OS Laboratory
Exercise 9

Mostafa Fazli - 9822803

14 December 2021

What's Bash Script

A Bash script is a plain text file which contains a series of commands. These commands are a mixture of commands we would normally type ouselves on the command line (such as ls or cp for example) and commands we could type on the command line but generally wouldn't (you'll discover these over the next few pages). An important point to remember though is:

Anything you can run normally on the command line can be put into a script and it will do exactly the same thing. Similarly, anything you can put into a script can also be run normally on the command line and it will do exactly the same thing.

It is convention to give files that are Bash scripts an extension of .sh (myscript.sh for example). As you would be aware (and if you're not maybe you should consider reviewing our Linux Tutorial), Linux is an extensionless system so a script doesn't necessarily have to have this characteristic in order to work.

1

Bash expr

EXPR

The expr command in Unix evaluates a given expression and displays its corresponding output. It is used for:

Basic operations like addition, subtraction, multiplication, division, and modulus on integers.

Evaluating regular expressions, string operations like substring, length of strings etc.

Syntax expr

$expr expression

```
$expr 12 + 8
```

2

IF

IF

A basic if statement effectively says, if a particular test is true, then perform a given set of actions. If it is not true then don't perform those actions.

Anything between then and fi (if backwards) will be executed only if the test (between the square brackets) is true.

Syntax if

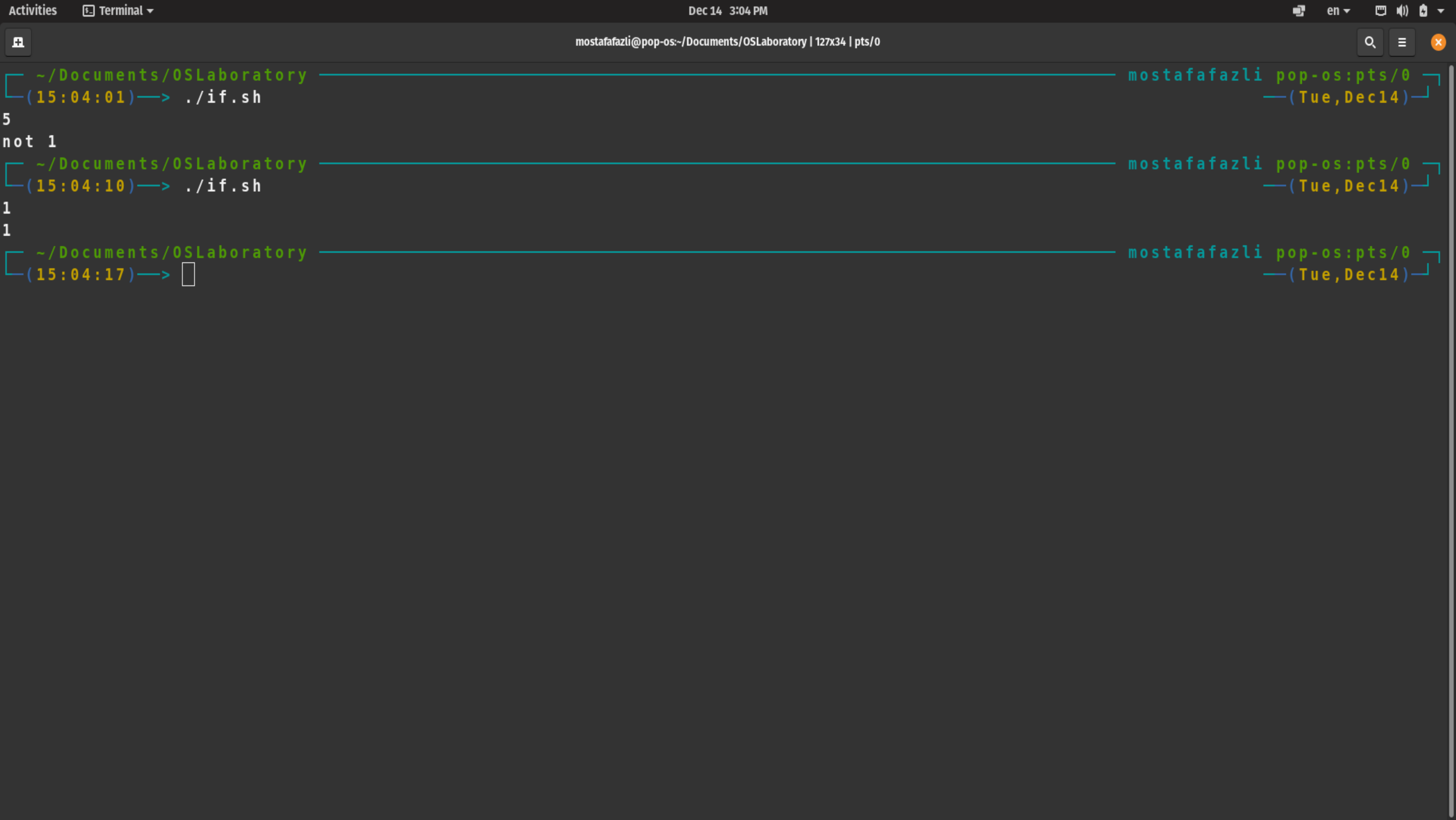
```
if [ <some test> ]
then
<commands>
fi
```

```
#!/bin/bash
if [ $1 -gt 100 ]
then
echo Hey that\'s a large number.
pwd
fi
```

Syntax if else

```
if [ <some test> ]
then
<commands>
else
<other commands>
fi
```

```
#!/bin/bash
if [ $# -eq 1 ]
then
        nl $1
else
        nl /dev/stdin
fi
```



```
mostafafazli@pop-os:~/Documents/OSLaboratory | 127x34 | pts/0

┌─ ~/Documents/OSLaboratory ──────────────────────────────── mostafafazli pop-os:pts/0 ─┐
└─(15:04:01)──> ./if.sh                                                  ──(Tue,Dec14)─┘
5
not 1
┌─ ~/Documents/OSLaboratory ──────────────────────────────── mostafafazli pop-os:pts/0 ─┐
└─(15:04:10)──> ./if.sh                                                  ──(Tue,Dec14)─┘
1
1
┌─ ~/Documents/OSLaboratory ──────────────────────────────── mostafafazli pop-os:pts/0 ─┐
└─(15:04:17)──>                                                          ──(Tue,Dec14)─┘
```

3

CASE

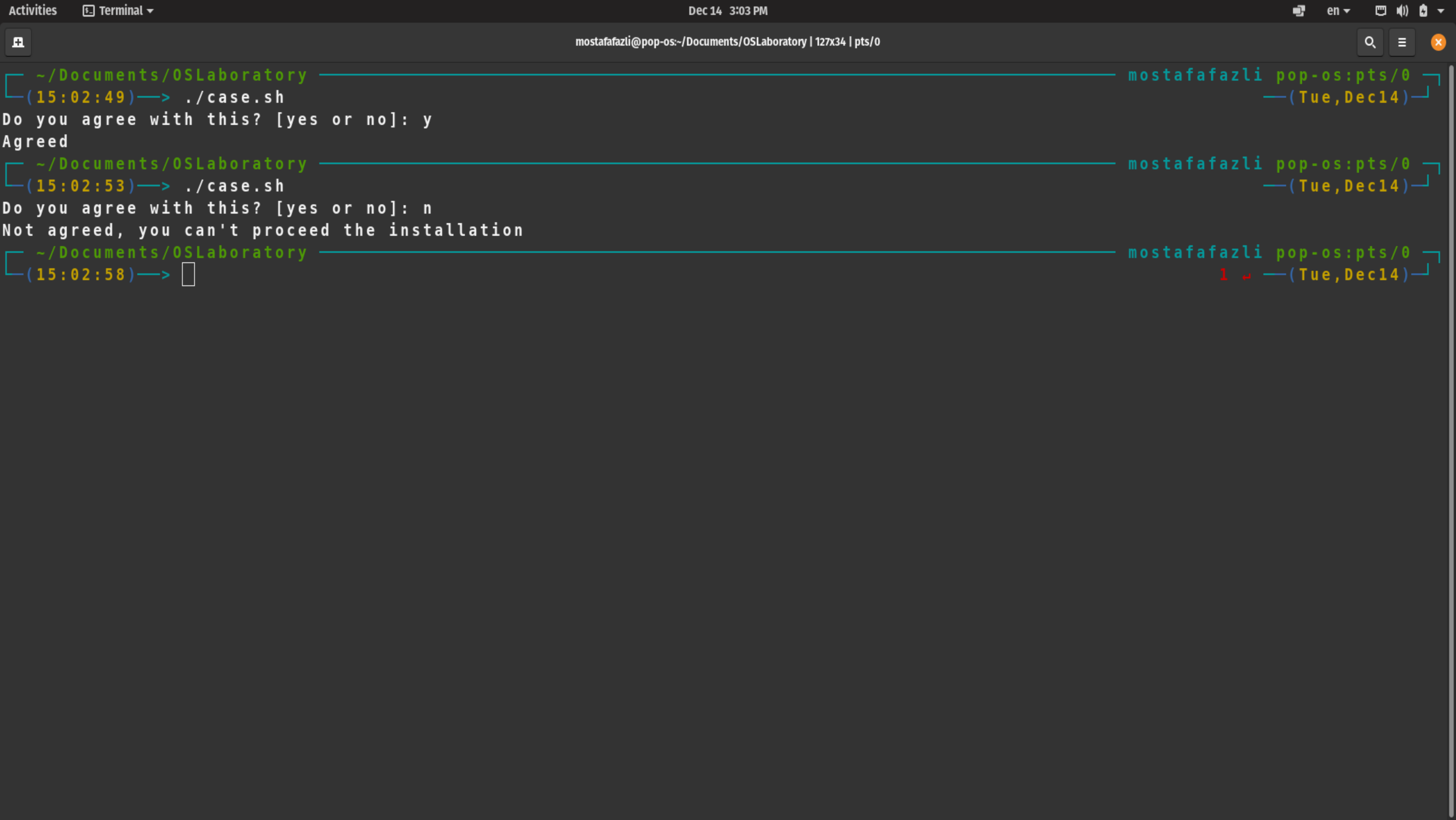
CASE

The bash case statement is generally used to simplify complex conditionals when you have multiple different choices. Using the case statement instead of nested if statements will help you make your bash scripts more readable and easier to maintain.

Syntax case

```
case EXPRESSION in
 PATTERN_1)
  STATEMENTS
  ;;
 PATTERN_N)
  STATEMENTS
  ;;
 *)
  STATEMENTS
  ;;
esac
```

```
#!/bin/sh
echo "Please talk to me ..."
while :
do
  read INPUT_STRING
  case $INPUT_STRING in
                hello)
                                echo "Hello yourself!"
                                ;;
                bye)
                                echo "See you again!"
                                break
                                ;;
                *)
                                echo "Sorry, I don't understand"
                                ;;
  esac
done
```



```
mostafafazli@pop-os:~/Documents/OSLaboratory | 127x34 | pts/0

┌─ ~/Documents/OSLaboratory ──────────────────────────────── mostafafazli pop-os:pts/0 ─┐
└─(15:02:49)──> ./case.sh                                                ──(Tue,Dec14)─┘
Do you agree with this? [yes or no]: y
Agreed
┌─ ~/Documents/OSLaboratory ──────────────────────────────── mostafafazli pop-os:pts/0 ─┐
└─(15:02:53)──> ./case.sh                                                ──(Tue,Dec14)─┘
Do you agree with this? [yes or no]: n
Not agreed, you can't proceed the installation
┌─ ~/Documents/OSLaboratory ──────────────────────────────── mostafafazli pop-os:pts/0 ─┐
└─(15:02:58)──>                                                    1 ↵ ──(Tue,Dec14)─┘
```

4

FOR

FOR

A ‘for loop’ is a bash programming language statement which allows code to be repeatedly executed. A for loop is classified as an iteration statement i.e. it is the repetition of a process within a bash script.

For syntax

```
for (( VARIABLE = 1 ; VARIABLE <= VARIABLE2 ; VARIABLE++))
do
        command1
        command2
        commandN
done
```

```
#!/bin/bash
for (( c=1; c<=5; c++ ))
do
   echo "Welcome $c times"
done
```

5

FORIN

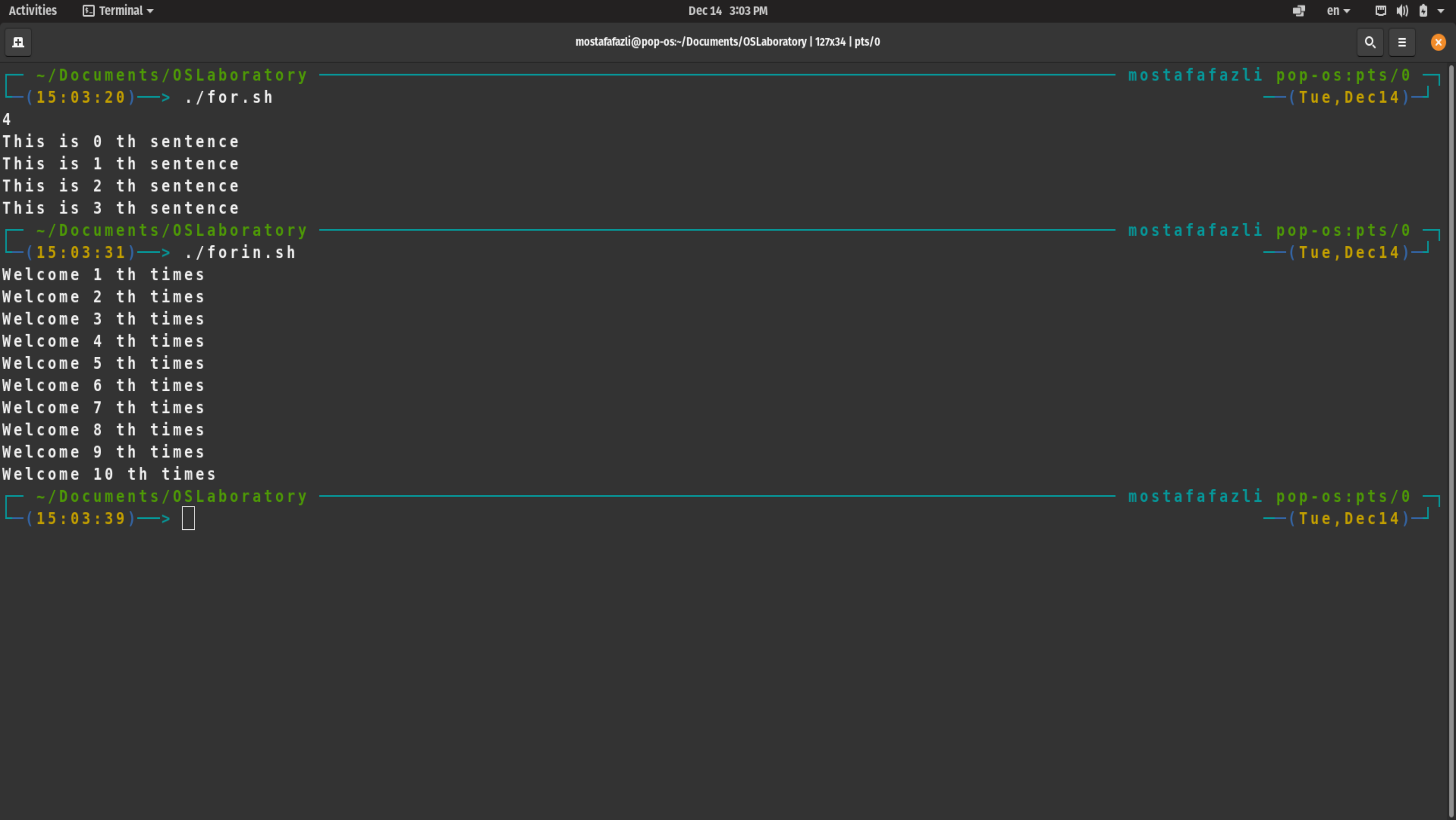
For in

For in like for but some different structure in syntax

ForIn syntax

```
for VARIABLE in 1 2 3 4 5 .. N
do
        command1
        command2
        commandN
done
```

```
#!/bin/bash
for i in 1 2 3 4 5
do
   echo "Welcome $i times"
done
```



```
Activities   Terminal ▾                     Dec 14  3:03 PM                          en ▾

            mostafafazli@pop-os:~/Documents/OSLaboratory | 127x34 | pts/0

┌─ ~/Documents/OSLaboratory ──────────────────────────────── mostafafazli pop-os:pts/0 ─┐
└─(15:03:20)──> ./for.sh                                                ──(Tue,Dec14)─┘
4
This is 0 th sentence
This is 1 th sentence
This is 2 th sentence
This is 3 th sentence
┌─ ~/Documents/OSLaboratory ──────────────────────────────── mostafafazli pop-os:pts/0 ─┐
└─(15:03:31)──> ./forin.sh                                              ──(Tue,Dec14)─┘
Welcome 1 th times
Welcome 2 th times
Welcome 3 th times
Welcome 4 th times
Welcome 5 th times
Welcome 6 th times
Welcome 7 th times
Welcome 8 th times
Welcome 9 th times
Welcome 10 th times
┌─ ~/Documents/OSLaboratory ──────────────────────────────── mostafafazli pop-os:pts/0 ─┐
└─(15:03:39)──>                                                         ──(Tue,Dec14)─┘
```

5

WHILE

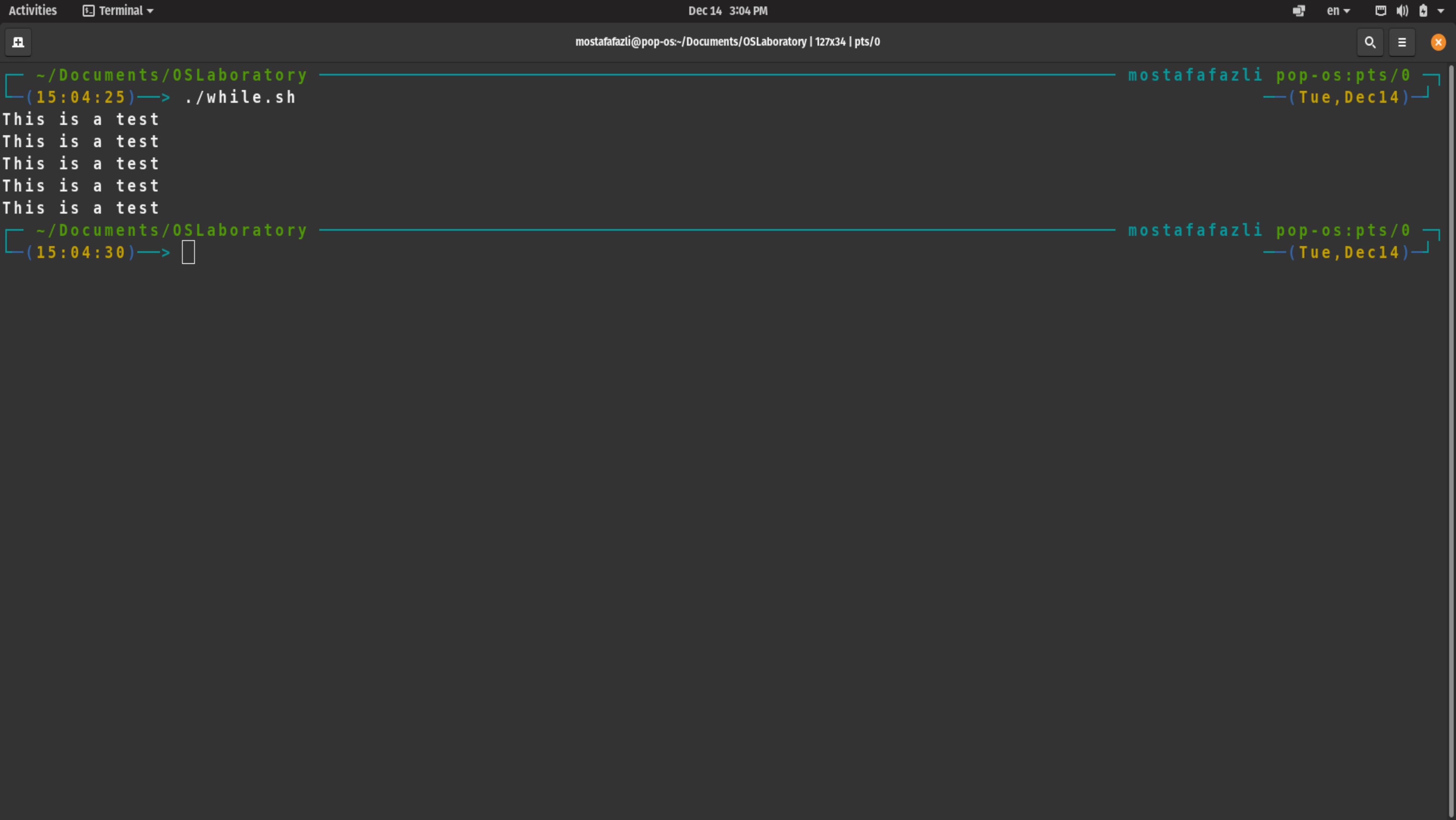
WHILE

While a different loop like for but with different syntax and structure

While syntax

```
while [ condition ]
do
  command1
  command2
  command3
done
```

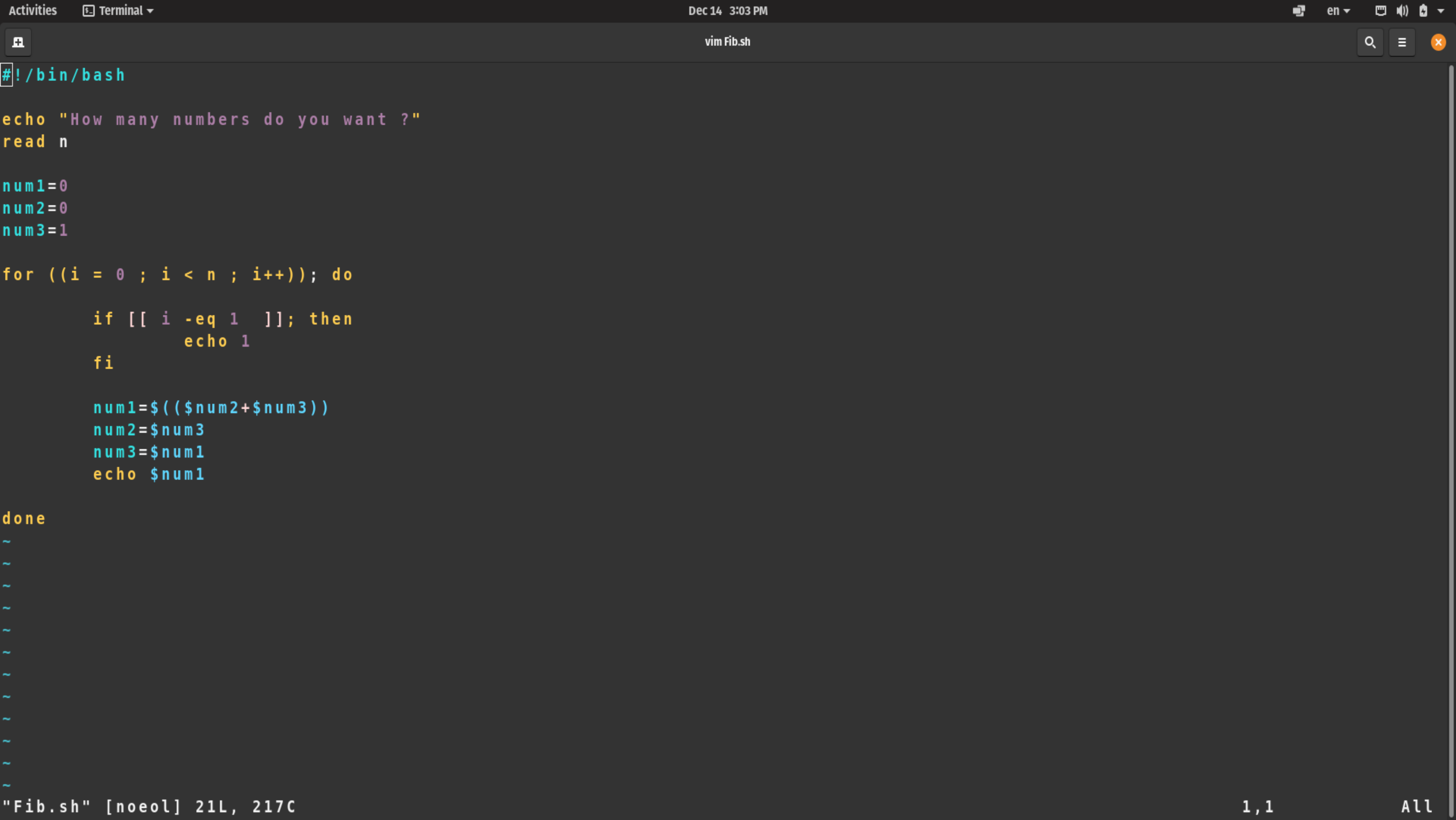
```
#!/bin/bash
x=1
while [ $x -le 5 ]
do
  echo "Welcome $x times"
  x=$(( $x + 1 ))
done
```


Activities
Terminal
Dec 14 3:04 PM
en
mostafafazli@pop-os:~/Documents/OSLaboratory | 127x34 | pts/0

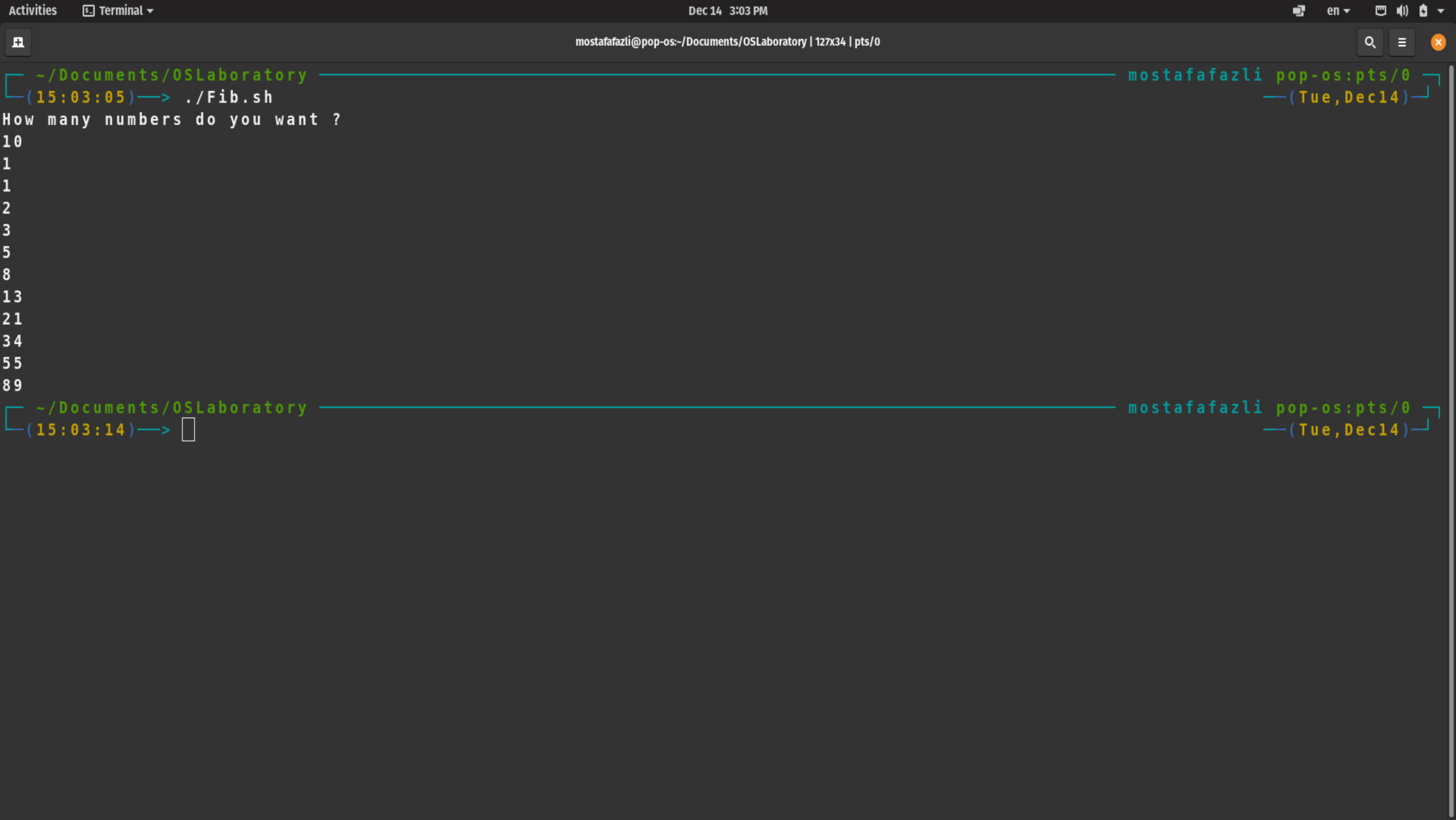
```
┌─ ~/Documents/OSLaboratory ──────────────────────────── mostafafazli pop-os:pts/0 ─┐
└─(15:04:25)──> ./while.sh                                           ──(Tue,Dec14)─┘
This is a test
This is a test
This is a test
This is a test
This is a test
┌─ ~/Documents/OSLaboratory ──────────────────────────── mostafafazli pop-os:pts/0 ─┐
└─(15:04:30)──>                                                      ──(Tue,Dec14)─┘
```

Exercise 9

This Exercise and Fibonacci funtions with together in a RAR file.



```
#!/bin/bash

echo "How many numbers do you want ?"
read n

num1=0
num2=0
num3=1

for ((i = 0 ; i < n ; i++)); do

        if [[ i -eq 1  ]]; then
                echo 1
        fi

        num1=$(($num2+$num3))
        num2=$num3
        num3=$num1
        echo $num1

done
~
~
~
~
~
~
~
~
~
~
~
~
"Fib.sh" [noeol] 21L, 217C                                                                    1,1           All
```



```
Activities   Terminal                    Dec 14  3:03 PM                                   en
                mostafafazli@pop-os:~/Documents/OSLaboratory | 127x34 | pts/0
┌─ ~/Documents/OSLaboratory ──────────────────────────── mostafafazli pop-os:pts/0 ─┐
└─(15:03:05)──> ./Fib.sh                                              ──(Tue,Dec14)─┘
How many numbers do you want ?
10
1
1
2
3
5
8
13
21
34
55
89
┌─ ~/Documents/OSLaboratory ──────────────────────────── mostafafazli pop-os:pts/0 ─┐
└─(15:03:14)──>                                                       ──(Tue,Dec14)─┘
```