

OS Assignment

by Ashleigh Simons

FILE	23605_ASHLEIGH_SIMONS_OS_ASSIGNMENT_331893_1866116845.DOCX (5.68M)		
TIME SUBMITTED	25-MAR-2021 08:00AM (UTC+0000)	WORD COUNT	2523
SUBMISSION ID	147878898	CHARACTER COUNT	12461

Student Name: Ashleigh Simons

Student Number: 2003948

Course Name: Computing

Module Name: Computer Architecture
and Operating Systems

Contents

Page 3 – Introduction Page

Page 4 – Key Functions of General Operating Systems

Page 5 – Use of CLI commands in Windows OS

Page 6 – Use of CLI commands in a Linux OS

Page 7 - 13 – Use of Batch scripts in a Windows OS

Page 14 - 22 – Use of Bash script in a Linux OS

Page 23 – Conclusion

Page 24 – Reference List

Introduction Page

Within this report it will include an analysis of the key and common functions of a modern general operating system (OS). Also, it will include how CLI commands can be used within a Windows OS and how CLI commands can be used within a Linux OS. Then, it will document the use of batch scripts in a Windows OS. Furthermore, it will document the use of bash shell scripts using a Linux OS. Additionally, it will include a conclusion about the findings of this report and the learning achieved. Finally, it will include a list of all the references used throughout the report that will be to the correct standard of IEEE.

Key Functions of General Operating Systems

Within an operating system there are many key functions. These key functions make the computer able to function properly so that the hardware of the device can work correctly.

The OS functions should include “Security, Memory and Processor management” [1]. The OS can have passwords to protect data on the OS and stops anyone that is not authorized from accessing it. Memory management is all about RAM, the CPU makes sure that programs have enough memory that they can use so that they run smoothly for the user and can perform tasks efficiently. The CPU swaps which application is being used and letting other applications use the RAM when the user selects them leaving room for the other applications running simultaneously. Processor management is needed as it “allocates the CPU that is processor to a process” and “De-allocates processor what a process is no more required” [2] so multiple processes can be done a lot faster and more organized.

Also, “Device & File management and Networking” [3]. File management is the part of the system that keeps documents, files and folders in order. It also manages the software and applications on the device and makes sure that they are in the locations on the device that they are required to be and where the computer expects them to be so that when processes are done, they are done correctly and without problems. Device management is what is used alongside the processor management. This is because it keeps track of the device and what it is doing. Networking manages processors that communicate with each other through networks, it also manages the connections of networks and makes sure that if the device needs to connect to the internet it has a connection and has access to the connection using passwords given by the users if required.

Finally, “Error Detecting and coordination between other software and users” [4]. Error detecting is done by the computer so that it can record the errors in the hardware and then present the information to the user so that they are able to act on it. The operating system is also used to “coordinate the interpreters, compilers and assemblers” [5]. This makes sure that all the software and the hardware is working okay and informing the user so that they do not have confusion about the device and what it is doing.

Use of CLI commands in Windows OS

Command line interface (CLI) can be used by a user on a machine to gain access/ create, change and delete details/ information on a device. Where most people would use a graphical user interface to find out information you are still able to do it in the command prompt on a Windows OS.

Some of the basic Windows CLI command can include “dir, cd\, cd, md, mkdir” [6] and these can all be used to command different options. To find out a list of all the commands, you can type “help” [7] into the command prompt and it will print out all of them in the prompt.

```
Microsoft Windows [Version 10.0.15363.1440]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\ashle>cd\

C:\>dir
Volume in drive C is OS
Volume Serial Number is 4442-98A5

Directory of C:\

16/06/2020 17:32 <DIR>      Apps
16/06/2020 18:21 <DIR>      Dell
07/07/2020 09:22 <DIR>      Downloads
16/06/2020 17:26 <DIR>      Drivers
25/02/2021 16:14 <DIR>      GarethSessionsNewWeb
16/03/2021 08:46 <DIR>      Intel
11/07/2020 22:43 <DIR>      Perflogs
22/02/2021 15:13 <DIR>      Program Files
20/02/2021 17:40 <DIR>      Program Files (x86)
18/02/2021 16:59 <DIR>      ThursdayGarethClass
07/07/2020 09:19 <DIR>      Users
15/03/2021 13:00 <DIR>      Windows
0 File(s)          0 bytes
12 Dir(s)  60,975,265,280 bytes free

C:\>cd Users

C:\Users>dir
Volume in drive C is OS
Volume Serial Number is 4442-98A5

Directory of C:\Users

07/07/2020 09:19 <DIR>      .
07/07/2020 09:19 <DIR>      ..
20/02/2021 20:54 <DIR>      ashle
07/07/2020 09:24 <DIR>      Public
0 File(s)          0 bytes
4 Dir(s)  60,975,265,280 bytes free

C:\Users>cd ashle
```

To show some of the command you can use in the Windows OS command prompt I have gone into my own and will document what they have done.

Firstly, you can see the version of windows installed and the directory at the top of the prompt.

To get to the root directory you can use cd\ . This will then take you to 'C:\>', this is the root directory and from here there is access to other sub folders/ directories.

Then typing dir which is another useful command shows all the directories in the 'C:\>' directory.

I have then selected to go into 'C:\>cd Users' and I have then typed dir again to see the users on this drive. After finding what directory I want which in this case was ashle, I can type 'C:\Users>cd ashle' to change to 'C:\Users\ashle>'.

Now that we are in the right directory, I can use the command 'md' or 'mkdir' which can be used to make a new directory. I have decided to call the directory MYfolder so that I know what the new directory is.

Within this file we are going to use a command to create a text file. To create an empty text file, you can type 'copy nul "MYfile"'. However, "typing this command will always display the file that was copied" [8], this might not be what you want.

Typing 'type nul>....txt' will create a text file and I prefer this method as it takes out that problem.

Now that we have a text file created, we can see it appears within the directory of MYfolder as MYfile.txt. This text file can now be copied, renamed, moved, deleted, or replaced by another directory. Whatever the user wants to do to the file they can from the command prompt.

```
Microsoft Windows [Version 10.0.15363.1440]
(c) 2019 Microsoft Corporation. All rights reserved.

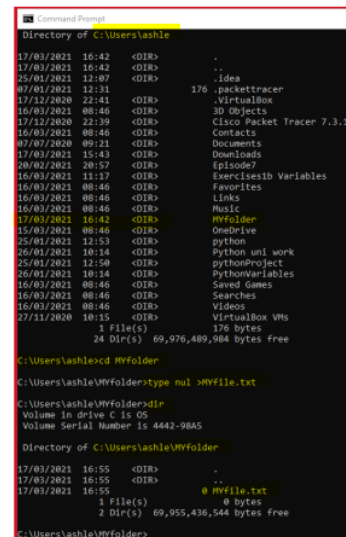
C:\Users\ashle>cd MYfolder
C:\Users\ashle\MYfolder>type nul >MYfile.txt

C:\Users\ashle\MYfolder>dir
Volume in drive C is OS
Volume Serial Number is 4442-98A5

Directory of C:\Users\ashle\MYfolder

17/03/2021 16:55 <DIR>      .
17/03/2021 16:55 <DIR>      ..
17/03/2021 16:55 0 File(s)          0 bytes
2 Dir(s)  60,955,436,544 bytes free

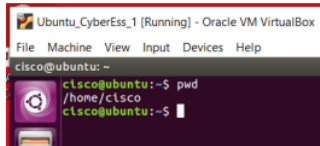
C:\Users\ashle\MYfolder>
```



Use of CLI commands in a Linux OS

To demonstrate the use of commands in a Linux OS it requires the user to firstly have a Linux OS, in this case the demonstrations will be on a Virtual machine running a windows machine as this is an easier way to show how they can be used rather than having two devices with different OS.

The CLI Commands used on Linux OS can be entered into the terminal, this is the place where users can access all their documents and information on a Linux OS. There are many CLI commands for Linux however some of the main basic commands include “pwd”, “ls”, “cd”, “mkdir” and “rmdir” [9].

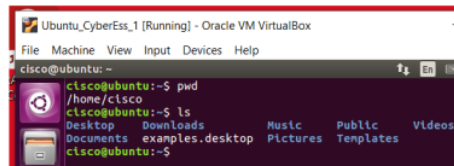


```
Ubuntu_CyberEss_1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
cisco@ubuntu:~$ pwd
/home/cisco
cisco@ubuntu:~$
```

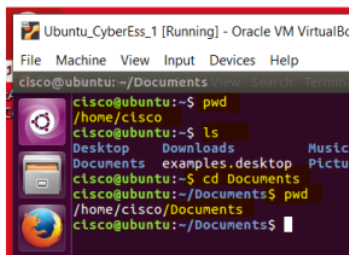
This is one of the simplest commands to understand. Typing ‘pwd’ allows you to find out what directory you are in, as this terminal has just been loaded up it shows that we are in the home and the username is cisco.

The next CLI command we can use is ‘ls’ this will list all of the directories that are within ‘/home/cisco’.

As you can see there is several directories that we can change to access information and the one that is going to be selected is Documents. To be able to do this ‘cd Documents’ needs to be entered. ‘cd’ is what is used to change directories and ‘Documents’ is the directory that is the selected directory to change to.



```
Ubuntu_CyberEss_1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
cisco@ubuntu:~$ pwd
/home/cisco
cisco@ubuntu:~$ ls
Desktop  Downloads  Music     Public   Videos
Documents examples.desktop Pictures  Templates
cisco@ubuntu:~$
```

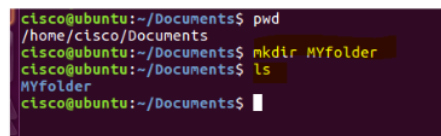


```
Ubuntu_CyberEss_1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
cisco@ubuntu:~/Documents$ pwd
/home/cisco
cisco@ubuntu:~/Documents$ ls
Desktop  Downloads  Music     Public   Videos
Documents examples.desktop Pictures  Templates
cisco@ubuntu:~/Documents$ cd Documents
/home/cisco/Documents
cisco@ubuntu:~/Documents$
```

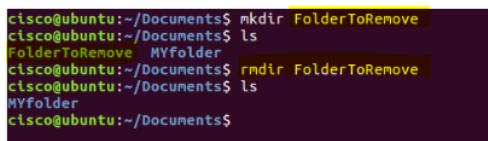
Next, the directory has changed to ‘~/Documents’ to make sure of this you can type ‘pwd’ once again will also show this to us.

Now that we are within the ‘Documents’ directory we can create a folder or a new directory within ‘Documents’. The CLI command to do this is ‘mkdir’ once typing the command you should follow the command with the name of the directory that you want, in this case we will choose ‘MYfolder’.

After we create a folder, typing ‘ls’ will show the directories within ‘Documents’ and the file has been created. To demonstrate another command ‘rmdir’ we can repeat the previous command, so there is another directory to delete.



```
cisco@ubuntu:~/Documents$ pwd
/home/cisco/Documents
cisco@ubuntu:~/Documents$ mkdir MYfolder
cisco@ubuntu:~/Documents$ ls
MYfolder
cisco@ubuntu:~/Documents$
```



```
cisco@ubuntu:~/Documents$ mkdir FolderToRemove
cisco@ubuntu:~/Documents$ ls
FolderToRemove  MYfolder
cisco@ubuntu:~/Documents$ rmdir FolderToRemove
cisco@ubuntu:~/Documents$ ls
MYfolder
cisco@ubuntu:~/Documents$
```

At the top of this terminal, you can see the use of the ‘mkdir’ command again and then ‘rmdir’ that is used to remove the directories that you no longer want.

Use of Batch scripts in a Windows OS

To demonstrate the use of Batch scripts used in a Windows OS I have created a few '.bat' files. A '.bat' file "is a list of several commands" [10]. This means that the script will run and present the workings within the script to the command prompt. I will then go through the script and output and explain with comments what the code is doing. I have decided to use Notepad++ as it is a software that allows you to write .bat files easily and it works with a Windows OS.

File Commands and Variables in Batch Scripts

(Please zoom in if you cannot see any of the script or comments)

```
batch1.bat
1  @echo off
2  :: Above is the command that stops
   the prompt from inserting the
   location of code everytime it is
   echo's to the prompt window
3
4
5  :: The code below will change the
   title of the prompt window so it
   will no longer be 'Command Prompt'
   but Batch Script in a Windows OS
   instead
6  title Batch Script in a Windows OS
7
8  :: Echo can be used to display
   text/ numbers or symbols to the
   display
9  echo Hello and Welcome -
10
11 :: Set is a command to create
   variables
12 set myName=Ash
13 set /A myAge=19
14
15 :: You have to use %% either side
   of a variable for the variable to
   be called and work correctly
16 echo %myName%
17 echo %myAge%
18
19 :: This is a way you can use the
   set command to use other variables
   and create a sentence out of
   defined variables
20 set info=My name is %myName%, I am
   %myAge%
21
22 :: This will echo everthing within
   the info variable
23 echo %info%
24
25 :: This keeps the prompt window
   open so that we can see the
   outputs of the script
26 pause
27
28
```

```
Batch Script in a Windows OS
Hello and Welcome -
Ash
19
My name is Ash, I am 19
Press any key to continue . . .
```

Operators in Batch Scripts

```
batch1.bat batch2.bat
1 @echo off
2 :: Above is the command that stops the prompt from inserting
   the location of code everytime it is echo's to the prompt window
3
4
5 :: The code below will change the title of the prompt window so
   it will no longer be 'Command Prompt' but Batch Script in a
   Windows OS instead
6 title Batch Script in a Windows OS
7
8 :: This is the first line of text that will be visuable in the
   command prompt
9 echo Example of Batch Operators
10
11
12 :: We first have to set the variable using /A and then a
   variable name as we are telling the script that the variable
   will be a number
13
14 :: The first arithmetic operator is * this is used to multiply
   a number by another number, we will call this variable sum1
15 set /A sum1=5*3
16 echo Sum 1 =%sum1%
17 :: echo will display the calculation to sum1 to the command
   prompt
18
19 :: The second operator is - and this will minus the second
   number from the first in this case
20 set /A sum2=6-2
21 echo Sum 2 =%sum2%
22 :: echo will be used again to display the result of the
   calculation
23
24 :: The next operator is + and this will be used to add the two
   numbers together
25 set /A sum3=7+1
26 echo Sum 3 =%sum3%
27 :: echo will again display the result to the command prompt
28
29 :: Then we have / which is dividing the first number by the
   second number
30 set /A sum4=8/4
31 echo Sum 4 =%sum4%
32 :: echo will be used again to display the result of the
   calculation
33
34 :: Then we have % so it will be 8 modulus 5
35 set /A sum5=8%%5
36 echo Sum 5 =%sum5%
37 :: Finally, echo will display the last result to the command
   prompt
38
39 pause
```

CA: Batch Script in a Windows OS

```
Example of Batch Operators
Sum 1 =15
Sum 2 =4
Sum 3 =8
Sum 4 =2
Sum 5 =3
Press any key to continue . . .
```

The GOTO command in Batch Scripts

```
batch1.bat batch2.bat batch3.bat
1 @echo off
2 :: Above is the command that stops the
  prompt from inserting the location of
  code everytime it is echo's to the prompt
  window
3
4
5 :: The code below will change the title
  of the prompt window so it will no longer
  be 'Command Prompt' but Batch Script in a
  Windows OS instead
6 title Batch Script in a Windows OS
7
8 :: This is the first line of text that
  will be visuable in the command prompt
9 echo GOTO command in Batch Scripts -
10
11 :: This is the GOTO command and we will
  use this to skip over the code that is
  between it and where it is jumping to
12 GOTO GoToThis
13
14 ::This is some text we can see now but
  will not appear when we use the GOTO
15 echo This is text that should appear
  however,
16 echo this code will be jumped over.
17 echo So, I will leave this here to show
  the purpose of it
18
19 :: This is where we are telling the GOTO
  command to go to instead of displaying
  whatever comes next in the script
20 :GoToThis
21     echo The GOTO command worked and the
      irrelivant information was not
      displayed
22 ::This can be used in many ways so that
  certain pieces of commands or code is not
  used or displayed to the command prompt
23
24 pause
```

Batch Script in a Windows OS

```
GOTO command in Batch Scripts -
The GOTO command worked and the irrelivant information was not displayed
Press any key to continue . . .
```

If / if else Statements in Batch Scripts

```
batch1.bat batch2.bat batch3.bat batch4.bat
1 @echo off
2 :: Above is the command that stops the prompt from
   inserting the location of code everytime it is echo's to
   the prompt window
3
4 :: The code below will change the title of the prompt
   window so it will no longer be 'Command Prompt' but
   Batch Script in a Windows OS instead
5 title Batch Script in a Windows OS
6
7 :: This is the first line of text that will be visuable
   in the command prompt
8 echo if / if else Statements in Batch Scripts -
9
10 :: First we need to have some variables so that the if
    statement can act on what its being told
11 set /A Number1=18
12 set /A Number2=21
13 :: These are the two variables with numbers that we will
    use to pass through the if statement
14
15 :: This is the if statement code, if the code is passed
    through and its true then it will do one thing and if it
    comes as false then it will do something else
16 if %Number1%==%Number2% (echo The two numbers are the
    same) else (echo The two numbers are different)
17
18 :: We can also use if statements using input from the
    user and this can be done by setting the users input as
    a variable and then seeing if it is the same as the
    number we set it as
19 :: We will use number 3 as the set number for this
    statement
20 set /A Number3=5
21
22 :: Now we will ask the user to enter a number
23 echo Please enter a number
24 set /p userInput=Number:
25
26 :: This is decide what output to show to the user based
    on the number that they have entered, we expect to see
    either of the two responses
27 if %Number3%==%userInput% (echo You have chosen the same
    number to me) else (echo You have chosen a different
    number to me)
28
29 pause
30
```

```
C:\> Batch Script in a Windows OS
if / if else Statements in Batch Scripts -
The two numbers are different
Please enter a number
Number:7
You have chosen a different number to me
Press any key to continue . . .
```

Loops in Batch Scripts

```
batchFilesAndVariables.bat | batchOperators.bat | batchGOTO.bat | batchIfElse.bat | batchLoops.bat
1 @echo off
2 :: Above is the command that stops the prompt from inserting
   the location of code everytime it is echo's to the prompt window
3
4 :: The code below will change the title of the prompt window
   so it will no longer be 'Command Prompt' but Batch Script in a
   Windows OS instead
5 title Batch Script in a Windows OS
6
7 :: This is the first line of text that will be visuable in the
   command prompt
8 echo Loops in Batch Scripts -
9
10 :: Some text to give the loop a heading
11 echo This is the first loop:
12
13 :: Within this Loop, everytime it initiates it looks for the
   number in the list of numbers and then runs the code
   accordinly. The %%c is the variable for the loop and we can
   see the number of each time it is echoing as we can put it at
   the end of Printing-
14 For %%c IN (0, 1, 2, 3, 4, 5) Do (
15   echo Printing- %%c
16 )
17
18 :: Here is some text to split the two loops up
19 echo This is the next loop:
20
21 :: You can also create a loop using a lower range, a higher
   range and an increment. This is the code below
22 For /L %%c In (0, 1, 10) Do (
23   echo Printing Again- %%c
24 )
25
26 pause
27
```

CA Batch Script in a Windows OS

```
Loops in Batch Scripts -
This is the first loop:
Printing- 0
Printing- 1
Printing- 2
Printing- 3
Printing- 4
Printing- 5
This is the next loop:
Printing Again- 0
Printing Again- 1
Printing Again- 2
Printing Again- 3
Printing Again- 4
Printing Again- 5
Printing Again- 6
Printing Again- 7
Printing Again- 8
Printing Again- 9
Printing Again- 10
Press any key to continue . . .
```

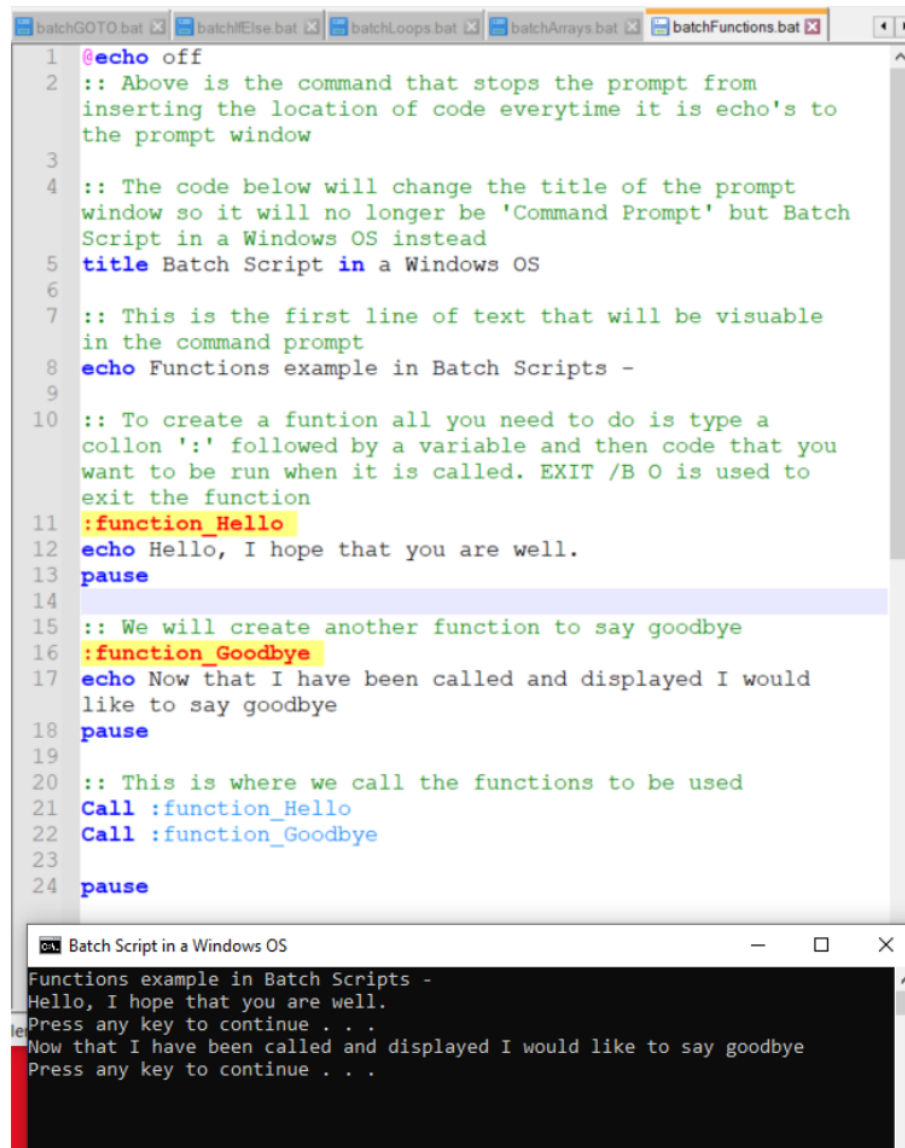
Arrays in Batch Scripts

```
batchFileAndVariables.bat batchOperators.bat batchGOTO.bat batchIfElse.bat batchLoops.bat batchArrays.bat
1  @echo off
2  :: Above is the command that stops the prompt from inserting the location
   of code everytime it is echo's to the prompt window
3
4  :: The code below will change the title of the prompt window so it will
   no longer be 'Command Prompt' but Batch Script in a Windows OS instead
5  title Batch Script in a Windows OS
6
7  :: This is the first line of text that will be visuable in the command
   prompt
8  echo Arrays in Batch Scripts -
9
10 :: Creating a heading to show this is about Arrays
11 echo This section is the output of the Array example:
12
13 :: To create an array element we need to set a variable as an array
   number in [] and then we can type .and then a variable to describe what
   this is after we need =and a value. Below is an example of this using
   lessons, subjects and teachers
14 Set SchoolLesson[1].Teacher=Mr Griffiths
15 Set SchoolLesson[1].Subject=Maths
16
17 Set SchoolLesson[2].Teacher=Mr Coy
18 Set SchoolLesson[2].Subject=English
19
20 Set SchoolLesson[3].Teacher=Mrs Allen
21 Set SchoolLesson[3].Subject=Welsh
22
23
24 :: Now that we have our teachers and subjects assigned to lessons we can
   print these out within a loop
25 For %%c IN (1, 2, 3) Do (
26     call echo The teacher for %SchoolLesson[%%c].Subject% is %S
       choolLesson[%%c].Teacher%
27 )
28
29 pause
```

C:\> Batch Script in a Windows OS

```
Arrays in Batch Scripts -
This section is the output of the Array example:
The teacher for Maths is Mr Griffiths
The teacher for English is Mr Coy
The teacher for Welsh is Mrs Allen
Press any key to continue . . .
```

Functions in Batch Scripts



The image shows a Windows batch script editor with several tabs open: batchGOTO.bat, batchIfElse.bat, batchLoops.bat, batchArrays.bat, and batchFunctions.bat. The active tab is batchFunctions.bat, which contains the following script:

```
1 @echo off
2 :: Above is the command that stops the prompt from
   inserting the location of code everytime it is echo's to
   the prompt window
3
4 :: The code below will change the title of the prompt
   window so it will no longer be 'Command Prompt' but Batch
   Script in a Windows OS instead
5 title Batch Script in a Windows OS
6
7 :: This is the first line of text that will be visuable
   in the command prompt
8 echo Functions example in Batch Scripts -
9
10 :: To create a funtion all you need to do is type a
    collon ':' followed by a variable and then code that you
    want to be run when it is called. EXIT /B 0 is used to
    exit the function
11 :function_Hello
12 echo Hello, I hope that you are well.
13 pause
14
15 :: We will create another function to say goodbye
16 :function_Goodbye
17 echo Now that I have been called and displayed I would
    like to say goodbye
18 pause
19
20 :: This is where we call the functions to be used
21 Call :function_Hello
22 Call :function_Goodbye
23
24 pause
```

Below the script editor, a command prompt window titled "Batch Script in a Windows OS" shows the output of the script:

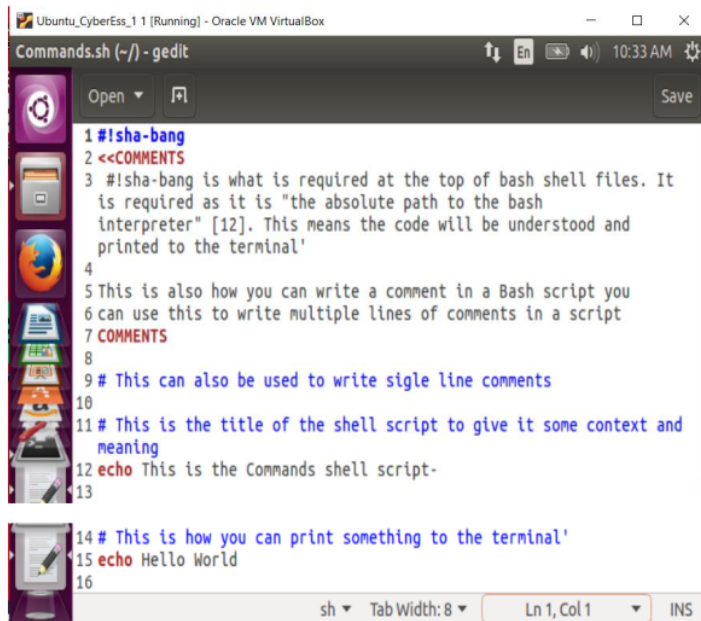
```
Functions example in Batch Scripts -
Hello, I hope that you are well.
Press any key to continue . . .
Now that I have been called and displayed I would like to say goodbye
Press any key to continue . . .
```

Use of Bash in a Linux OS

To demonstrate the use of bash in a Linux OS I will be using Oracle VM Virtual Box. I will be using this as it will allow me to “achieve total Linux system emulation in your Windows PC” [11] meaning that it is a perfect way for me to show this. The demonstrations within these shell scripts will include commands, variables, arithmetic operators, user inputs, if statements, case statements, loops, arrays, and functions.

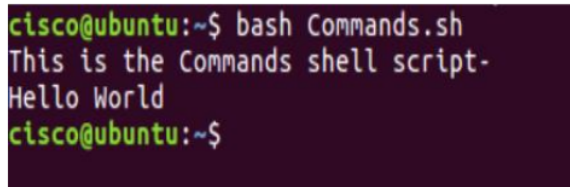
Commands in Bash in a Linux OS

This is the code that is within the first bash shell script and the comments within the script describe what is happening and describing about the echo command, #!sha-bang, and comments.



```
1 #!sha-bang
2 <<COMMENTS
3 #!sha-bang is what is required at the top of bash shell files. It
  is required as it is "the absolute path to the bash
  interpreter" [12]. This means the code will be understood and
  printed to the terminal'
4
5 This is also how you can write a comment in a Bash script you
6 can use this to write multiple lines of comments in a script
7 COMMENTS
8
9 # This can also be used to write sigle line comments
10
11 # This is the title of the shell script to give it some context and
  meaning
12 echo This is the Commands shell script-
13
14 # This is how you can print something to the terminal'
15 echo Hello World
16
```

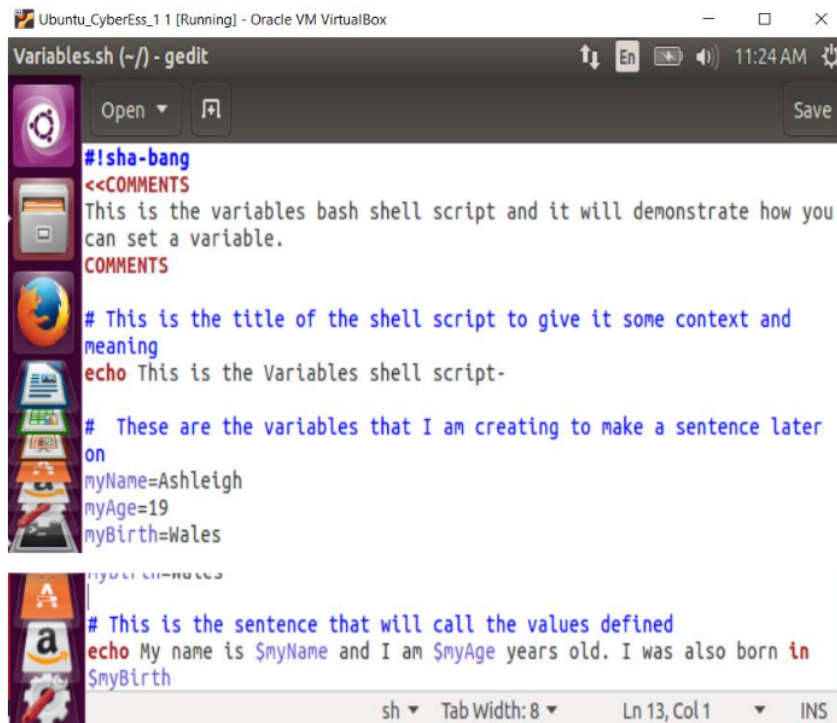
This is what the output of the bash shell script looks like after being interpreted.



```
cisco@ubuntu:~$ bash Commands.sh
This is the Commands shell script-
Hello World
cisco@ubuntu:~$
```

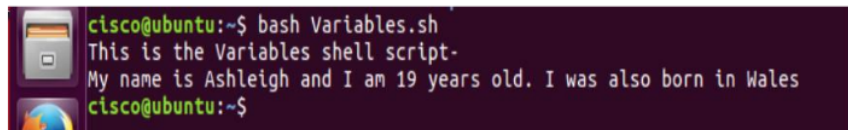
Variables in Bash in a Linux OS

This is the bash shell script that shows how to use a variable. A variable is a “named symbol that represents either a string or numeric value” [13]. This means that you need to have a name for example ‘myName’ to be able to assign a word or a number to it.



```
#!/sha-bang
<<COMMENTS
This is the variables bash shell script and it will demonstrate how you
can set a variable.
COMMENTS
# This is the title of the shell script to give it some context and
meaning
echo This is the Variables shell script-
# These are the variables that I am creating to make a sentence later
on
myName=Ashleigh
myAge=19
myBirth=Wales
# This is the sentence that will call the values defined
echo My name is $myName and I am $myAge years old. I was also born in
$myBirth
```

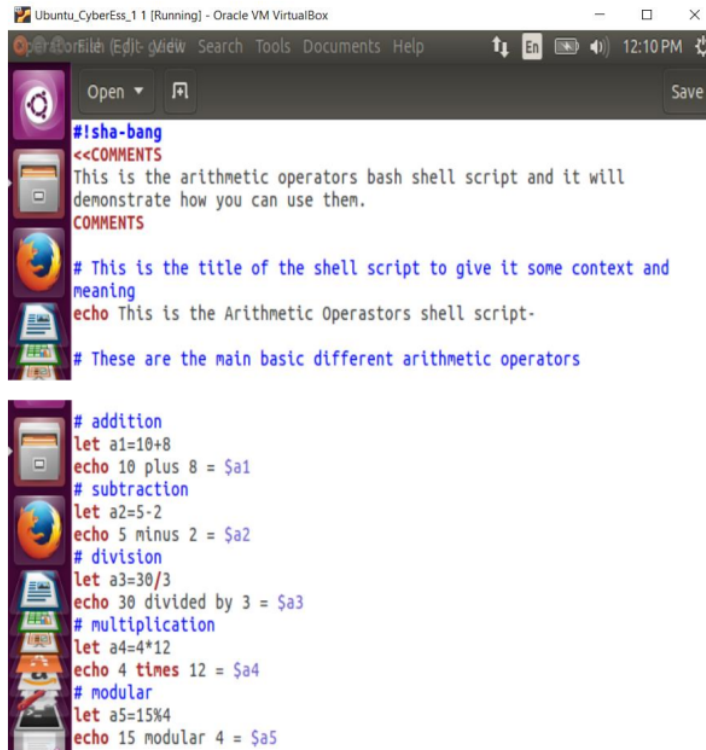
This is the output that you see in the terminal from this bash shell script being interpreted.



```
cisco@ubuntu:~$ bash Variables.sh
This is the Variables shell script-
My name is Ashleigh and I am 19 years old. I was also born in Wales
cisco@ubuntu:~$
```

Arithmetic Operators in Bash in a Linux OS

This is the bash shell script for demonstrating how arithmetic operators can be used. There are many that can be used however the main basic ones are “+ , - , / , * , %” [14]. They can all be used to complete calculations within the scripts. “let” [15] is also a command that can be used along side of the arithmetic operators to allow you to perform calculations.



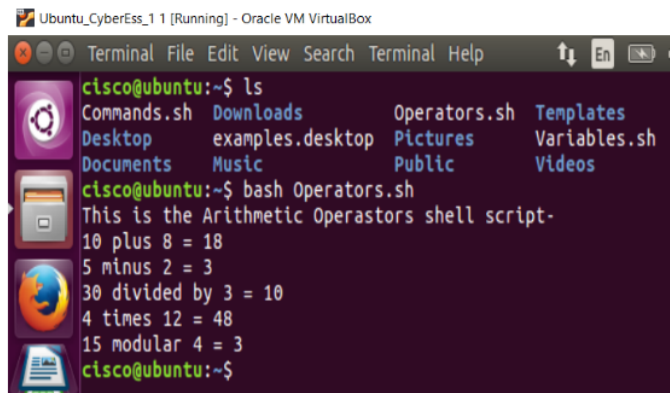
```
#!/sha-bang
<<COMMENTS
This is the arithmetic operators bash shell script and it will
demonstrate how you can use them.
COMMENTS

# This is the title of the shell script to give it some context and
meaning
echo This is the Arithmetic Operastors shell script-

# These are the main basic different arithmetic operators

# addition
let a1=10+8
echo 10 plus 8 = $a1
# subtraction
let a2=5-2
echo 5 minus 2 = $a2
# division
let a3=30/3
echo 30 divided by 3 = $a3
# multiplication
let a4=4*12
echo 4 times 12 = $a4
# modular
let a5=15%4
echo 15 modular 4 = $a5
```

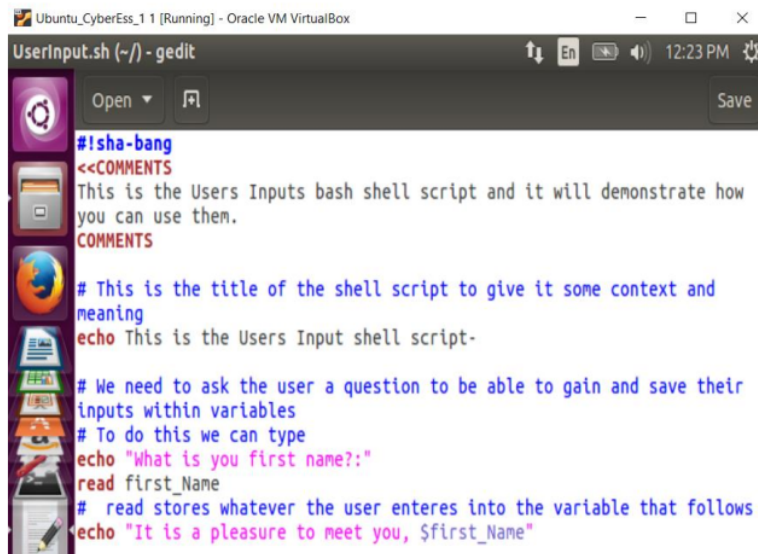
This is the outcome of the script being interpreted.



```
cisco@ubuntu:~$ ls
Commands.sh  Downloads      Operators.sh  Templates
Desktop      examples.desktop  Pictures      Variables.sh
Documents    Music          Public        Videos

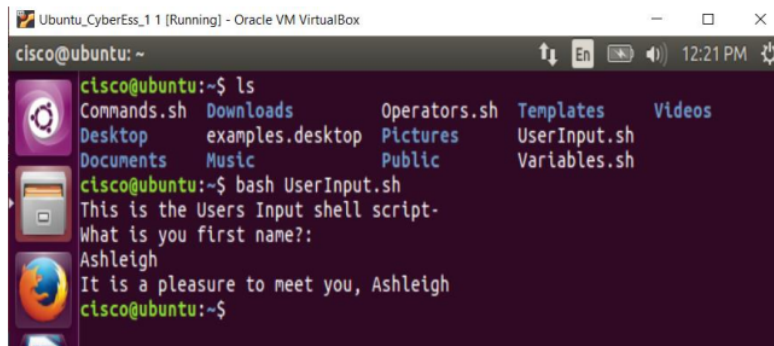
cisco@ubuntu:~$ bash Operators.sh
This is the Arithmetic Operastors shell script-
10 plus 8 = 18
5 minus 2 = 3
30 divided by 3 = 10
4 times 12 = 48
15 modular 4 = 3
cisco@ubuntu:~$
```

User Inputs in Bash in a Linux OS



```
#!/sha-bang
<<COMMENTS
This is the Users Inputs bash shell script and it will demonstrate how
you can use them.
COMMENTS
# This is the title of the shell script to give it some context and
meaning
echo This is the Users Input shell script-
# We need to ask the user a question to be able to gain and save their
inputs within variables
# To do this we can type
echo "What is you first name?:"
read first_Name
# read stores whatever the user enters into the variable that follows
echo "It is a pleasure to meet you, $first_Name"
```

This is the output in the terminal after the code has been interpreted by bash.



```
clisco@ubuntu: ~
Commands.sh  Downloads      Operators.sh  Templates    Videos
Desktop      examples.desktop Pictures      UserInput.sh
Documents    Music          Public       Variables.sh

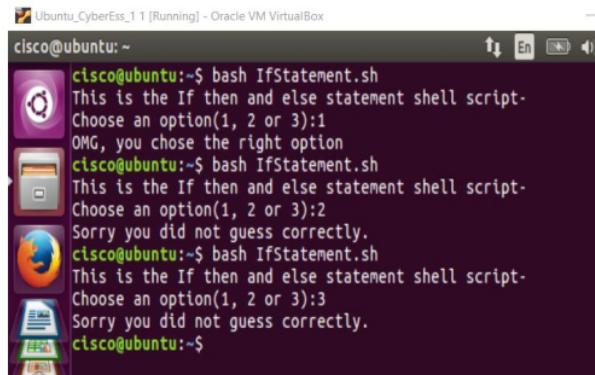
clisco@ubuntu:~$ bash UserInput.sh
This is the Users Input shell script-
What is you first name?:
Ashleigh
It is a pleasure to meet you, Ashleigh
clisco@ubuntu:~$
```

If Statements in Bash in a Linux OS



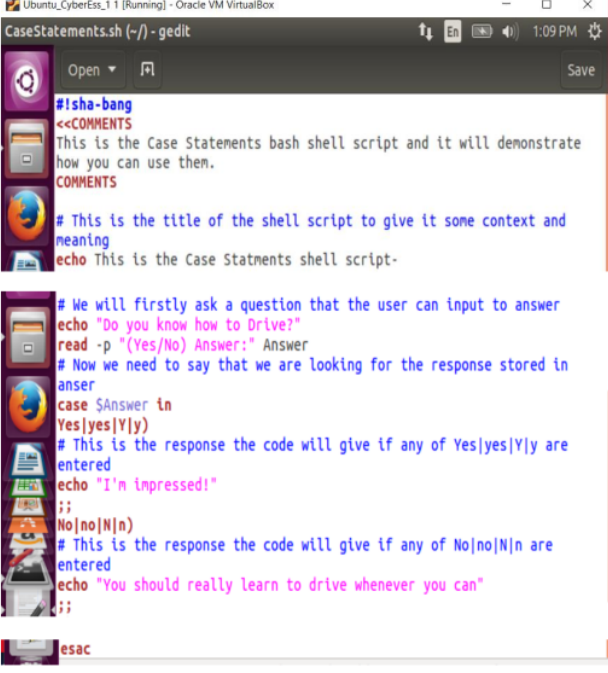
```
1#!/sha-bang
2<<COMMENTS
3This is the if then and else statement bash shell script and it will
  demonstrate how you can use them.
4COMMENTS
5
6# This is the title of the shell script to give it some context and
  meaning
7echo "This is the If then and else statement shell script-"
8
9# To demonstrate this we are going to ask the user to guess a colour
10read -p "Choose an option(1, 2 or 3):" option
11
12# Then we need to create the if statement
13# This first statment will be if they guess correct
14if [ $option == 1 ];
15then
16echo "OMG, you chose the right option"
17fi
18
19# This will be for in incorrect guesses
20if [ $option == 2 ] || [ $option == 3 ];
21then
22echo "Sorry you did not guess correctly."
23fi
24
```

This is the outcome from all three of the possible answers the user could've given.



```
cisco@ubuntu: ~
cisco@ubuntu:~$ bash IfStatement.sh
This is the If then and else statement shell script-
Choose an option(1, 2 or 3):1
OMG, you chose the right option
cisco@ubuntu:~$ bash IfStatement.sh
This is the If then and else statement shell script-
Choose an option(1, 2 or 3):2
Sorry you did not guess correctly.
cisco@ubuntu:~$ bash IfStatement.sh
This is the If then and else statement shell script-
Choose an option(1, 2 or 3):3
Sorry you did not guess correctly.
cisco@ubuntu:~$
```

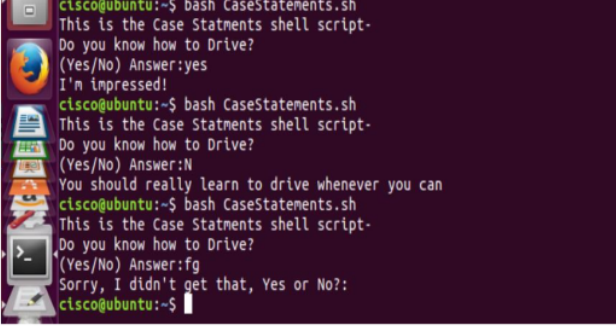
Case Statements in Bash in a Linux OS



```
CaseStatements.sh (~) - gedit
#isha-bang
<<COMMENTS
This is the Case Statements bash shell script and it will demonstrate
how you can use them.
COMMENTS
# This is the title of the shell script to give it some context and
meaning
echo This is the Case Statments shell script-

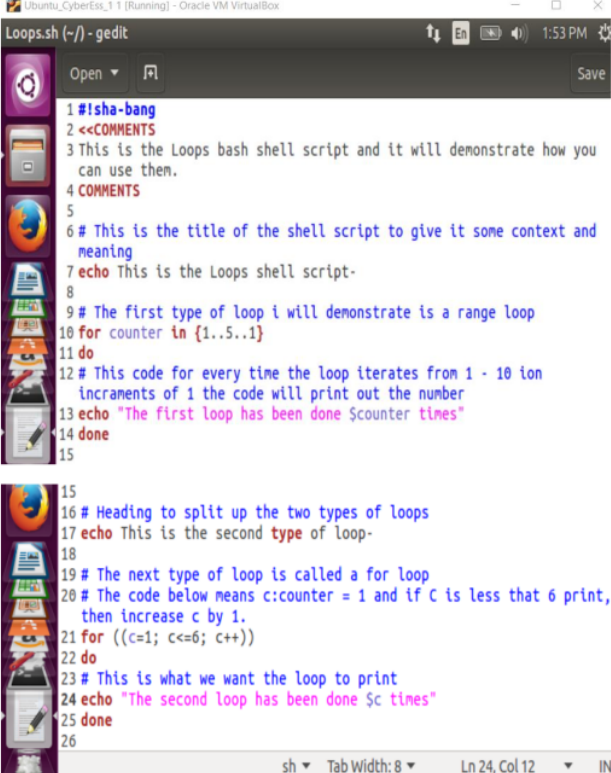
# We will firstly ask a question that the user can input to answer
echo "Do you know how to Drive?"
read -p "(Yes/No) Answer:" Answer
# Now we need to say that we are looking for the response stored in
anser
case $Answer in
Yes|yes|Y|y)
# This is the response the code will give if any of Yes|yes|Y|y are
entered
echo "I'm impressed!"
;;
No|no|N|n)
# This is the response the code will give if any of No|no|N|n are
entered
echo "You should really learn to drive whenever you can"
;;
esac
```

This is the possible outcomes or the case statements shell script.



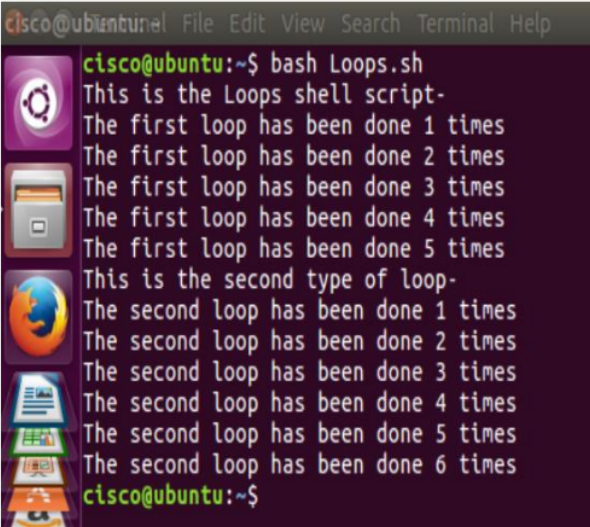
```
clisco@ubuntu:~$ bash CaseStatements.sh
This is the Case Statments shell script-
Do you know how to Drive?
(Yes/No) Answer:yes
I'm impressed!
clisco@ubuntu:~$ bash CaseStatements.sh
This is the Case Statments shell script-
Do you know how to Drive?
(Yes/No) Answer:N
You should really learn to drive whenever you can
clisco@ubuntu:~$ bash CaseStatements.sh
This is the Case Statments shell script-
Do you know how to Drive?
(Yes/No) Answer:fg
Sorry, I didn't get that, Yes or No?:
clisco@ubuntu:~$
```

Loops in Bash in a Linux OS



```
1 #Isha-bang
2 <<COMMENTS
3 This is the Loops bash shell script and it will demonstrate how you
  can use them.
4 COMMENTS
5
6 # This is the title of the shell script to give it some context and
  meaning
7 echo This is the Loops shell script-
8
9 # The first type of loop i will demonstrate is a range loop
10 for counter in {1..5..1}
11 do
12 # This code for every time the loop iterates from 1 - 10 ion
  incraments of 1 the code will print out the number
13 echo "The first loop has been done $counter times"
14 done
15
16 # Heading to split up the two types of loops
17 echo This is the second type of loop-
18
19 # The next type of loop is called a for loop
20 # The code below means c:counter = 1 and if C is less that 6 print,
  then increase c by 1.
21 for ((c=1; c<=6; c++))
22 do
23 # This is what we want the loop to print
24 echo "The second loop has been done $c times"
25 done
26
```

This is the output of the Loops bash shell script.



```
cisco@ubuntu:~$ bash Loops.sh
This is the Loops shell script-
The first loop has been done 1 times
The first loop has been done 2 times
The first loop has been done 3 times
The first loop has been done 4 times
The first loop has been done 5 times
This is the second type of loop-
The second loop has been done 1 times
The second loop has been done 2 times
The second loop has been done 3 times
The second loop has been done 4 times
The second loop has been done 5 times
The second loop has been done 6 times
cisco@ubuntu:~$
```

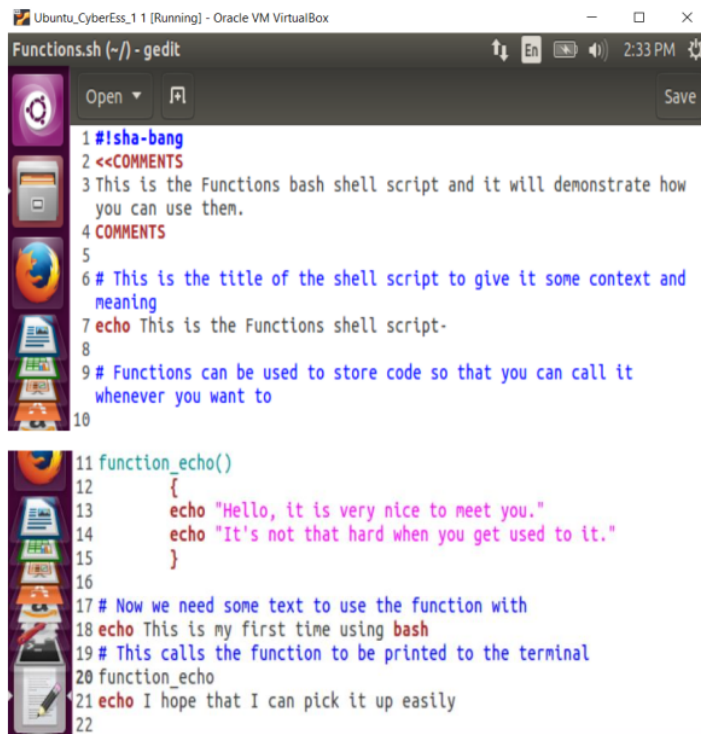
Arrays in Bash in a Linux OS

```
Ubuntu_CyberEss_1 1 [Running] - Oracle VM VirtualBox
Arrays.sh (-/) - gedit
1 #!sha-bang
2 <<COMMENTS
3 This is the Arrays bash shell script and it will demonstrate how you
  can use them.
4 COMMENTS
5
6 # This is the title of the shell script to give it some context and
  meaning
7 echo This is the Arrays shell script-
8
9 # Arrays can have many different pieces of text in one array
10 # For the array demonstration we will use a shopping list
11 declare shopping=( "Milk" "Bread" "Ham" "Butter" "Sauce" "Cheese" )
12 # We can now select items of the shopping list if we want to
13
14 # This will look for the declared shopping and then the third item
  that is listed as the array starts 0, 1, 2, 3, ... meaning Ham will
  be selected
15
16 # We can add some context to what we want
17 echo What things do we need to buy?
18 echo ${shopping[2]}
19 echo ${shopping[4]}
20 echo ${shopping[5]}
21
22 # Arrays can also be made in multi-lines
23 declare -a chocolate=(
24 [1]="Dairy Milk"
25 [2]="Galaxy"
26 )
27 # We echo to the terminal the same as before
28 echo ${chocolate[1]}
```

This is the output of the Arrays bash shell script.

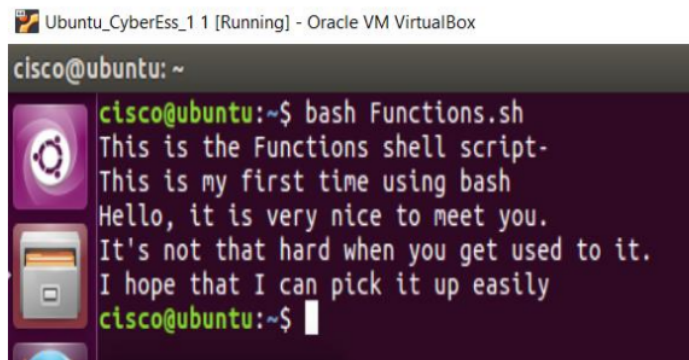
```
Ubuntu_CyberEss_1 1 [Running] - Oracle VM VirtualBox
cisco@ubuntu: ~
cisco@ubuntu:~$ bash Arrays.sh
This is the Arrays shell script-
What things do we need to buy?
Ham
Sauce
Cheese
Dairy Milk
cisco@ubuntu:~$
```

Functions in Bash in a Linux OS



```
1 #!sha-bang
2 <<COMMENTS
3 This is the Functions bash shell script and it will demonstrate how
  you can use them.
4 COMMENTS
5
6 # This is the title of the shell script to give it some context and
  meaning
7 echo This is the Functions shell script-
8
9 # Functions can be used to store code so that you can call it
  whenever you want to
10
11 function_echo()
12 {
13     echo "Hello, it is very nice to meet you."
14     echo "It's not that hard when you get used to it."
15 }
16
17 # Now we need some text to use the function with
18 echo This is my first time using bash
19 # This calls the function to be printed to the terminal
20 function_echo
21 echo I hope that I can pick it up easily
22
```

This is the output of the Functions bash shell script.



```
cisco@ubuntu: ~
cisco@ubuntu:~$ bash Functions.sh
This is the Functions shell script-
This is my first time using bash
Hello, it is very nice to meet you.
It's not that hard when you get used to it.
I hope that I can pick it up easily
cisco@ubuntu:~$
```

Conclusion

To conclude, from completing this report I can now say that I am fully more informed of the key functions of general operating systems. I now know the requirements of operating systems for instance having security, memory, and processor management and how important these and its functionality is important for an operating system along with device, network, and file management and it shows the operating system needs so many aspects to be as efficient as it can be. Also, having error detection and coordination between the software and users controlling the device is vital in getting the best out of a operating system on a device for a user.

Furthermore, from documenting CLI Commands in both Windows OS and Linux OS I now understand a part of why this way of using a computer can be such more efficient than using graphical user interfaces. The information that you can gain from the terminal and command prompt is colossal if you know how to use it efficiently and you know what you are looking for. You can also gain access to a lot of information about your device that you can understand in ways that many might not be able to as the use of graphical user interfaces are so popular. I defiantly realise a lot more now that I have learnt about them and what devices really have to offer.

Finally, after completing the Batch scripts in a Windows OS and the use of Bash scripts in a Linux OS I now realise just how similar the two operating systems are, they both use the same sort of fundamentals given there are differences to make them unique and different however with these differences they can do what is required of them in the environments of the operating systems. It has given me a good insight into what goes into devices and the scripts that can be written, run and used on them.

Reference List

- [1, 2] A. Singh, "Functions of Operating System", [Accessed Online], <https://www.geeksforgeeks.org/functions-of-operating-system/>, Dec. 23, 2020, [Last Accessed: March. 17, 2021].
- [3] GURU99, "What is an Operating System? Types of OS, Features and Examples", [Accessed Online], <https://www.guru99.com/operating-system-tutorial.html>, [Last Accessed: March. 17, 2021].
- [4] tutorialspoint, "Operating System – Overview", [Accessed Online], https://www.tutorialspoint.com/operating_system/os_overview.htm, [Last Accessed: March. 17, 2021].
- [5] P. Pedamkar, "Functions of Operating System", [Accessed Online], <https://www.educba.com/functions-of-operating-system/>, [Last Accessed: March. 17, 2021].
- [6, 7] w3schools.com, "What is Command Line Interface (CLI)", "Table: Basic Window CLI Commands", [Accessed Online], https://www.w3schools.com/whatis/whatis_cli.asp, [Last Accessed: March. 17, 2021].
- [8] A. Farrad, "How to create empty files from the command prompt (cmd)", [Accessed Online], <https://techcult.com/how-to-create-empty-files-from-the-command-prompt-cmd/>, Feb. 17, 2021, [Last Accessed: March. 17, 2021].
- [9] A. Naushad, "Learn basic commands for Linux, a free and open-source operating system that you can make changes to and redistribute", [Accessed Online], [https://maker.pro/linux/tutorial/basic-linux-commands-for-beginners#:~:text=Linux%20Shell%20or%20%22%22%20Terminal%22%20%20&text=Linux%20shell%20is%20%22%20main,CLI%20\(command%20line%20interface\).&text=To%20open%20the%20terminal%2C%20press,%20terminal%2C%20and%20press%20enter](https://maker.pro/linux/tutorial/basic-linux-commands-for-beginners#:~:text=Linux%20Shell%20or%20%22%22%20Terminal%22%20%20&text=Linux%20shell%20is%20%22%20main,CLI%20(command%20line%20interface).&text=To%20open%20the%20terminal%2C%20press,%20terminal%2C%20and%20press%20enter), March. 21, 2018, [Last Accessed: March. 20, 2021].
- [10] C. Hoffman, "How to Write a Batch Script on Windows", [Accessed Online], <https://www.howtogeek.com/263177/how-to-write-a-batch-script-on-windows/>, July. 5, 2017, [Last Accessed: March. 23, 2021].
- [11] M. Hasan, "Top 15 Best Linux Emulators for Windows System in 2020", [Accessed Online], <https://www.ubuntupit.com/best-linux-emulators-for-windows-system/>, [Last Accessed: March. 24, 2021].
- [12] Linux, "Shebang", [Accessed Online], <https://bash.cyberciti.biz/guide/Shebang>, [Last Accessed: March. 24, 2021].
- [13] D. McKay, "How to Work with Variables in Bash", [Accessed Online], <https://www.howtogeek.com/442332/how-to-work-with-variables-in-bash/>, Oct. 7, 2019, [Last Accessed: March. 24, 2021].
- [14, 15] Shell-Tips.com, "Math Arithmetic: How To Do Calculation in Bash?", [Accessed Online], <https://www.shell-tips.com/bash/math-arithmetic-calculation/>, [Last Accessed: March. 24, 2021].

OS Assignment

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Instructor

85/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

GRADING FORM: UWTSD GRADING FORM

ASHLEIGH SIMONS

85

1. RESEARCH LITERATURE

Extent of research and/or own reading, selection of credible sources, application of appropriate referencing conventions.



+ Good use of citations. + formatted reference list.
- 1/2 books/academic papers could be included in reference list.

17

2. KNOWLEDGE AND UNDERSTANDING

Extent of knowledge and understanding of concepts and underlying principles associated with the discipline.



1. Key functions: Good knowledge of key OS functions documented. – Memory allocation description could be more accurate. 2. Bat Commands: + Good use of relevant commands with accompanying narration that demonstrates a

16

sound understanding of the commands. – 1/2 additional commands could be used. 3. Bash Commands: + Good use of relevant commands with accompanying narration that demonstrates a sound understanding of the commands. – 1/2 additional commands could be used. 4. Bat Scripts: + Very Good use of relevant scripts. 5. Bash Scripts: + Very Good use of relevant scripts.

3. ANALYSIS

Analysis, evaluation and synthesis; logic, argument and judgement; analytical reflection; organisation of ideas and evidence.



Evidence of analysis and understanding of OS functions and the relevant languages is clearly documented. Very good. + Very Good conclusion.

10

4. PRACTICAL APPLICATION

Deployment of methods, materials, tools and techniques; application of concepts; formulation of innovative and creative solutions to solve problems.



1. Bat Commands: + Good use of relevant commands. – 1/2 additional commands could be used. 2. Bash Commands: + Good use of relevant commands. – 1/2 additional commands could be used. 3. Bat Scripts: + Very Good use of relevant scripts. 4. Bash Scripts: + Very Good use of relevant scripts.

34

5. PROFESSIONAL PRACTICE

Attributes in professional practice: individual and collaborative working; deployment of appropriate media; presentation and organisation.



+ Good report format, structure, and presentation.
- Report sections could be numbered. - Appendix:
Activity 1 answer not included. - 1/2 minor spelling
errors.

8