
└─$ /bin/bash hello.sh Nathan Hailu
Your name is: Nathan
Your Father name is Hailu

┌──(rexder㉿HunterMachine)-[~/Desktop]
└─$
```

Exercise 2 10 min

1) Create an Array of fruits with element of apple,banana,pineapple,mango
    a) Display "I love apple banana pineapple mango"
2) Accept a normal username and hidden secret key input then display The Computer Owner name.
    a) The user after inserting username wait for 1 second.
    b) HINT: to get computer owner name use SYSTEM variables
3) Accept 2 values from the user before the script start, your name and age
    a) Display: "Your name is ___ you are __ years old."

Operation

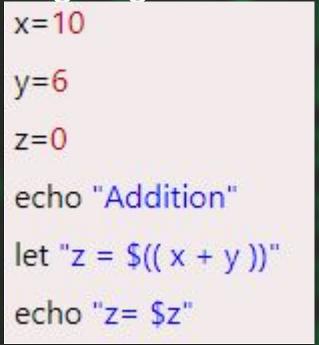
- To do mathematical operations you have to do $((expression))
- we will use let keyword for assigning variable

A) Arithmetic Operations

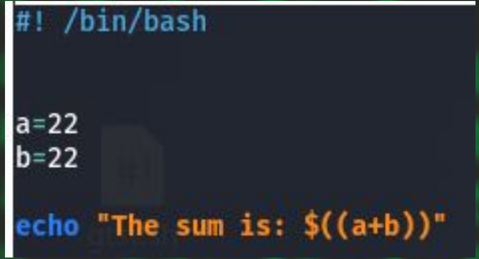
a) Addition $((a + b))
b) Subtraction $((a - b))
c) Multiplication $((a * b))
d) Division $((a / b))
e) Exponential $((a ** b))
f) Modulo $((a % b))

B) Assignment Operations

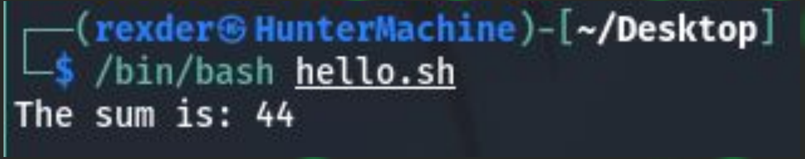
a) Increment “let a+= 3”
b) Decrement “ let a-= 3”
c) Multiply “ let a*= 3 “
d) Divide “ let a/=3 “



```
x=10

y=6

z=0

echo "Addition"

let "z = $(( x + y ))"

echo "z= $z"
```



```
#! /bin/bash


a=22
b=22

echo "The sum is: $((a+b))"
```



```
┌──(rexder㉿HunterMachine)-[~/Desktop]
└─$ /bin/bash hello.sh
The sum is: 44
```

Cont...

C) Comparison operation

Alphabetic comparison

- Greater Than => -gt
- Less Than => -lt
- Greater than and equals to => -ge
- Less than and equals too => -le
- Equal => -eq
- Not equal => -ne

Sign comparison

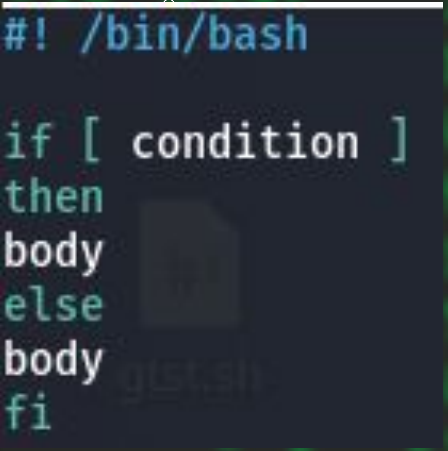
- >
- <
- >=
- <=
- =
- !=

Exercise 3 3min

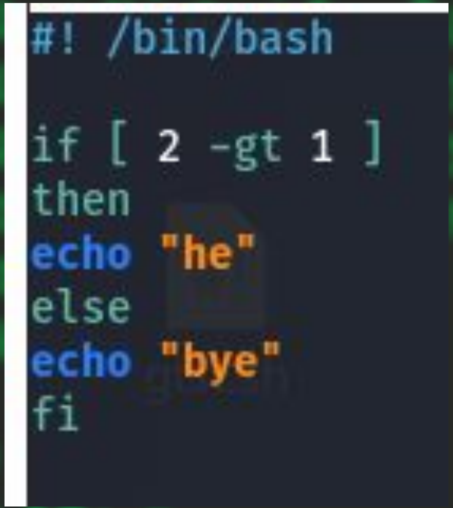
1) Create 2 variables and do the 5 mathematical operations

If else conditions

- Syntax:



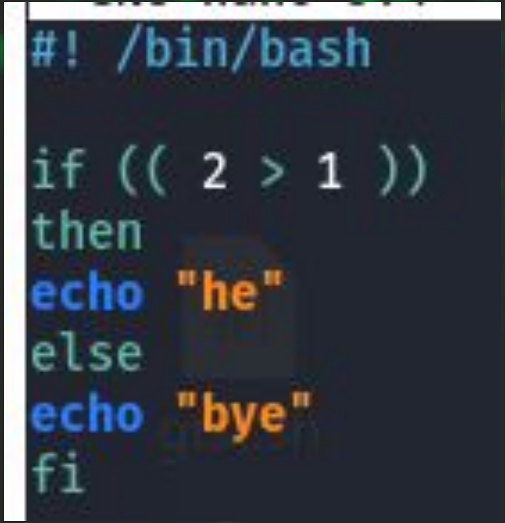
```
#! /bin/bash

if [ condition ]
then
body
else
body
fi
```



```
#! /bin/bash

if [ 2 -gt 1 ]
then
echo "he"
else
echo "bye"
fi
```

- If you used [condition] => you will use alphabetic comparison
- But for strings you can use sign too

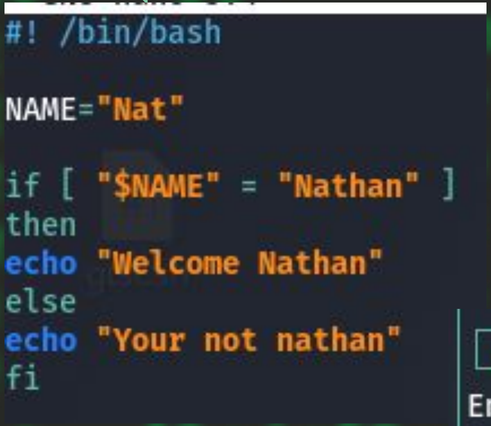
On bash we don't have indentation but if u finished writing the body you type "fi"



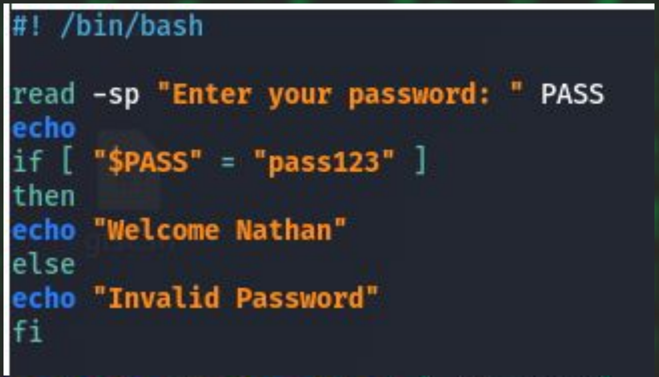
```
#! /bin/bash

if (( 2 > 1 ))
then
echo "he"
else
echo "bye"
fi
```

If you used ((condition)) => you will use numeric comparison

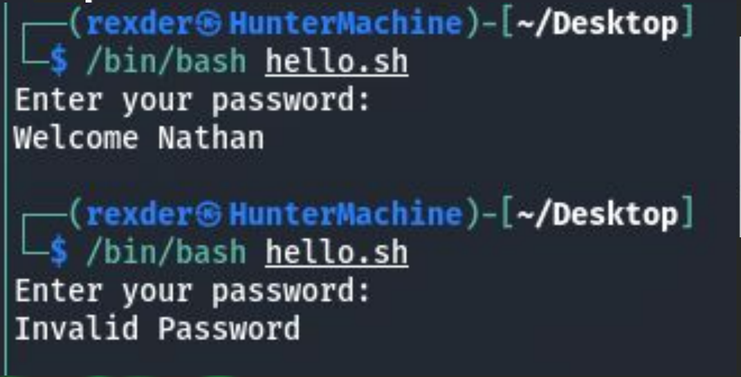
Examples



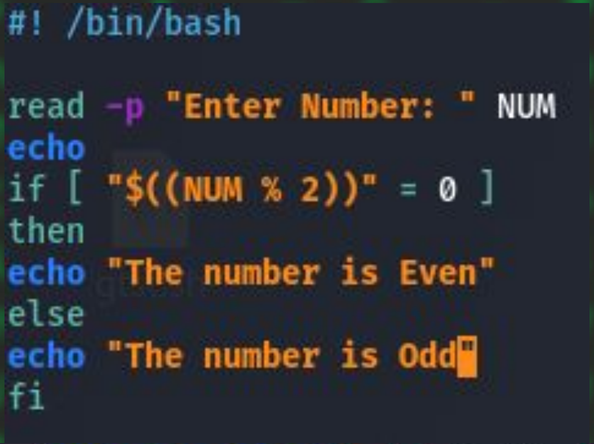
```
#! /bin/bash

NAME="Nat"

if [ "$NAME" = "Nathan" ]
then
echo "Welcome Nathan"
else
echo "Your not nathan"
fi
```



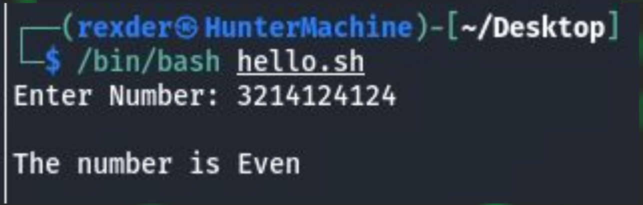
```
#! /bin/bash

read -sp "Enter your password: " PASS
echo
if [ "$PASS" = "pass123" ]
then
echo "Welcome Nathan"
else
echo "Invalid Password"
fi
```



```
┌──(rexder㉿HunterMachine)-[~/Desktop]
└─$ /bin/bash hello.sh
Enter your password:
Welcome Nathan

┌──(rexder㉿HunterMachine)-[~/Desktop]
└─$ /bin/bash hello.sh
Enter your password:
Invalid Password
```



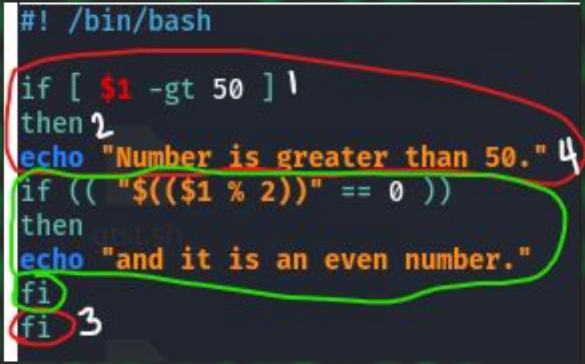
```
#! /bin/bash

read -p "Enter Number: " NUM
echo
if [ "$((NUM % 2))" = 0 ]
then
echo "The number is Even"
else
echo "The number is Odd"
fi
```



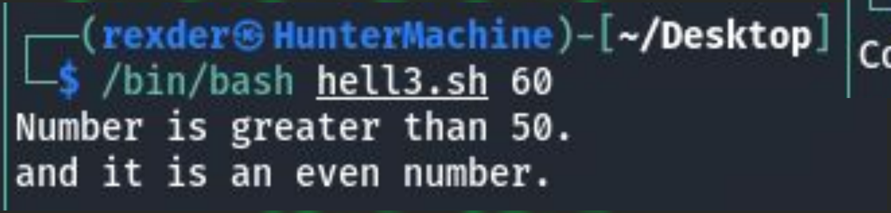
```
┌──(rexder㉿HunterMachine)-[~/Desktop]
└─$ /bin/bash hello.sh
Enter Number: 3214124124

The number is Even
```

Cont...

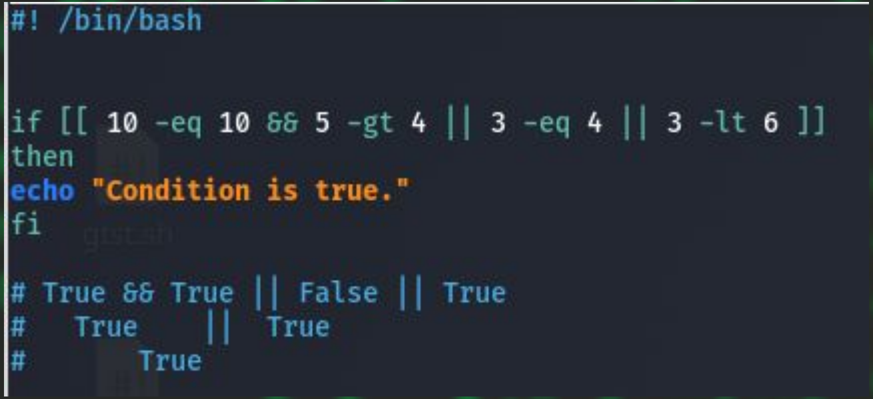
Nested if



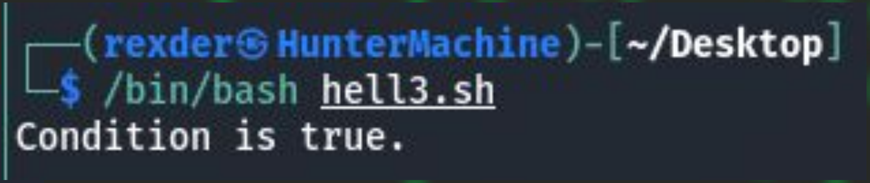
```
#! /bin/bash

if [ $1 -gt 50 ] 1
then 2
echo "Number is greater than 50." 4
if (( "$(($1 % 2))" == 0 ))
then
echo "and it is an even number."
fi
fi 3
```



```
┌──(rexder㉿HunterMachine)-[~/Desktop]
└─$ /bin/bash hell3.sh 60
Number is greater than 50.
and it is an even number.
```



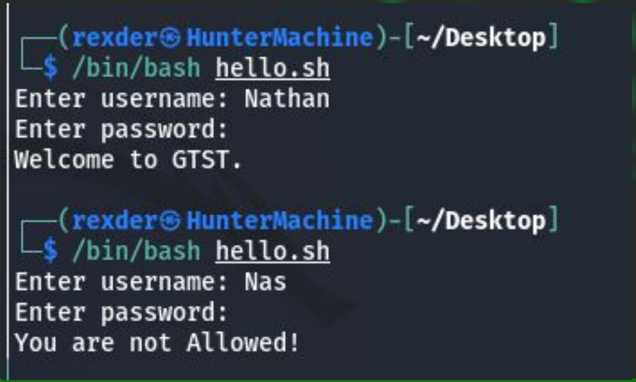
```
#! /bin/bash


if [[ 10 -eq 10 && 5 -gt 4 || 3 -eq 4 || 3 -lt 6 ]]
then
echo "Condition is true."
fi

# True && True || False || True
#   True      ||   True
#         True
```



```
┌──(rexder㉿HunterMachine)-[~/Desktop]
└─$ /bin/bash hell3.sh
Condition is true.
```

Exercise 4

1) Write a script that accepts input and validate that the number is 2212
   a) Display: on true “The number is matched!” on false: “Invalid!”
2) Accept 2 values, then check that if the 2 values added and if their sum is 10
   a) Display: on true “You are master!” on false “Your poor on maths…”
3) Accept username and password, if username is “gtst” and password is 1234 display “Welcome to GTST” other wise “You are not allowed!”



```
┌──(rexder㉿HunterMachine)-[~/Desktop]
└─$ /bin/bash hello.sh
Enter username: Nathan
Enter password:
Welcome to GTST.

┌──(rexder㉿HunterMachine)-[~/Desktop]
└─$ /bin/bash hello.sh
Enter username: Nas
Enter password:
You are not Allowed!
```

…

Recap!

Class is over

1) Any questions
2) Do your notes
3) Practice them