/

---------------------------------------------------------------------------

| | | |

opt home bin sys

| /------------------\

rh ahmed yasser

/---------|----------------\ \

folder\_x file\_1 folder\_y my\_folder

Relative Path:-

1-Example ( You’re current location is folder “folder\_x” and you want to move into folder “rh")

[ahmad@localhost folder\_x]$ cd ../../../opt/rh

Absolute path:-

2-Example ( You’re current location is folder “folder\_x” and you want to move into folder “rh)

[ahmad@localhost folder\_x]$ cd /opt/rh

Assuming your current location is “rh” and you want to

3- create a file named “file\_1” under the folder “ahmad

[ahmad@localhost ~]$ touch ../../home/ahmad/file\_1

4-create multiple files having a common sub-name like for example (f1, f2,f3, f4, f5, f6)

[ahmad@localhost ~]$ touch f{1..6}

5- remove a file named “file\_1” under your current location

[ahmad@localhost ~]$ rm file\_1

6-Suppose you want to remove multiple files having a common sub-name like for example (f1, f2,f3)

[ahmad@localhost ~]$ rm f\*

7- Assuming your current location is “rh” and you want to remove a file named “file\_1” under the folder “ahmad”

[ahmad@localhost ~]$ rm ../../home/ahmad/file\_1

8-Copy a file using the com --> cp src\_file dest\_fil (your syntax)

Assuming being in “folder\_x”, where we have a file named “my\_file” and we want to copy this file into “folder\_y”

[ahmad@localhost ~]$ cp file\_1 ../folder\_y

9-create file and moves the file “file\_3003” into the directory“Desktop”

[ahmad@localhost ~]$ touch file\_3003

[ahmad@localhost ~]$ mv file\_3003 Desktop

10-create the directory “my\_new\_folder” inside of your current location --> /home/ahmad

[ahmad@localhost ~]$ mkdir my\_new\_folder

11-remove the directory “my\_new\_folder” inside of your current location --> /home/ahmad

[ahmad@localhost ~]$ rmdir my\_new\_folder

12-create and copiy the directory “my\_new\_folder\_4” into the directory “Desktop”

[ahmad@localhost ~]$ mkdir my\_new\_folder\_4

[ahmad@localhost ~]$ cp -r my\_new\_folder\_4 Desktop

13- create and moves the directory “my\_new\_folder\_6” and all of its contents into the directory “Desktop”

[ahmad@localhost ~]$ mkdir my\_new\_folder\_6

[ahmad@localhost ~]$ mv my\_new\_folder\_6 Desktop

14- Login as “root” using command “su”

[ahmad@localhost ~]$ su

Password: [enter your root user password here]

[root@localhost ~]$ (but your password here)

15-create user

[root@localhost ~]$ useradd user\_1 or

[root@localhost ~]$ adduser user\_1

16- create group

[root@localhost ~]$ groupadd group\_1 #Note here no command addgroup

17- change the password of “user\_1”

[root@localhost ~]$ passwd user\_1

New password: [enter the new password here]

Retype new password: [enter the new password here again]

18- Add a user to group/groups

[root@localhost ~]$ usermod user\_1 --append --group G1,G2 or

[root@localhost ~]$ usermod user\_1 -aG G1,G2

19- Another useful option is --expiredate .. which can be used to set a date after which the user will not be able to login to the system

[root@localhost ~]$ usermod user\_1 --expiredate 2021-04-21

20- Another option is --home .. which is used to set the default location of a user

[root@localhost ~]$ usermod user\_1 --home /bin

This means when user\_1 logs into the system he will be moved automatically to /bin

21- delete user

[root@localhost ~]$userdel user\_1

\*\*\* Permissions for Files \*\*\*

R ==> Read (Being able to view a file contents)

W ==> Write (Being able to modify a file contents)

X ==> Execute (Being able to run a file as a program

Every file has both an owner and a group

If we list the permissions of a file named “file\_1”, using the commd

[ahmad@localhost ~]$ ls -l file\_1

//output: - --- --- --- Owner Group Other File\_name

- rwx rw- r-- ahmad admins file\_1

The first cell show the type (- for normal file ,d for directory)

The next three cells represent (owner,group,others) permissions

🡪 If we list the permissions of another file and get the following output:

- r-- rw- r - - ahmad ,What permissions does ahmad has over file\_2 ?

-->ahmad only has (Read), admins whose members should be able to (Read and Write)

**Setting Permissions Numerically**

Every permission of the three permissions is assigned a number

Read R ==> 4

Write W ==> 2

Execute X ==> 1

Read and Execute) we assign them the number (4 + 1 ==> 5)

For example the following number 536 means that:

The first digit for the Owner ,the second digit for the Group, the third digit for the Others

Owner has=> Read & Execute (4 + 1 = 5)

Group has=> Write & Execute (2 + 1 = 3)

Others has=> Read & Write (4 + 2 = 6)

Changing the permission is performed using the command chmod which is

short for (change mode) : chmod 3-digits file\_name (syntax)

22- List “file\_1” permissions using

[ahmad@localhost ~]$ ls -l file\_1

//output:- rwx rw- r - - ahmad admins file\_1

23-Change “file\_2” permissions using

[ahmad@localhost ~]$ chmod 536 file\_1

24- List “file\_1” permissions again

[ahmad@localhost ~]$ ls -l file\_1

//output: - r-x -wx rw- ahmad admins file\_1

25-**setting Permissions Symbolically**

Set the user (user = owner) permission to read and write as follows

[ahmad@localhost ~]$ chmod u=rw file\_1

26- Set the group permission to read as follows:

[ahmad@localhost ~]$ chmod g=r file\_1

27- Set the others permission to execute as follows:

[ahmad@localhost ~]$ chmod o=x file\_1

28- Set the user permission to read and write and the group to read as follows:

[ahmad@localhost ~]$ chmod u=rw ,g=r file\_1

Note the two permissions are separated by a comma

29-**Adding Permissions**

Add execute permission to the user as follows

[ahmad@localhost ~]$ chmod u+x file\_1

30- Add (write) permission to the group and (read and write) to the others asfollows

[ahmad@localhost ~]$ chmod g+w,o+rw

31-**Removing Permissions:-**

Remove execute permission from the user as follows

[ahmad@localhost ~]$ chmod u-x file\_1

32-Remove (write) permission from the group and (read and write) from the others as follows

[ahmad@localhost ~]$ chmod g-w,o-rw file\_1

33-create a new folder

[ahmad@localhost ~]$ mkdir test-folder

34- Remove all permissions

[ahmad@localhost ~]$ chmod -rwx test-folder

35- Try to:

1. list the contents

2. create a new file inside that folder

3. enter the folder

You’ll always get ( Permission denied )

ls test-folder [READ]

touch test-folder/test-file [WRITE]

cd test-folder [EXECUTE]

4. Add [READ]

[ahmad@localhost ~]$ chmod u+r test-folder

[ahmad@localhost ~]$ ls test-folder

5. Add [EXECUTE]

[ahmad@localhost ~]$chmod u+x test-folder

[ahmad@localhost ~]$cd test-folder

6. Add [WRITE]

[ahmad@localhost ~]$chmod u+w test-folder

[ahmad@localhost ~]$touch test-folder/test-file

Note:If you apply WRITE before EXECUTE it will NOT work

36-To search for a specific file inside of a directory

find directory\_path -name “file\_name” (syntax)

[ahmad@localhost /home/ahmad]$ find ahmad -name “file\_300”

Note:If you don’t specify the directory, the find command will automatically

search in your current directory

[ahmad@localhost /home]$ find -name “file\_300”

To perform a specific operation on the returned results use the -exec option

find directory\_path -name “file\_name”- exec (syntax)

37- To copy all of the pdf files ( the ones inside /usr ) into /home/ahmad home directory, use the following command

[ahmad@localhost /]$ find /usr -name “\*.pdf” -exec cp {} /home/ahmad \;

**Searching by size**

38-Search for files with size equal to 50 Bytes

[ahmad@localhost /]$ find /usr -name “\*.pdf” -size 50c

39-Search for files with with size higher (Note the plus sign) than 50 Kilo Bytes

[ahmad@localhost /]$ find /usr -size +50k

40-Search for files with with size lower (Note the minus sign) than 50 MegaBytes

[ahmad@localhost /]$ find /usr -size -50M

41-Search for files with with size lower (Note the minus sign) than 50 GigaBytes

[ahmad@localhost /]$ find /usr -size -50G

**Searching for directories**

42-Search for directories instead of files using the “-type” option

[ahmad@localhost /]$ find /usr -type d -name ”a\*”

[ahmad@localhost /]$ find /usr -iname ”a\*”

Note:The previous command searches for files starting with either the small letter “a” or the capital letter “A”

The “Locate” command is also used for searching files and directories

43-If you create a new file the locate command will not find it until you update the database,

using the command updatedb (Note, you have to login as a root user first to be able to update the database)

[ahmad@localhost /home]$ touch file\_400

[ahmad@localhost /home]$ locate “file\_400” ==> Nothing is found

[ahmad@localhost /home]$ su ==> Login as root

[root@localhost /home]$ updatedb

[root@localhost /home]$ exit ==> returns to original user

[ahmad@localhost /home]$ locate “file\_400” ==> Returns the path for the file

SHELLS

43-To find all of the available shells on your system

[ahmad@localhost /home]$ cat /etc/shells

Output: /bin/sh

/bin/bash

/bin/tcsh

/bin/cs

44-To use a specific shell temporarily, write the full path of the shell.

[ahmad@localhost /home]$ /bin/sh

45-To go back to the previous shell, just write “exit​” or hit ( ctrl + D).

[ahmad@localhost /home]$ exit

46-To use a specific shell permanently​, write use the command “chsh​”(short for: change shell).

[ahmad@localhost /home]$ chsh

Changing shell for ahmad.

New shell [/bin/bash]: ​/bin/sh

Password:

Shell changed.

47-To find the name of the current shell

[ahmad@localhost /home]$ echo​ $SHELL

Notice the $​ and that the word “SHELL​” is all in CAPS.

48-Define a variable by writing the variable name followed by an equals sign​ followed by the value ​(integers, strings).

[ahmad@localhost /home]$ a=hello-world

don’t need to add double quotes as long as the value consists of one word with no spaces​.

[ahmad@localhost /home]$ b=”Hello World”

Double quotes were added because the value consists of two words separated by a space​.

[ahmad@localhost /home]$ c=19

49-Print the value of a variable using the echo command followed by a dollar sign followed by the variable name​.

[ahmad@localhost /home]$ echo $a output :hello-world

[ahmad@localhost /home]$ echo $b output: Hello World

[ahmad@localhost /home]$ echo $c output: 19

50-The output is a list of colon(:) separated list of directories in which the shell looks for commands.

echo $PATH

51-prompt settings

echo $PS1

52-current Shell

echo $SHELL

53-to print all the environment variables

[ahmad@localhost /home]$ printenv

**Arithmetic Operation**

54-Arithmetic operations should be wrapped inside of double parenthese ==> c=$(( a + b ))

[ahmad@localhost /home]$ a=3

[ahmad@localhost /home]$ b=6

[ahmad@localhost /home]$ c=$(( a + b ))

[ahmad@localhost /home]$ echo $c output: 9

Notice the increment ​and decrement ​operators are also available.

[ahmad@localhost /home]$ a=3

[ahmad@localhost /home]$ echo $a output: 3

[ahmad@localhost /home]$ (( a++ ))

[ahmad@localhost /home]$ echo $a output:4

[ahmad@localhost /home]$ (( a-- ))

[ahmad@localhost /home]$ echo $a output: 3

Notice that floating point math is not available

[ahmad@localhost /home]$ d=$(( 1/3 ))

[ahmad@localhost /home]$ echo $d unexpected output: 0

55-Steps of creating and running a bash shell script

1.[ahmad@localhost /home]$ touch​ first\_script.sh

2.[ahmad@localhost /home]$ nano​ first\_script.sh

3.Add ( #!/bin/bash​ ) to the first line of your file.

Move to the next line and add some commands.

--------------------------------

| #!/bin/bash |

|file\_name=”Ahmad” |

|touch $​file\_name |

| |

-------------------------------

Save → ctrl + o

Exit → ctrl + x

4. to run write in terminal

bash first\_script.sh​

56-check by using (if statment and square bracket[]) if x is less than 50 print value of x is less than 50 else print value of x is not less than 50 (x=15):

x=15

if [ $x -lt 50 ] # single Brackets

then

echo “$x ​is less than 50”

else

echo “$x ​is NOT less than 50”

fi

57-check by using (if statment and circle bracket()) if x is less than 50 print value of x is less than 50 else print value of x is not less than 50 (x=15);

x=15

if (( $x < 50 )) # # integer comparison

then

echo “$x ​is less than 50”

else

echo “$x ​is NOT less than 50”

fi

58-what is the output of the following code:-

x=100

if [[ $x < 20​ ]];then

echo “$x ​is less than 20”

else

echo “$x ​is NOT less than 20”

fi unexpected output: print “ 100 is less than 20 ”

59-using if statment and double bracket to check string:

if [[ “cat” == “cat” ]]

then

echo “True”

else

echo “False”

fi

60-using if statment and single bracket to check string

Note:If he use ( )it will not work

if [ “cat” == “cat” ] # RIGHT

then

echo “True”

else

echo “False”

fi

61- write example for while loop

i=0

while [[ $i -lt 10 ]]

do

echo “i now is ”$i

(( i++ ))

done

62- write example for until loop

i=0

until [[ $i -g​t 10 ]] ; do

echo “i now is ”$i

(( i++ ))

done

63-what is the output of the following code:-

for i in 1 2 3

do

echo “i now is ”$i

done output :i now is 1

i now is 2

i now is 3

64-what is the output of the following code:-

for i in {1..3}

do

echo “i now is ”$i

done output :i now is 1

i now is 2

i now is 3

65-what is the output of the following code:-

for i in {1..8..3}

do

echo “i now is ”$i

done output: i now is 1

i now is 4

i now is 7

66-example for OUTPUT​ of another command is input of other command

for i in $( ls )

do

echo “current file is: ”$i

done