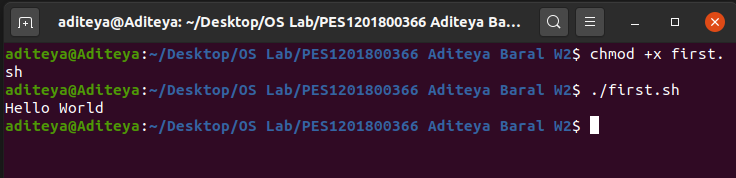
OS Lab Report – Week 2

PES1201800366

Aditeya Baral

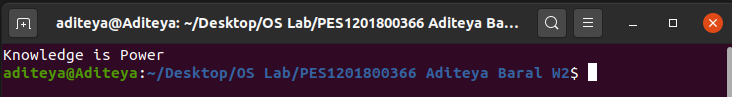
1. First.sh

1. #!/bin/sh
2. # This is a comment!
3. echo Hello World # This is a comment, too!



2. Second.sh

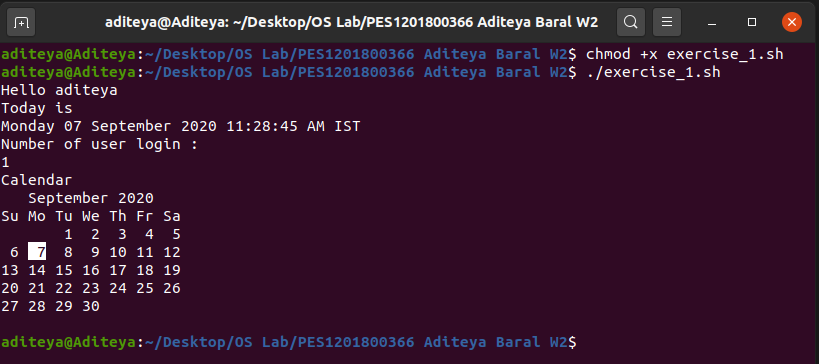
1. #!/bin/sh
2. clear
3. echo "Knowledge is Power



3. Exercise 1

1. #!/bin/sh
2. # Script to print user information who currently login, current date & time
3. # Enter the following commands in a file
4. echo "Hello $USER"
5. echo "Today is ";date
6. echo "Number of user login : " ; who | wc -l
7. echo "Calendar"
8. cal
9. exit 0

 3.1 Output of Code Snippet



3.2 Reason behind exit 0

Every Linux or Unix command executed by the shell script or user has an exit status which is an integer number. exit 0 status means the command was successful without any errors. A non-zero (1-255 values) exit status means the command was a failure.

3.3 Significance of $

In a shell script, the $ symbol is used to access the value stored in an identifier or variable. If the $ symbol is not used as a prefix before the variable name, the shell will just display the name of a variable.

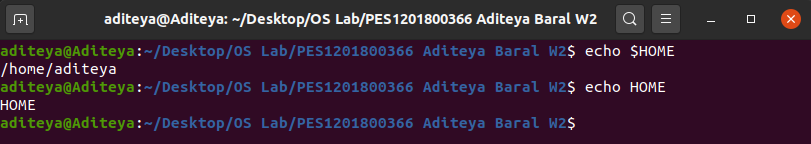
4. System Variables

1. echo $USERNAME
2. echo $USER
3. echo $HOME
4. echo $BASH



5. Exercise 2

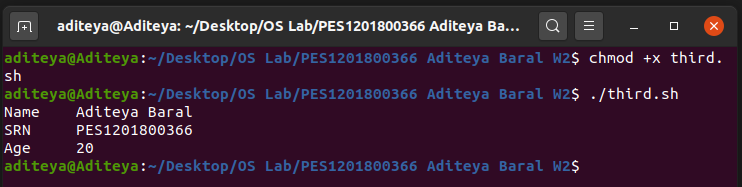
1. echo $HOME
2. echo HOME



Hence, echo $HOME is the right command to display the home directory, since HOME is a system variable, and values stored in variables in a shell script can only be accessed using a $ symbol as a prefix.

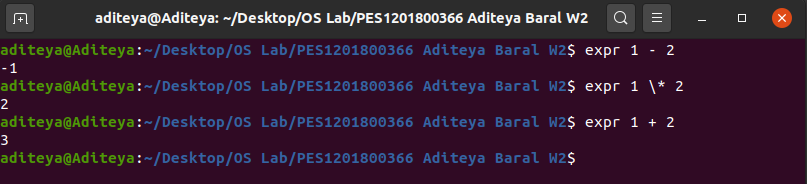
6. User Defined Variables

1. #!/bin/sh
2. name="Aditeya Baral"
3. srn=PES1201800366
4. age=20
5. echo "Name\t$name"
6. echo "SRN\t$srn"
7. echo "Age\t$age"



7. Shell Arithmetic

1. expr 1 – 2
2. expr 1 \\* 2

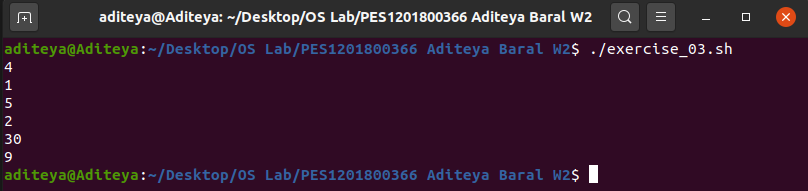


8. Exercise 3

8.1 Code

1. #1/bin/sh
2. expr 1 + 3
3. expr 2 - 1
4. expr 10 / 2
5. expr 20 % 3
6. expr 10 \\* 3
7. echo `expr 6 + 3`

8.2 Output

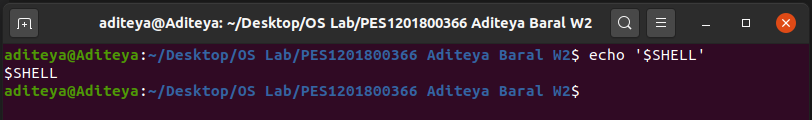


9. Exercise 4

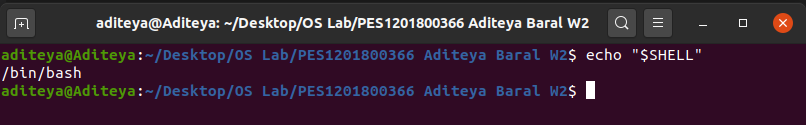
**Question - What is the meaning of Single quote (‘), Double quote (“) and Back quote (`) in shell?**

**Answer –** The shell understands special characters (such as escape sequences) with special meanings. For example, $variable is used to expand and obtain the value stored in variable. It also expands wildcards (such as \* and ?). However, sometimes, we need to display them as is. In such cases, we can use the various quoting methods.

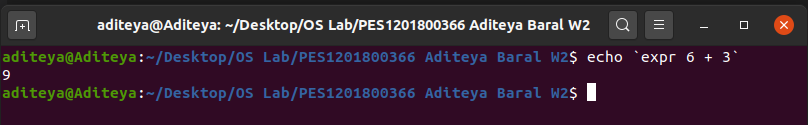
* **Single Quotes (‘):** 
  + The single quote is used to remove the special meaning of **any character** enclosed within them i.e. the special characters are treated as ordinary strings
  + Variables, wildcards as well as command substitutions are disabled
  + Example: echo ‘$SHELL’ will display $SHELL



* **Double Quotes (“):**
  + The double quote is used to remove the special meaning of **most characters** enclosed within them i.e. most special characters are treated as ordinary strings
  + Only wildcards are disabled
  + This does not apply to $, ‘, “, `, \$, and \
  + Example: echo ‘$SHELL’ will display /bin/bash

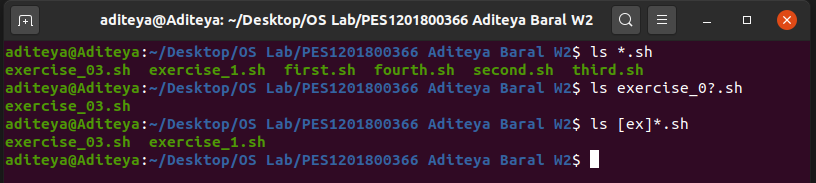


* **Back Quotes (`):**
  + The back quote is used to **execute any command** enclosed within back quotes
  + Example: echo `expr 6 + 3` will display 9



10. Wildcards

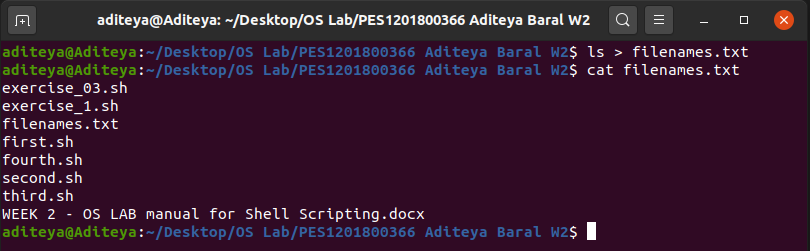
1. #!/bin/sh
2. ls \*.sh
3. ls exercise\_0?.sh
4. ls [ex]\*.sh



11.  Redirection of Standard IO

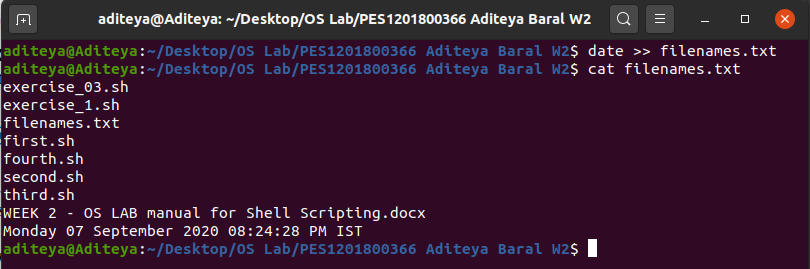
11.1 Output Redirection

1. ls > filenames.txt
2. cat filenames.txt



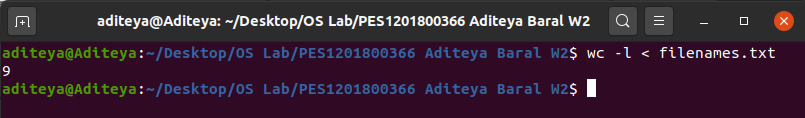
11.2 Output Redirection with Redirector

1. date >> filenames.txt
2. cat filenames.txt



11.3 Input Redirection

1. wc -l filenames.txt

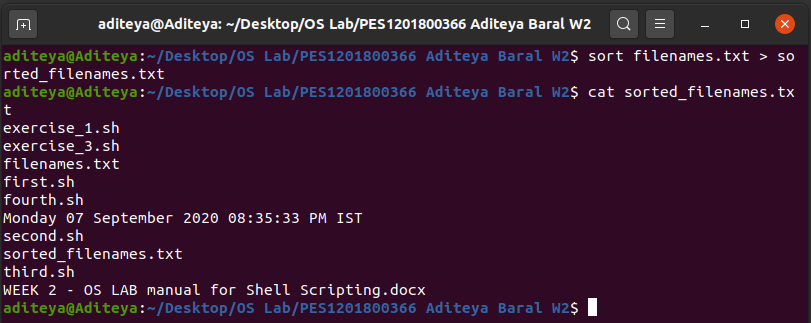


12. Exercise 5

**Question – What does the following command do?**

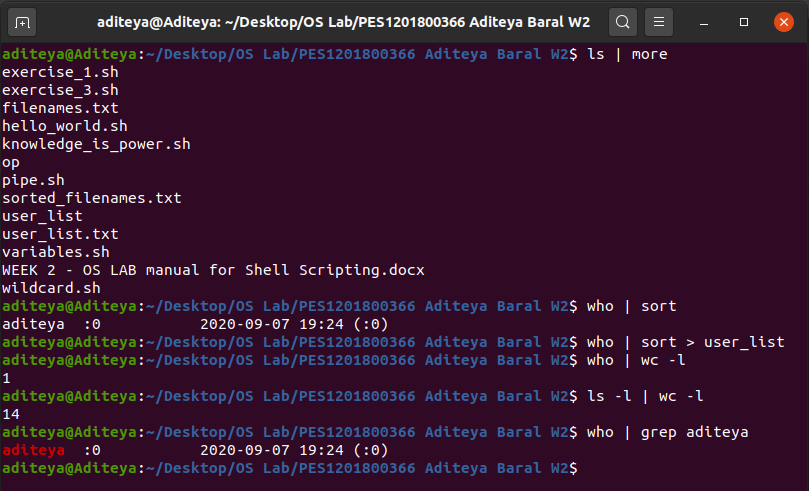
1. sort < myfile > sorted\_file

**Answer –** The given command will first use input redirection to obtain the contents (lines) of myfile, which is sent to the sort command. After sorting the lines in alphabetical order, the output is redirected to sorted\_file and stored.



13. Pipes

1. ls | more
2. who | sort
3. who | sort > user\_list.txt
4. who | wc -l
5. ls -l | wc -l
6. who | grep aditeya

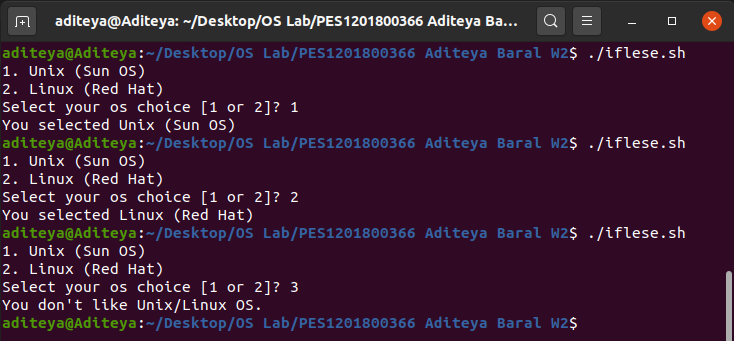


14. if-else-fi Construct

14.1 Code

1. #!/bin/sh
2. osch=0
4. echo "1. Unix (Sun OS)"
5. echo "2. Linux (Red Hat)"
6. echo -n "Select your os choice [1 or 2]? "
7. read osch
9. if [ $osch -eq 1 ]
10. then
11. echo "You selected Unix (Sun OS)"
13. else
14. if [ $osch -eq 2 ]
15. then
16. echo "You selected Linux (Red Hat)"
17. else
18. echo "You don't like Unix/Linux OS."
19. fi
20. fi

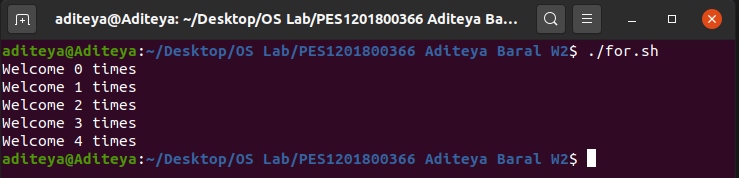
 14.2 Output



15. Loops

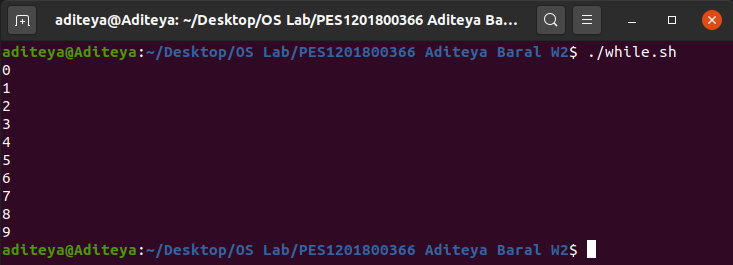
15.1 For Loop

1. #!/bin/sh
2. for ((i = 0;i <= 4;i++))
3. do
4. echo "Welcome $i times"
5. done



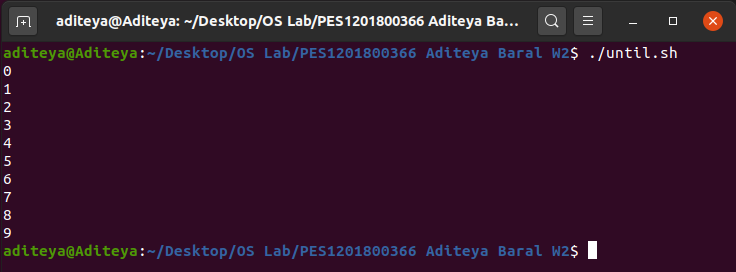
15.2 While Loop

1. #!/bin/sh
2. a=0
3. while [ $a -lt 10 ]
4. do
5. echo $a
6. a=`expr $a + 1`
7. done



15.3 Until Loop

1. #!/bin/sh
2. a=0
3. until [ ! $a -lt 10 ]
4. do
5. echo $a
6. a=`expr $a + 1`
7. done



16. Exercise 6

16.1 Code

1. #!/bin/sh
2. ls | tr '[:upper:]' '[:lower:]'

16.2 Output

