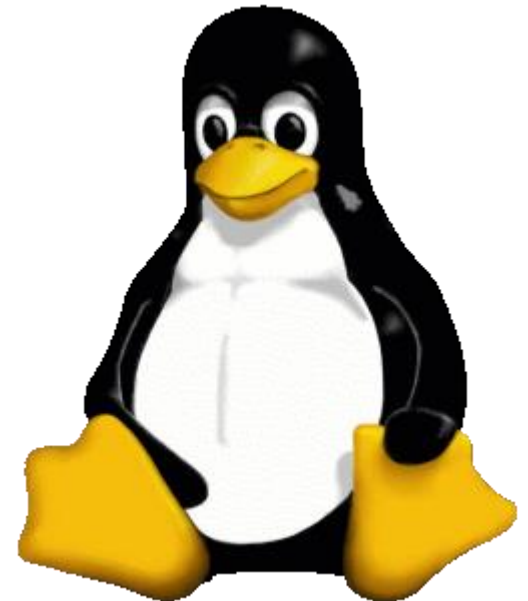


GNU/Linux

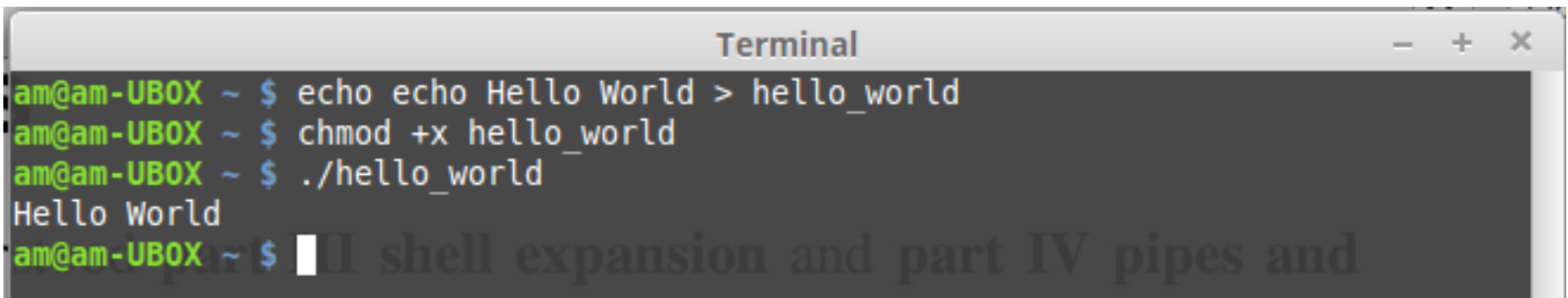
Scripting I

Lesson 9

By Dr.
Amir



Creating a file called `hello_world` and writing “Hello World” to it. Then we make it executable and run it as a

A terminal window titled "Terminal" with standard window controls (minimize, maximize, close) in the top right corner. The terminal shows a user at the "am@am-UBOX" prompt. The user enters three commands: "echo echo Hello World > hello_world", "chmod +x hello_world", and "./hello_world". The output of the third command is "Hello World". The prompt is followed by a cursor and some faint, semi-transparent text from another slide: "part III shell expansion and part IV pipes and".

```
Terminal
am@am-UBOX ~ $ echo echo Hello World > hello_world
am@am-UBOX ~ $ chmod +x hello_world
am@am-UBOX ~ $ ./hello_world
Hello World
am@am-UBOX ~ $
```

she-bang : #!/bin/bash

```
#!/bin/bash
echo -n hello
echo A bash subshell `echo -n hello`
~
~
~ further by putting #!/bin/bash (
~ (sometimes called sha-bang), whe
~
~
~
~
~ "hello world2" 3 lines, 63 characters
```

```
am@am-UBOX ~/Llesson9 $ chmod +x hello_world
am@am-UBOX ~/Llesson9 $ ./hello_world
helloA bash subshell echo -n hello
am@am-UBOX ~/Llesson9 $
```

bash, bash -x

To run a file in Bash. 'bash -x' can be used to run a program and debugging at the same time

```
am@am-UBOX ~ $ bash hello_world2
helloA bash subshell hello
am@am-UBOX ~ $ bash -x hello_world2
+ echo -n hello
hello++ echo -n hello
+ echo A bash subshell hello
A bash subshell hello
am@am-UBOX ~ $
```

```
Terminal
am@am-UBOX ~ $ cat hello_world2
#!/bin/bash
echo -n hello
echo A bash subshell `echo -n hello`
am@am-UBOX ~ $
```

To improve security, '- -'

```
Terminal
#!/bin/bash --
# This is an example of how to avoid spoofing
#
echo \#####
echo \##                Hello World                ##
echo \#####

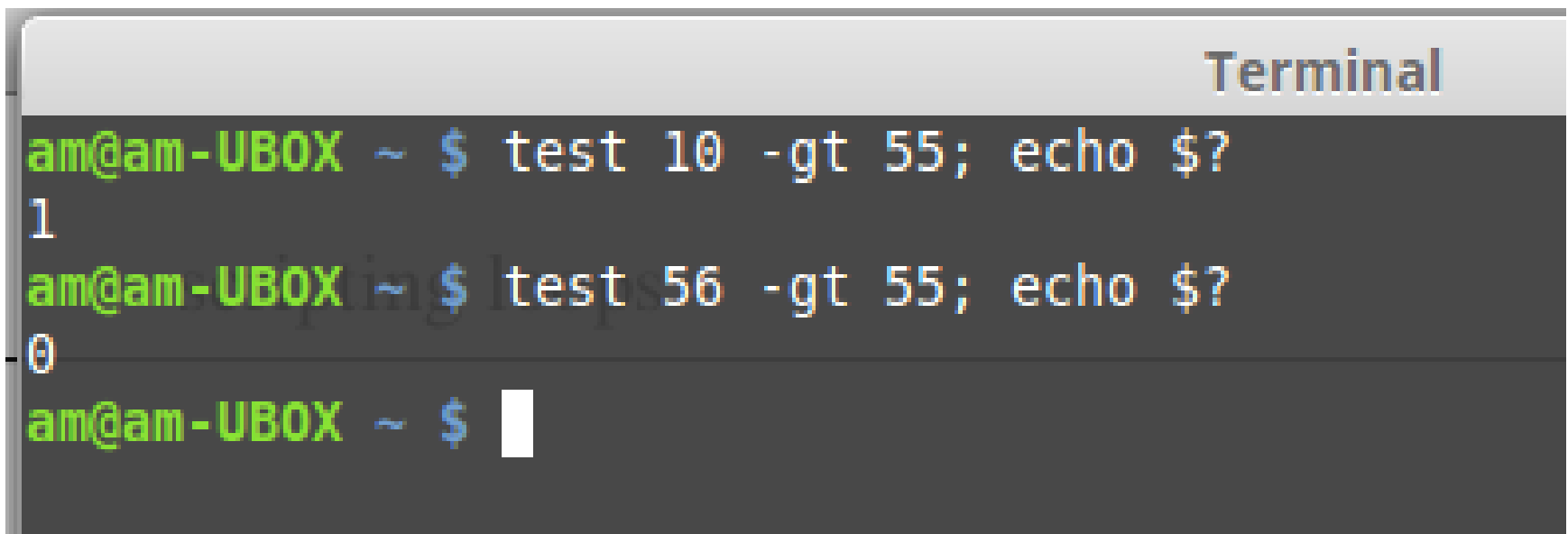
var1=4
echo var1 = $var1

~
"hello_world2" 11 lines, 249 characters
```

```
Terminal
am@am-UBOX ~ $ ./hello_world2
#####
## Hello World
#####
var1 = 4
am@am-UBOX ~ $
```

Command **test**

The test command returns 1 if the test fails. And as you see in the next screenshot, test returns 0 when a test succeeds.

A terminal window titled "Terminal" with a dark background. It shows three lines of command execution. The first line is "am@am-UBOX ~ \$ test 10 -gt 55; echo \$?", followed by the output "1". The second line is "am@am-UBOX ~ \$ test 56 -gt 55; echo \$?", followed by the output "0". The third line shows the prompt "am@am-UBOX ~ \$" with a cursor, indicating the command has been entered but not yet executed.

```
Terminal  
am@am-UBOX ~ $ test 10 -gt 55; echo $?  
1  
am@am-UBOX ~ $ test 56 -gt 55; echo $?  
0  
am@am-UBOX ~ $
```

Command **test** (true / false)

```
Terminal  
am@am-UBOX ~ $ test 10 -gt 55 && echo true || echo false  
false  
am@am-UBOX ~ $ test 56 -gt 55 && echo true || echo false  
true  
am@am-UBOX ~ $
```

Command : if , then , else

Terminal

```
#!/bin/bash

echo \#####
echo \## \ \ \ \ \ \lookfile \ \ \ \ \##
echo \#####

if [ -f isit.txt ]
then echo isit.txt exist!
else echo isit.txt not found!
fi
```

```
am@am-UBOX ~ $ ./lookfile.bash
#####
##          lookfile          ##
#####
isit.txt not found!
am@am-UBOX ~ $
```


Command: read

To read a value from the keyboard

```
GNU nano 2.2.6 File: re
#!/bin/bash

echo 'How old are you?'
read age
echo "You are $age years old"
```

scripting loops

^G Get Help ^O Write
^X Exit ^J Just

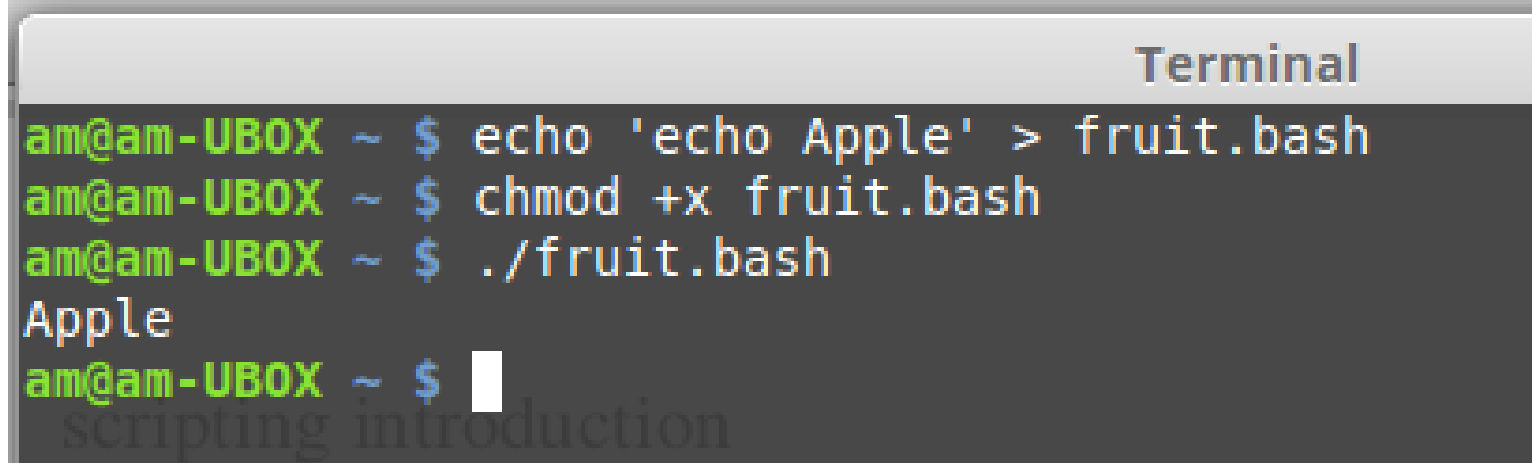
```
Terminal
am@am-UBOX ~ $ chmod +x readline
am@am-UBOX ~ $ ./readline
How old are you?
22
You are 22 years old
am@am-UBOX ~ $
```

Exercises 1

Write a script that check for a file in
`/usr/shared/man.nanorc`

Exercises 1

Write a script that outputs the name of a fruit.



```
Terminal
am@am-UBOX ~ $ echo 'echo Apple' > fruit.bash
am@am-UBOX ~ $ chmod +x fruit.bash
am@am-UBOX ~ $ ./fruit.bash
Apple
am@am-UBOX ~ $
```

Exercises 2

Make sure the script runs in the bash shell.

```
#!/bin/bash  
echo Apple
```

Exercises

Make sure the script runs in the Korn shell.

```
#!/bin/ksh  
echo Apple
```

Exercises

Create a script that defines two variables, and outputs their total.

```
am@am-UBOX ~$ cat >add_num <<end
> #!/bin/bash
>
> var1=10
> var2=15
> var3=$((var1+var2))
> echo $var3
> end
am@am-UBOX ~$
```

```
am@am-UBOX ~ $ cat > add_nums <<end  
> num1=4  
> num2=8  
> echo "$num1+$num2" |bc  
> end
```

```
am@am-UBOX ~ $ cat add_nums  
num1=4  
num2=8  
echo "+" |bc  
am@am-UBOX ~ $
```

```
Terminal
GNU nano 2.2.6      File: add_nums      Modified

#!/bin/bash
num1=4
num2=8
echo "$num1+$num2" | bc

^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```

```
Terminal

am@am-UBOX ~ $ chmod +x add_nums
am@am-UBOX ~ $ ./add_nums
12
am@am-UBOX ~ $
```


Exercises

Write a script to ask the user for name, age, and nationality. Then print out the collected information

Exercises

Write a script to receive two command line arguments and add them together, then display the result.

Exercises

Write a script to ask you for a file name. if it exists, run `ls -ahl`, if it doesn't exists, create and then run `ls -ahl`.