Nakerah Network:

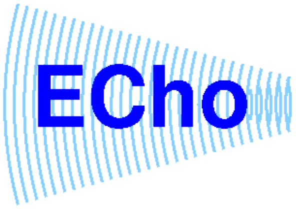
Hassan Al Achek

Github: https://github.com/Hassan-Al-Achek
Linked In: https://www.linkedin.com/in/hassan-al-achek-27b2b1204/
Twitter: https://twitter.com/HassanAlachek

Linux Fundamentals:

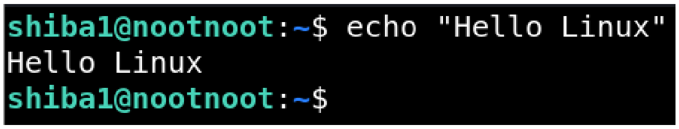
Linux Fundamentals 0x01:

Echo:



Echo command: Think about echo as literal meaning of the word, when you say hello, and you hear your hello back, okay the same apply here

~$ echo “hello Linux”
# will simply give you “Hello Linux”



```
shiba1@nootnoot:~$ echo "Hello Linux"
Hello Linux
shiba1@nootnoot:~$
```

Figure -1-

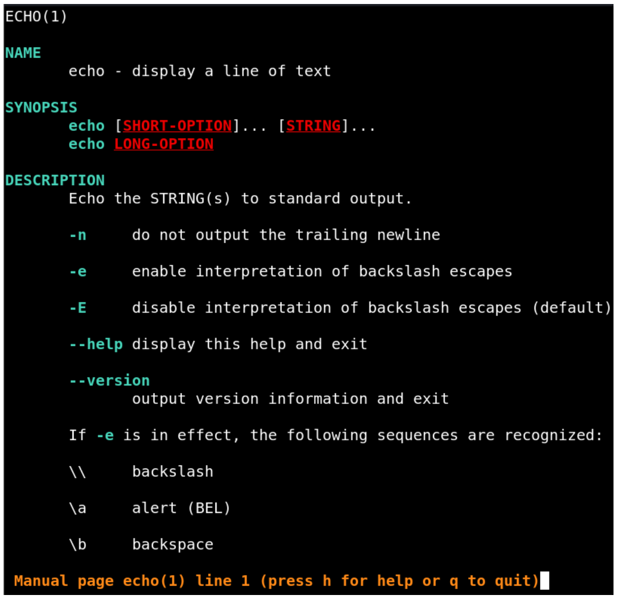
Manual Pages and Flags:

As you know, when you buy something new. As a first step you will read the manual :P but yeah, as we know the majority of us will only read a few lines and then go to try. So the manual will be the reference that will help us understand more about the functionality of a specific command.

To display the manual page use the following command
~$ man <command>
<command> can be any command but they need to have a manual page
For example:
~$ man echo
~$ man man
Note: Press q to quit :)
From the manual page:

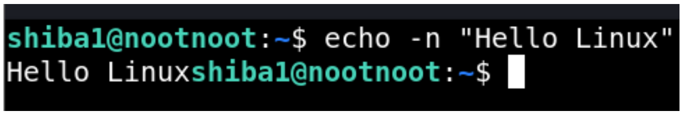
- Brief description of the command
- Synopsis
- Flags or options with description

Flags are may accept argument or not



```
ECHO(1)

NAME
       echo - display a line of text

SYNOPSIS
       echo [SHORT-OPTION]... [STRING]...
       echo LONG-OPTION

DESCRIPTION
       Echo the STRING(s) to standard output.

       -n     do not output the trailing newline

       -e     enable interpretation of backslash escapes

       -E     disable interpretation of backslash escapes (default)

       --help display this help and exit

       --version
              output version information and exit

       If -e is in effect, the following sequences are recognized:

       \\     backslash

       \a     alert (BEL)

       \b     backspace

 Manual page echo(1) line 1 (press h for help or q to quit)
```

Okay if we play with the manual page of the echo command we can recognize a flag “-n” which is used to trim the new line at the end of the echoed string, as you can see in the screenshot below.



```
shiba1@nootnoot:~$ echo -n "Hello Linux"
Hello Linuxshiba1@nootnoot:~$
```

Figure-2-

So in general commands in linux have the following form:

<command> <flags> <arguments>

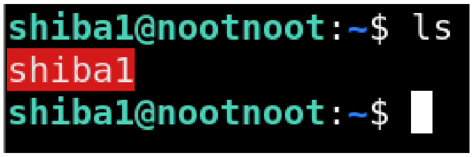
If you want to display brief help page you can use the following command:
<command> -h or <command> --help

~$ echo -n hello

List Command:

List command will display a list of files and directories that exist in your Current directory.

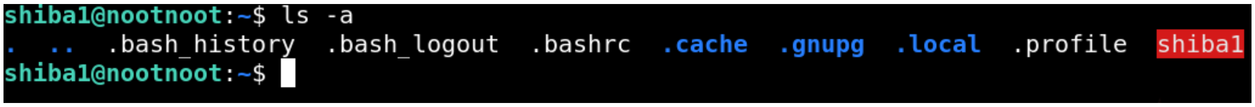
~$ ls



```
shiba1@nootnoot:~$ ls
shiba1
shiba1@nootnoot:~$
```

Figure -3-

If you visit the ls manual page you will notice different flags or options, as in windows there are hidden files and directories in linux. This is done by preceding the name of the file or directory by “.”, so by making normal “ls” command this files and directories will not shown in the output, to display hidden file we will use the “-a” flag which stand for “all”

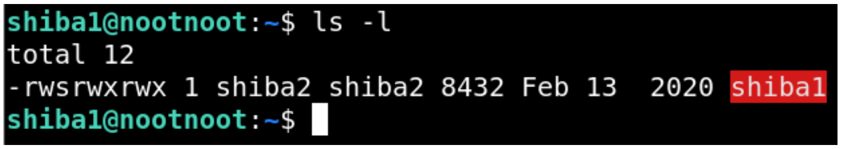


```
shiba1@nootnoot:~$ ls -a
.  ..  .bash_history  .bash_logout  .bashrc  .cache  .gnupg  .local  .profile  shiba1
shiba1@nootnoot:~$
```

Figure -4-

To display a long list format we will use the “-l” option, this will give us a detailed output describing permissions set in directories and files, as well as the owner and group, and date of last update affecting these files or directories.

Note: From the future this option will help you with the part of privilege escalation, or to control files and directories permission to detect evil permissions that may harm your system.



```
shiba1@nootnoot:~$ ls -l
total 12
-rwsrwxrwx 1 shiba2 shiba2 8432 Feb 13  2020 shiba1
shiba1@nootnoot:~$ 
```

Figure -5-

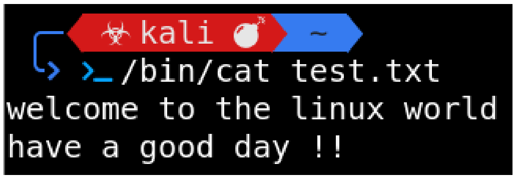
Cat:

Not really a cat :), we are not in a zoo.

Cat command used to display the content of a file to the “stdout”

Stdout: standard output (the terminal in our case)

~$ cat <file>



```
kali  ~
>_/bin/cat test.txt
welcome to the linux world
have a good day !!
```

Figure -6-

The content of test.txt:

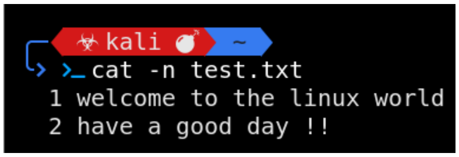
----

welcome to the linux world
have a good day !!

~$ cat --help

After parsing the displayed output we can recognize that the “-n” or “--number” which stand for numbering

~$ cat -n test.txt



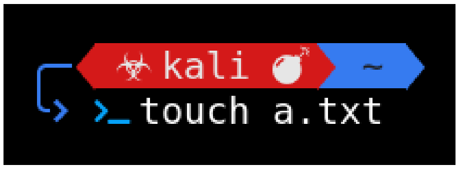
```
 kali  ~
>_cat -n test.txt
  1 welcome to the linux world
  2 have a good day !!
```

Figure -7-

Touch:

Touch command used to create a file

~$ touch <filename>
~$ ls # to see the new created file
Note: the # used mean that anything after the hash symbol will be considered as comment

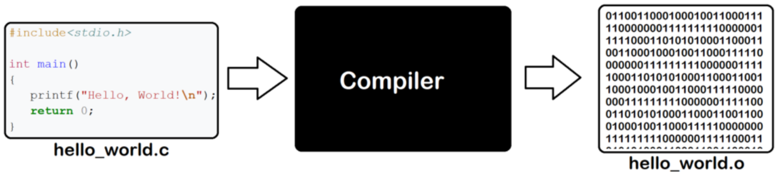


```
 kali  ~
>_touch a.txt
```

Figure -8-

Running a Binary:

A Binary means a program after compilation, this binary can run. By run i mean it can load itself in the memory (RAM) and run as a process.



```
#include<stdio.h>

int main()
{
   printf("Hello, World!\n");
   return 0;
}
```

```
01100110001000100110001111
11000000011111111110000001
11110001101010100011000011
00110001000100110001111110
00000011111111110000001111
10001101010100011000110001
10001000100110001111110000
00011111111110000001111100
01101010100011000110011100
01000100110001111110000000
11111111110000001111100011
```

So simply a binary in linux is like exe in windows system.

In linux there is two way to run an executable binary, the first consists of providing the full path to the binary which known by absolute path (e.g /home/kali/binary) and relative path (e.g ./binary)

Let's suppose we have the following hierarchy and we want to access Story.txt and our working directory is the project7 directory.

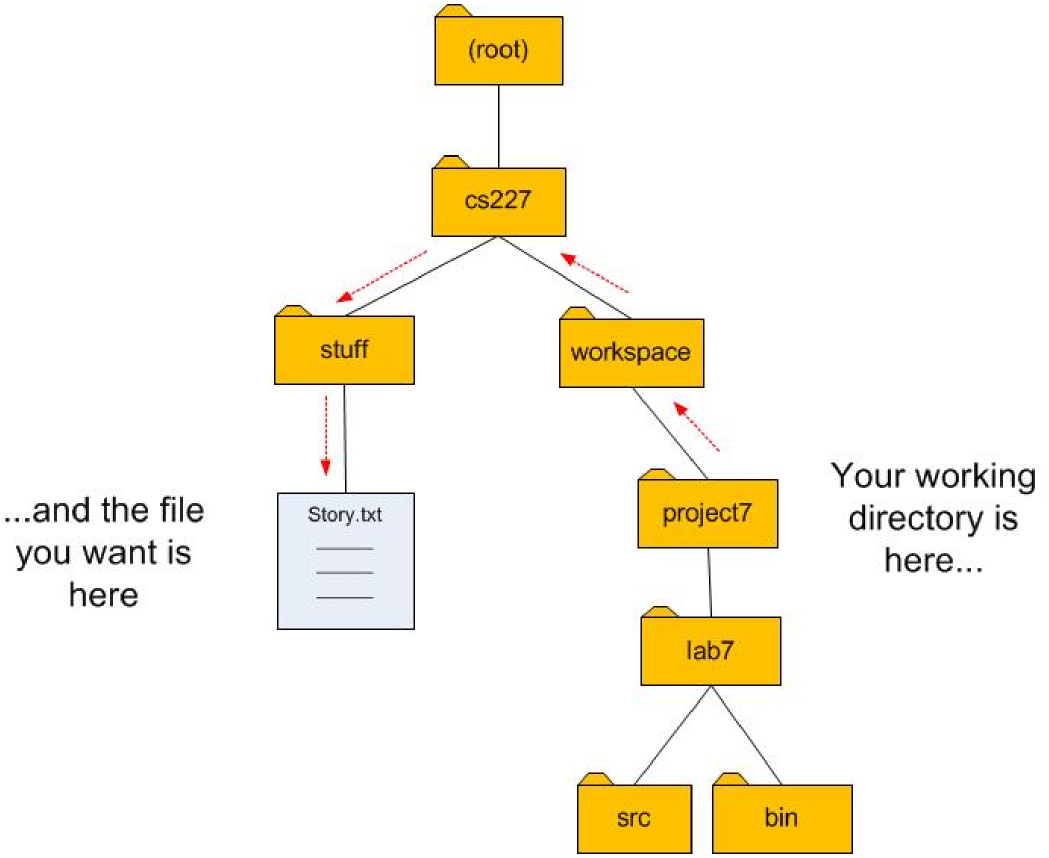

(root)
cs227
stuff
workspace
Story.txt
project7
lab7
src
bin
...and the file you want is here
Your working directory is here...

Figure -9-

If we want to access the Story.txt without change directory we will use absolute path:

```
kali@kali /cs227/workspace/project7 $ cat /cs227/stuff/Story.txt
```

If we want to use relative path, we need to change our working directory to stuff:
First let’s discover the “cd” command which stands for Change Directory.

In linux there are two dotted directories “.” and “..”

“.”: mean current directory
“..”: mean the parent directory

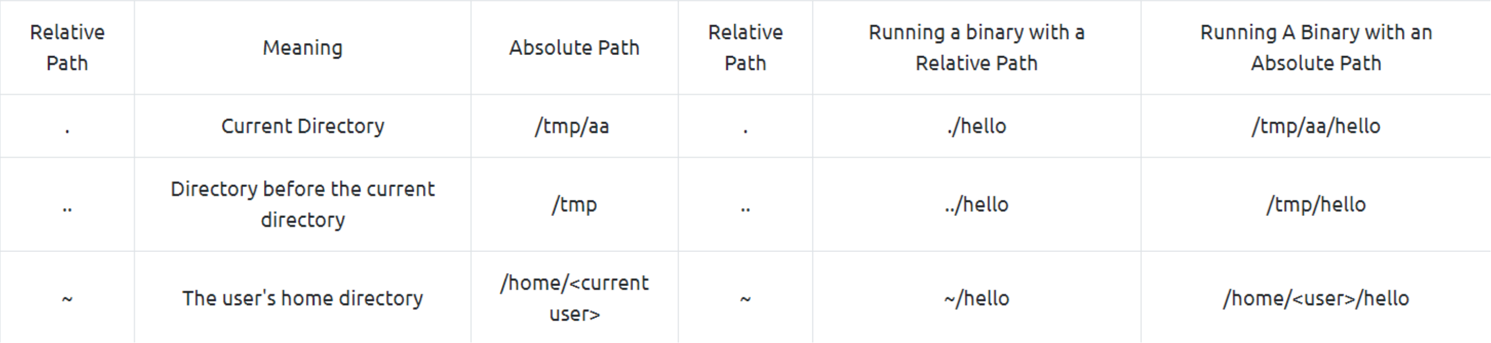
For example if we are in project7 and we want to go back one directory we will enter the following command:

```
kali@kali /cs227/workspace/project7 $ cd ..
kali@kali /cs227/workspace $
```

If we want to enter project7 back we will use the following command:

```
kali@kali /cs227/workspace $ cd project7
kali@kali /cs227/workspace/project7 $ cd ../../
kali@kali /cs227 $ cd stuff
kali@kali /cs227/stuff $ cat Story.txt # relative
```

So now after understanding the relative and absolute path we are able to run binary from either absolute or relative perspective.



Relative Path	Meaning	Absolute Path	Relative Path	Running a binary with a Relative Path	Running A Binary with an Absolute Path
.	Current Directory	/tmp/aa	.	./hello	/tmp/aa/hello
..	Directory before the current directory	/tmp	..	../hello	/tmp/hello
~	The user's home directory	/home/<current user>	~	~/hello	/home/<user>/hello

To run a binary (that print Hello world) from the relative path perspective (we are in the same directory of the binary)

```
~$ ls
binary
~$ ls -a
. .. binary
~$ ./binary # relative
Hello world
```

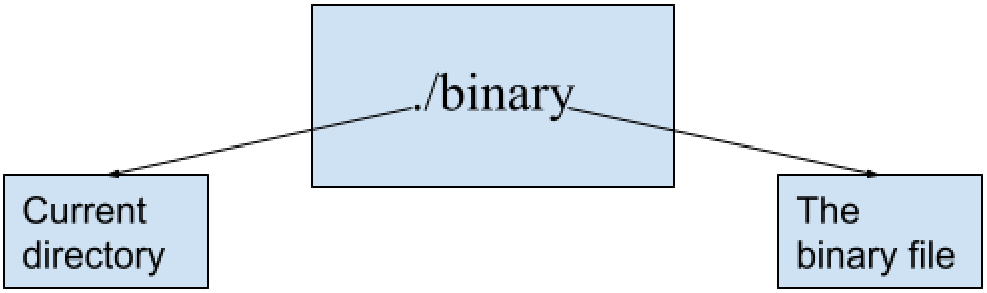


Figure -10-

To run a binary from the home directory from anywhere in the linux system we can use the following command:

```
/home/kali/Myfile $ ~/binary
Hello World
/home/kali/Myfile $
```

To run a binary that exist in the parent directory of our current directory we will use the following command:

```
/home/kali/Myfile $ ls
binary  New
/home/kali/Myfile $ cd New

/home/kali/Myfile/New $ ls -a
.  ..
/home/kali/Myfile/New $ ../binary
Hello World
```

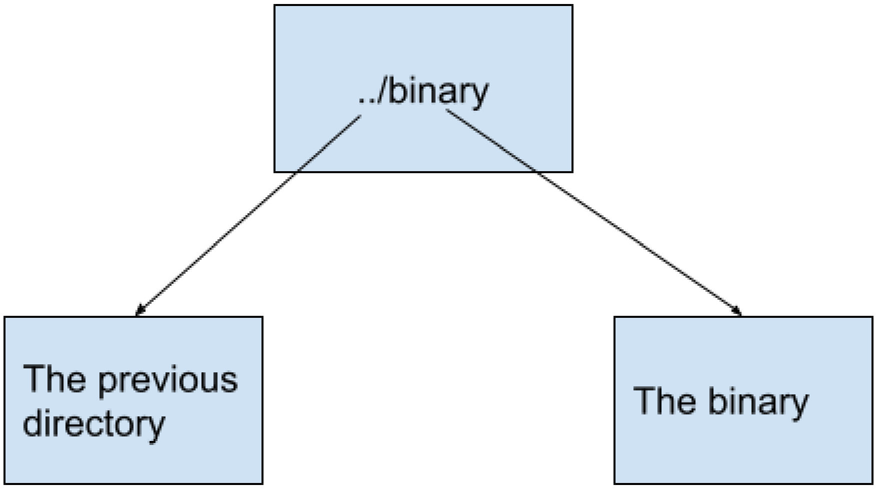
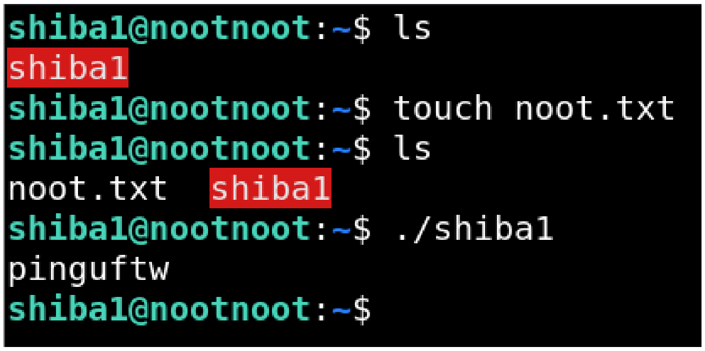


Figure -11-

To solve the challenge we need to create a file (noot.txt), and as we know we will use the “touch” command:

Note: kali@kali~:$ ssh shiba1@10.10.4.50 to connect via ssh to the machine

```
~$ touch noot.txt
~$ ./shiba1
pinguftw
~$
```



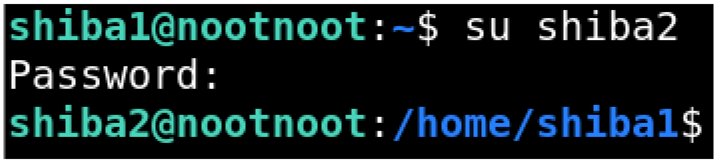
```
shiba1@nootnoot:~$ ls
shiba1
shiba1@nootnoot:~$ touch noot.txt
shiba1@nootnoot:~$ ls
noot.txt  shiba1
shiba1@nootnoot:~$ ./shiba1
pinguftw
shiba1@nootnoot:~$
```

Switch User:

As you can see, at this step we have a password for the level two user “shiba2”. So if we want to switch to another user in linux we will use the following command:

su <username>

In this case we will run:
~$ su shiba2
Password: pinguftw



```
shiba1@nootnoot:~$ su shiba2
Password:
shiba2@nootnoot:/home/shiba1$
```

Note: password will not displayed while typing in linux

Linux Fundamentals 0x02:

SSH:

The Secure Shell Protocol (**SSH**), we will use ssh to get a remote command line into the machine To connect to the machine use the following command (If you are in linux):

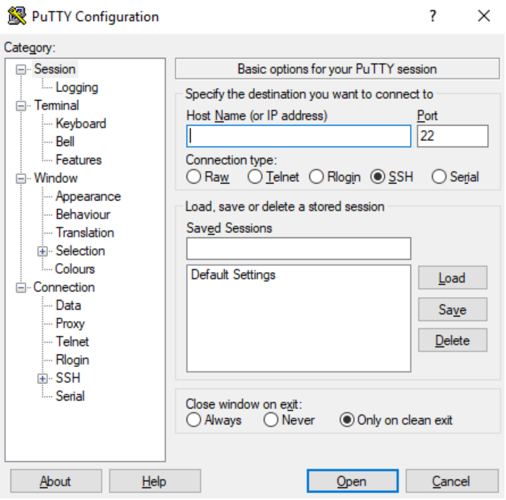
ssh <username>@<machineIP>
And then enter the password needed to get a remote session in the machine.

If you are in windows you can use a GUI (graphical user interface) program like putty

Putty Download:
https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

Once you finish download and open the program you will have the following GUI:

Note: make sure the the ssh radio button is clicked then The ssh default port is 22



Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address):
Port: 22

Connection type:
○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

Load, save or delete a stored session

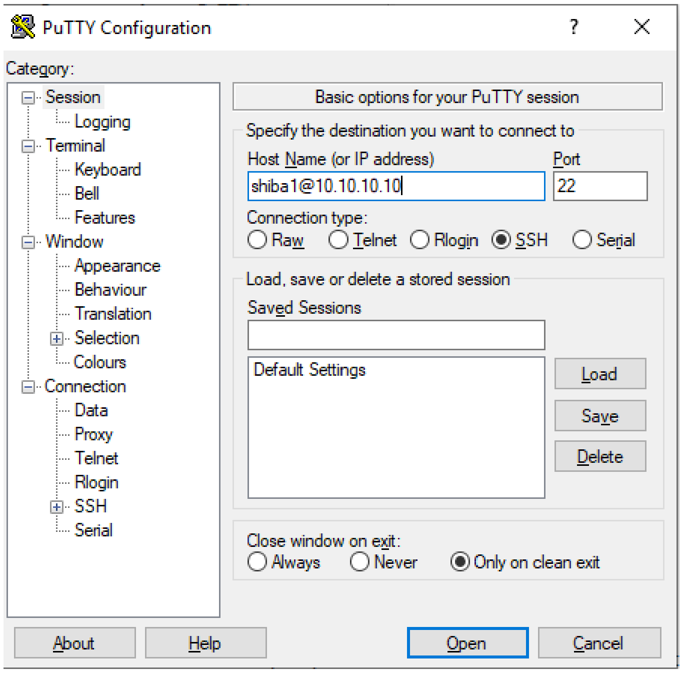
Saved Sessions

Default Settings

Close window on exit:
○ Always ○ Never ● Only on clean exit

Figure -12-

Enter the <username>@<machineIP>


PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): shiba1@10.10.10.10

Port: 22

Connection type: ○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

Load, save or delete a stored session

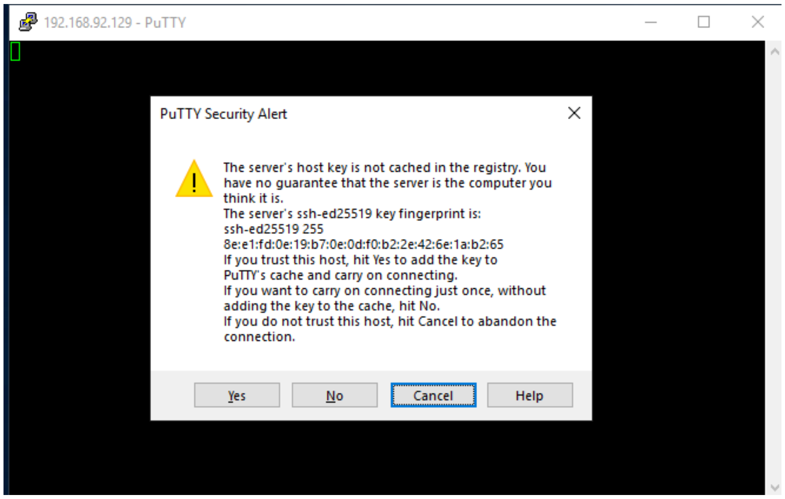
Saved Sessions

Default Settings

Close window on exit: ○ Always ○ Never ● Only on clean exit

Figure -13-

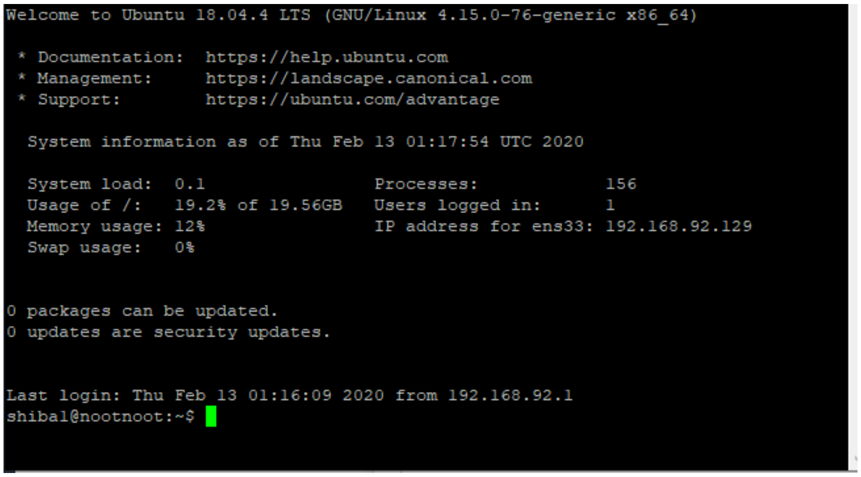
By clicking open you will prompted by the following photo:


192.168.92.129 - PuTTY

PuTTY Security Alert

The server's host key is not cached in the registry. You have no guarantee that the server is the computer you think it is.
The server's ssh-ed25519 key fingerprint is:
ssh-ed25519 255
8e:e1:fd:0e:19:b7:0e:0d:f0:b2:2e:42:6e:1a:b2:65
If you trust this host, hit Yes to add the key to PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without adding the key to the cache, hit No.
If you do not trust this host, hit Cancel to abandon the connection.

Click yes, and then enter password of shiba2 user



```
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-76-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu Feb 13 01:17:54 UTC 2020

  System load:  0.1               Processes:            156
  Usage of /:   19.2% of 19.56GB   Users logged in:      1
  Memory usage: 12%               IP address for ens33: 192.168.92.129
  Swap usage:   0%


0 packages can be updated.
0 updates are security updates.


Last login: Thu Feb 13 01:16:09 2020 from 192.168.92.1
shibal@nootnoot:~$ 
```

Figure -14-

Linux Operators:

&&:

If you are familiar with programming language like C language, you will recognize that && mean “and”, anyway “&&” mean “and” and as we know that the truth table of “and” is the following:

X	**Y**	**Z = X and Y**
True	True	True
True	False	False
False	True	False
False	False	False

So from this table we are able to know that if we use that in linux and suppose that X is command1 and Y is command2, and we run “command1 && command2”, command2 will not run if command1 is false.

Example:
ls && echo Hello
Will work fine

But ajsnjdnadbhad && echo Hello
Will not work, because command1 is a random character from my keyboard

&:

The “&” operator used to send a command (which is in reality a process that runs in the ram, a process meaning a program in execution) to the background, in the operating system there are different types of processes with different priorities.

If a command will take time to execute we will need to wait for that command until it finishes execution, because we are unable to make a command in this active session. Using the & operator we send the command to the background while it is still running but now we are able to make a command while the previous running.

```
~$ sleep 10
 #Wait 10 s before use the terminal again
~$ sleep  10 &
~$ ls
```

$:

This operator used to access variable in general (bash scripting), but here first we will use it to display the environment variables, this variables are set by the operating system by default and used by the OS to manage process (there is a lot of future related to environment variables but we don’t need to know a lot about them at this moment)

For example an environment variable PATH is a specific variable used to store the path that tells the system where to look for commands that can be executed anywhere in the system.

To display PATH variable use:

```
~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
~$

~$ echo $USER
shiba2
```

We can use the environment variable as input for others commands for example:

~$ touch $USER
This command will create a file that have as name the username
~$ls
shiba2

To set an environment variable all what we need is:

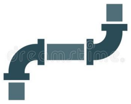
export <variable name>=<value>

Note: make sure to don't make space between <variable name> and the equal sign, and the equal sign and the <value> from the right hand part.

Example:

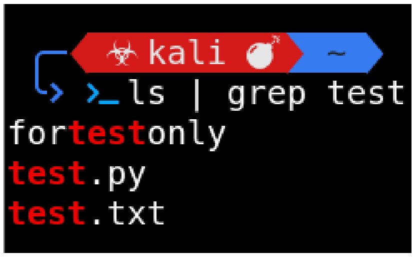
```
~$ export nootnoot=1111
~$ echo $HOME
/home/shiba2
```

|:



Pipe "|", pipe used to redirect standard output.
Using pipe we will be able to redirect output of one command and make it input to another command.
<first command> | <second command>

For example:
If we want to filter the "ls" output to check if a file or directory with a specific name exists or not, we will use a new command "grep". grep is a powerful text manipulation command, it will filter all lines that have the keyword and display them to standard output.



```
 ☣ kali   ~
 >_ls | grep test
fortestonly
test.py
test.txt
```

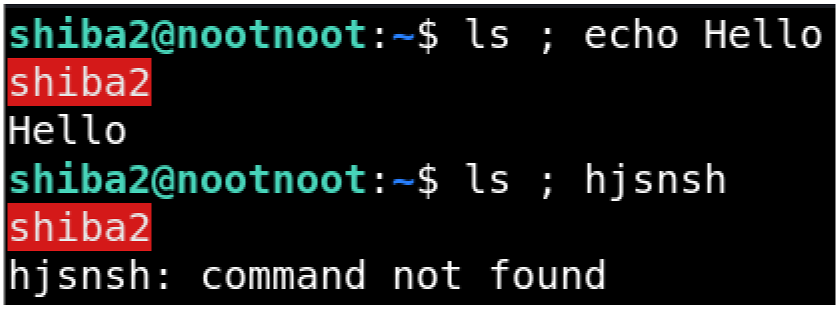
Figure -15-

If we want to filters the output of cat command to get all lines that contains a specific keyword use the following command:
~$ cat test.txt | grep hello

“;”:

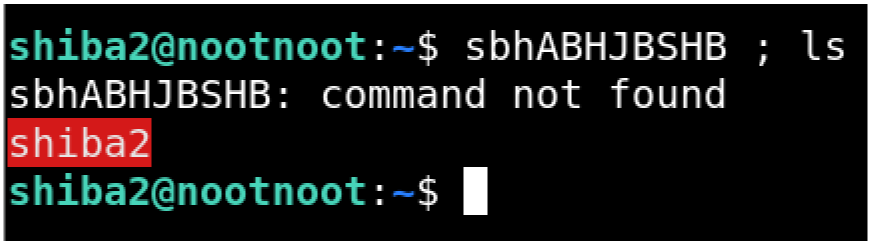
The semicolon used to take the benefit of multiprogramming and multitasking. In modern operating systems we can run multiple processes in the same time. Okay now if we want to run multiple commands, we will separate them by “;”
<command one > ; <command two>
It doesn’t matter if the command one executes successfully or not. Anyway command one or two will run concurrently.

For example:
ls ; echo Hello
ls ; sjnsns
Bdhbdhbah; ls



```
shiba2@nootnoot:~$ ls ; echo Hello
shiba2
Hello
shiba2@nootnoot:~$ ls ; hjsnsh
shiba2
hjsnsh: command not found
```

Figure -16-



```
shiba2@nootnoot:~$ sbhABHJBSHB ; ls
sbhABHJBSHB: command not found
shiba2
shiba2@nootnoot:~$ 
```

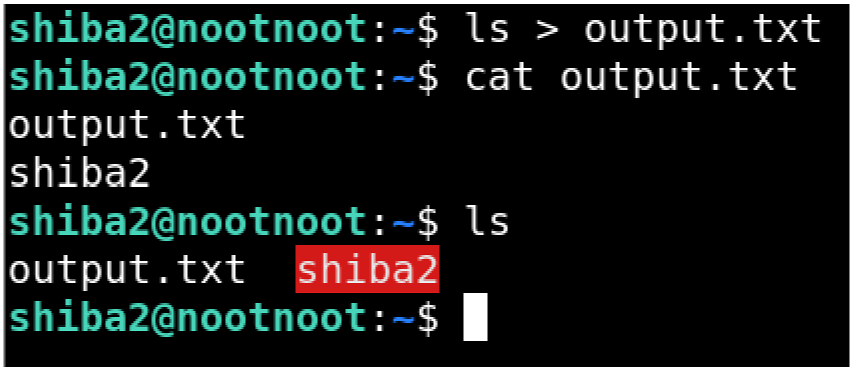
Figure -17-

“>”:

Greater than operator can be used to redirect standard output (your terminal) to a file.

For example if we want to save the output of ls command we can execute the following command:

~$ ls > output.txt
~$ cat output.txt   #to check the content of output.txt
Note: It's not necessary to have output.txt before executing the command, it will be created by the command.

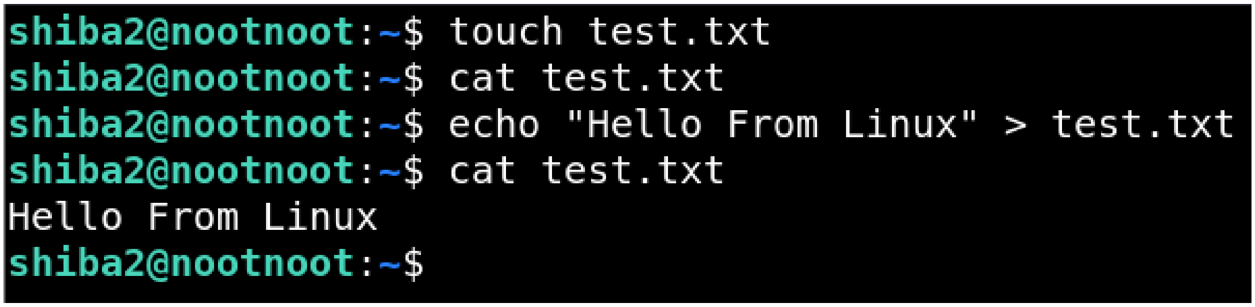


```
shiba2@nootnoot:~$ ls > output.txt
shiba2@nootnoot:~$ cat output.txt
output.txt
shiba2
shiba2@nootnoot:~$ ls
output.txt  shiba2
shiba2@nootnoot:~$ 
```

Figure -18-

Of Course we can create a file with "touch" and then write phrase into that file:

~$ touch test.txt
~$ cat test.txt # empty
~$ echo "Hello From Linux" > test.txt
~$ cat test.txt
Hello From Linux
~$



```
shiba2@nootnoot:~$ touch test.txt
shiba2@nootnoot:~$ cat test.txt
shiba2@nootnoot:~$ echo "Hello From Linux" > test.txt
shiba2@nootnoot:~$ cat test.txt
Hello From Linux
shiba2@nootnoot:~$
```

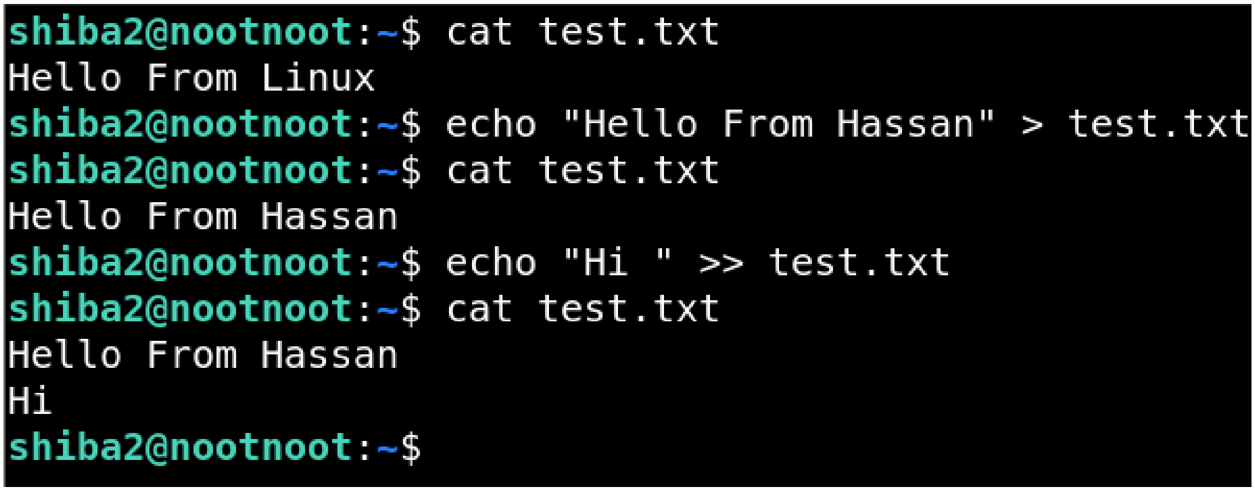
Figure -19-

~$ echo twenty > test

“>>”:

Similar to “>” but now if we have a file test.txt that contain: Hello From Linux
And execute the following command and we want to add new line into the test.txt file without wapping “Hello From Linux” we will not use “>” we will use “>>”, because “>” will clear the content of test.txt and write the new text.

```
~$ cat test.txt
Hello From Linux
~$ echo “Hello From Hassan” > test.txt
~$ cat test.txt
Hello From Hassan
~$ echo “Hi ” >> test.txt
~$ cat test.txt
Hello From Hassan
Hi
~$
```



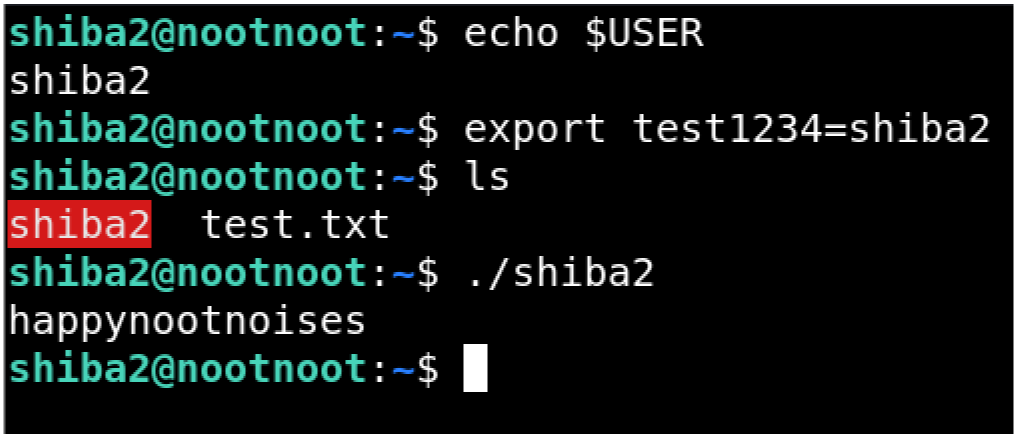
```
shiba2@nootnoot:~$ cat test.txt
Hello From Linux
shiba2@nootnoot:~$ echo "Hello From Hassan" > test.txt
shiba2@nootnoot:~$ cat test.txt
Hello From Hassan
shiba2@nootnoot:~$ echo "Hi " >> test.txt
shiba2@nootnoot:~$ cat test.txt
Hello From Hassan
Hi
shiba2@nootnoot:~$
```

Figure -20-

Binary Shiba2:

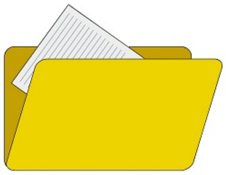
To solve the challenge, we need to reverse the functionality of the binary which check if the test123 variable is equal to USER environment variable which can be viewed by:

```
~$ echo $USER
<username>
~$
```



```
shiba2@nootnoot:~$ echo $USER
shiba2
shiba2@nootnoot:~$ export test1234=shiba2
shiba2@nootnoot:~$ ls
shiba2  test.txt
shiba2@nootnoot:~$ ./shiba2
happynootnoises
shiba2@nootnoot:~$ 
```

Figure -21-



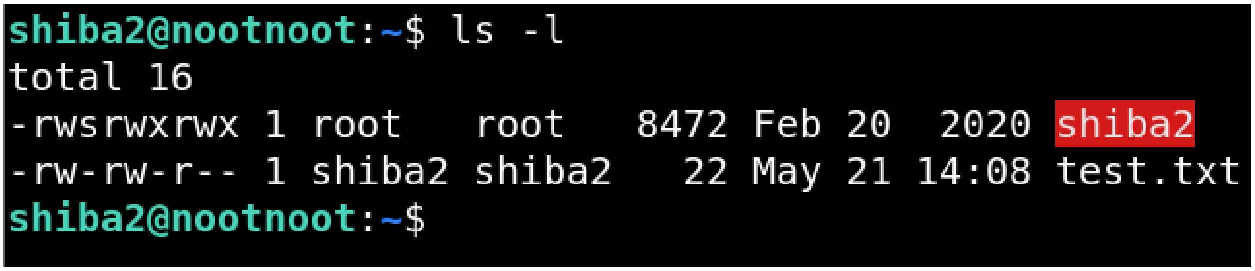
Advanced File operations:

As in the case of windows operating system, where every user has a set of permissions that allow them to read, write, execute that file or not, the same apply to directories but with some difference in the meaning of each permission.

In the next paragraph we will start learn about advanced file operations:

A Bit Of Background:

To files and directories attributes use the following command
~$ ls -al
# “a” or all and “l” to display attributes
# equivalent to ls -a -l



```
shiba2@nootnoot:~$ ls -l
total 16
-rwsrwxrwx 1 root   root   8472 Feb 20  2020 shiba2
-rw-rw-r-- 1 shiba2 shiba2   22 May 21 14:08 test.txt
shiba2@nootnoot:~$
```

Figure -22-

First 10 characters:
First character of this 10, represent the type:
"-" which stand for file
and
"d" which stand for directory

The rest 9 characters are distributed as following:

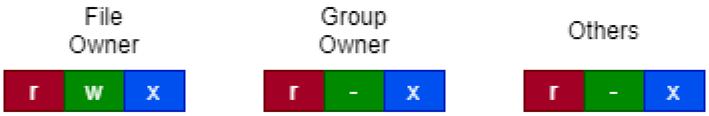
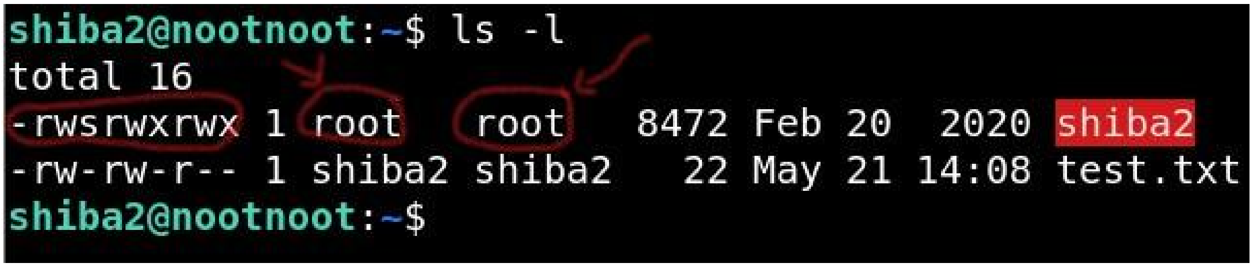


Figure -23-



```
shiba2@nootnoot:~$ ls -l
total 16
-rwsrwxrwx 1 root   root    8472 Feb 20  2020 shiba2
-rw-rw-r-- 1 shiba2 shiba2    22 May 21 14:08 test.txt
shiba2@nootnoot:~$
```

Figure -24-

We are interested in the circled output in the Figure -24-
In order: first 10 characters: Permissions, then Owner and the Group

Change Owner:

To change the owner of a file or directory, we will use the "chown" command with the general syntax:

```
~$ chown  <new owner>:<new group> filename
```

You can change only the owner of the file:

```
~$ chown <new owner> filename
```

Note: to change the permission of a file you need to have higher privilege than the original owner

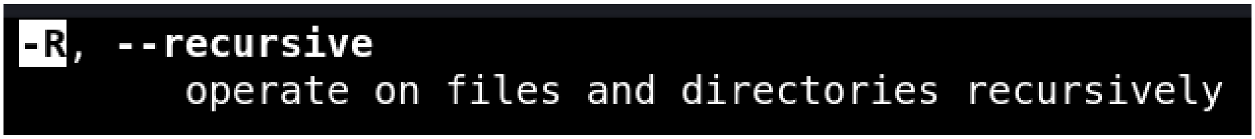
For example:
"~#" mean i am root now
~# chown shiba2:shiba2 /path/to/file1
file1 is a file owned by shiba.
~#

To change the owner of "file" to paradox we will use the following command:
chown paradox file

To change the owner and group of "file" to paradox we will use the following command:
chown paradox:paradox file

By examining the man page of chown:
~$ man chown
We recognize that "-R" is the option that allow to operate every file in directory at once
Note: to search in the man page try /<keyword>

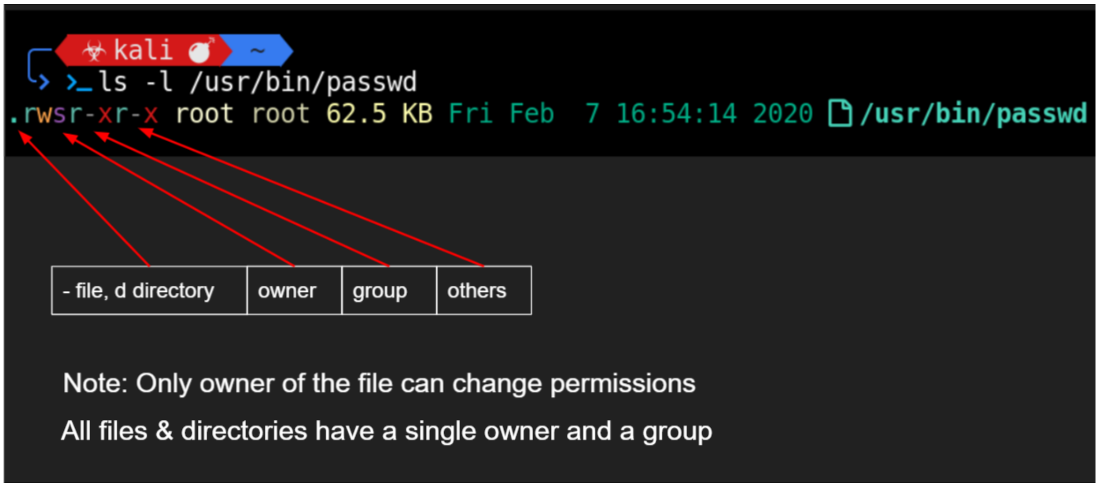
~$chown -R <new owner>:<new group> directory



```
-R, --recursive
       operate on files and directories recursively
```

Figure -25-

chmod:



```
┌ kali ~
└> ls -l /usr/bin/passwd
.rwsr-xr-x root root 62.5 KB Fri Feb  7 16:54:14 2020 /usr/bin/passwd
```

Note: Only owner of the file can change permissions

All files & directories have a single owner and a group chmod used to change permissions on files and directories and control who can read, write and execute.

```
~$ chmod <permissions> <file>
```

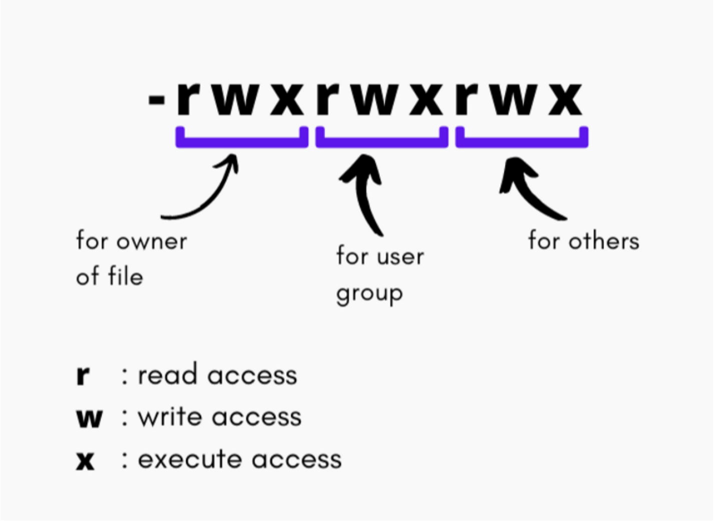


Figure -26-

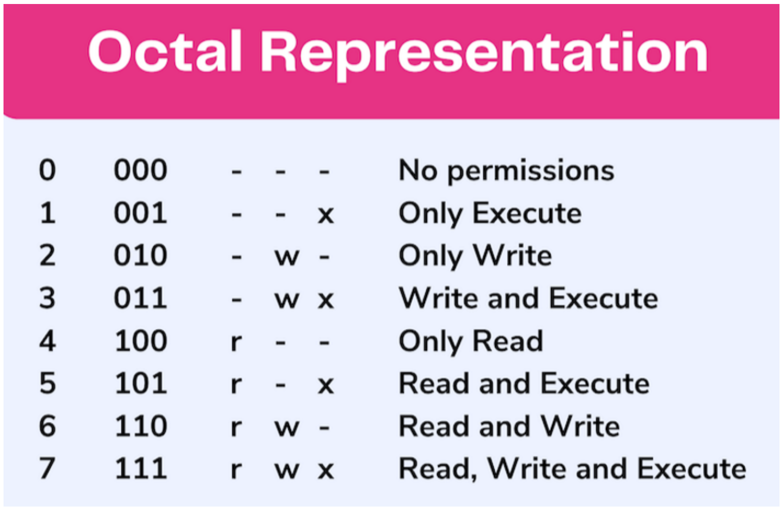
Check Figure -26- to understand the meaning of each character

There is two ways to set permissions “symbolic” and “absolute“ (numeric)

Symbolic:

- r: read
- w: write
- X: execute

Numeric: as explained in the following table



Octal Representation

0	000	-	-	-	No permissions
1	001	-	-	x	Only Execute
2	010	-	w	-	Only Write
3	011	-	w	x	Write and Execute
4	100	r	-	-	Only Read
5	101	r	-	x	Read and Execute
6	110	r	w	-	Read and Write
7	111	r	w	x	Read, Write and Execute

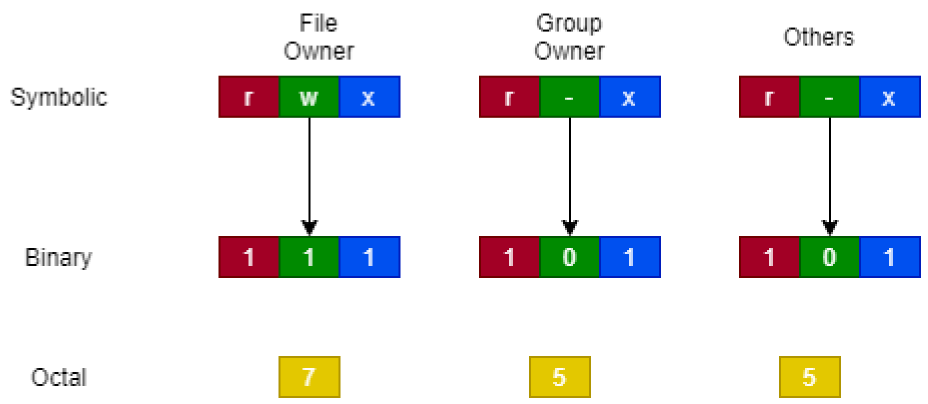


Figure -27- How to convert between symbolic and absolute

For example:
User can read
Group can read and write
Others: can’t read , write and execute

- r-- rw- ---

r-- : 100 from binary to decimal: 4
rw-: 110 from binary to decimal: 6
--- : 000 from binary to decimal: 0

Then the result is: 460

User can read, write, execute
Group can read, write, execute
Others: can read write, execute

- rwx rwx rwx

- : file
rwx: 111 from binary to decimal: 7
rwx: 111 from binary to decimal: 7
rwx: 111 from binary to decimal: 7
Then the result is: 777

Remove (rm):

Now you will be able to remove files created, Stay safe from rm -rf * :)

After you accidentally ran rm -fr /* on a production system.

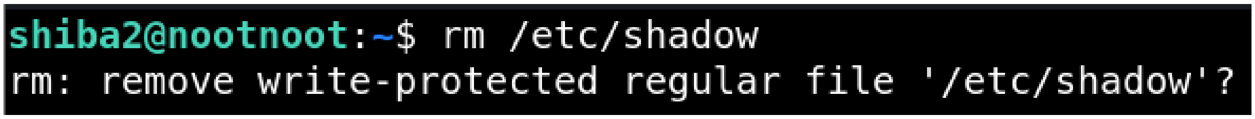
Oh no..

Don’t try at home: rm -rf * !!!

To remove a file in the current directory:

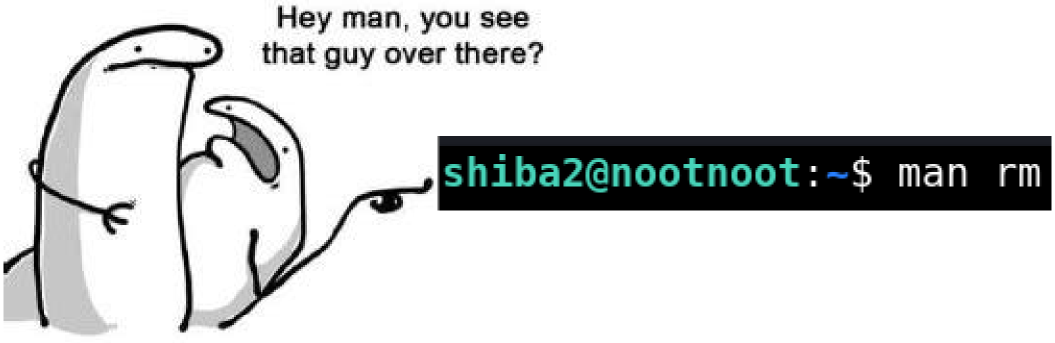
~$ rm <file name>

Note: to remove a file you need a write permission, and that is the important of permissions to prevent the harmful behavior of rm command



```
shiba2@nootnoot:~$ rm /etc/shadow
rm: remove write-protected regular file '/etc/shadow'?
```

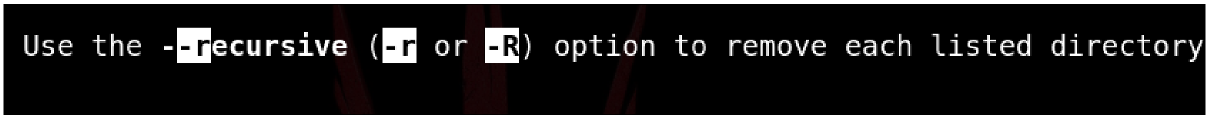
Figure -28-



```
shiba2@nootnoot:~$ man rm
```

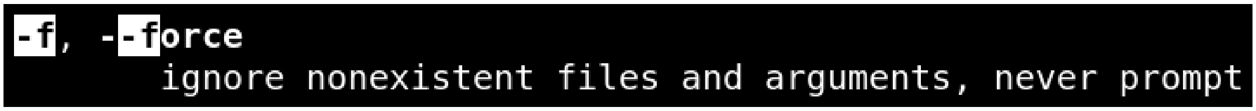
:) Okay to solve task read the manual page of rm:

Flag deletes every file in a directory: -r



```
Use the --recursive (-r or -R) option to remove each listed directory
```

How do you suppress all warning prompts: -f



```
-f, --force
       ignore nonexistent files and arguments, never prompt
```

Move (and rename) mv:

mv used to move a file from location to other, and can be used to rename a file

```
~$ mv <source file> <destination>
```

For example to move a file to tmp as indicated in the task:

```
~$ mv file /tmp
```

Linux Fundamentals 0x03:

Copy cp:

I don’t think that this needs an explanation :)

```
~$ cp <source file> <destination>
```

The difference between cp and mv that cp create an independent new file (duplicate)

Advanced File Operations:

cd && mkdir:

Folders and files in windows are similar to directories and files in linux

The “cd” command, which stands for Change Directory, allows us to change working directory.

```
~$ cd <directory>
```

For example:
To go back one step from current directory to parent of this directory we will use:

```
~$ cd ..
```

To go back to home directory from anywhere in the system we will use:

```
~$ cd ~
```

OR

```
~$ cd
```

The “mkdir” command, which stand for Make Directory, is used to create a new directory

~$ mkdir <new directory name>

For example:
Using absolute paths how would you make a directory called test in /tmp
~$ mkdir /tmp/test
Without absolute path(using relative path):
~$ cd /tmp
/tmp $ mkdir test

ln:

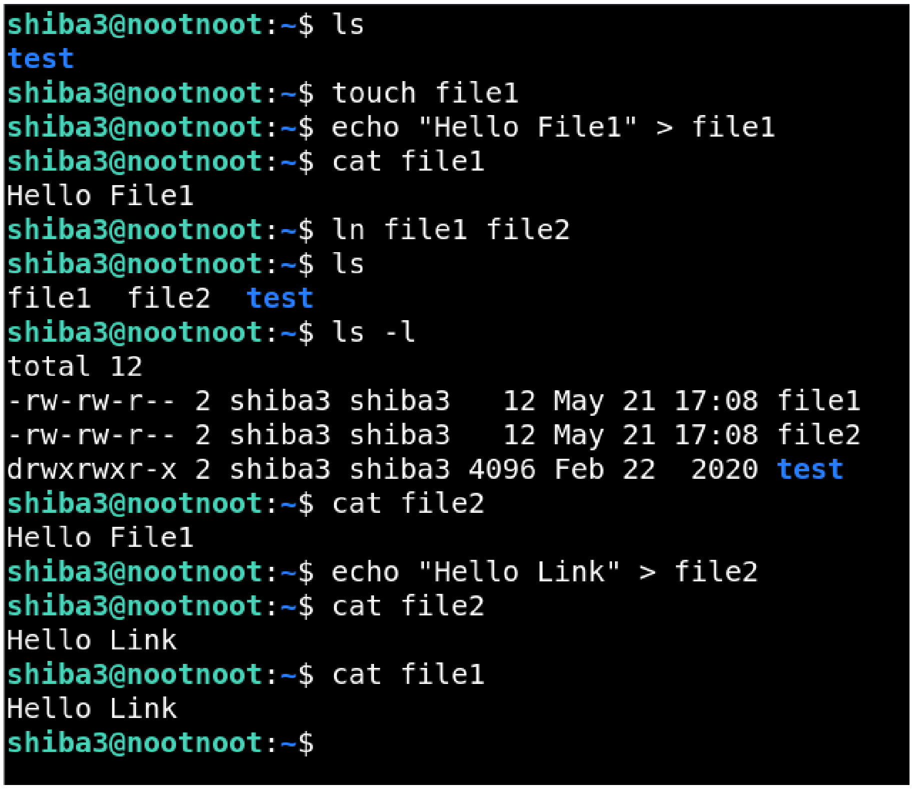
ln is used for creating a link to a file, it’s like creation of a pointer that points to a specific location which is the original file.

~$ ln <source> <destination>

There is two type of linking “hard linking” and “symbolic linking”

Hard linking:

- Everything affect the created link will affect the original (source file) and vice versa



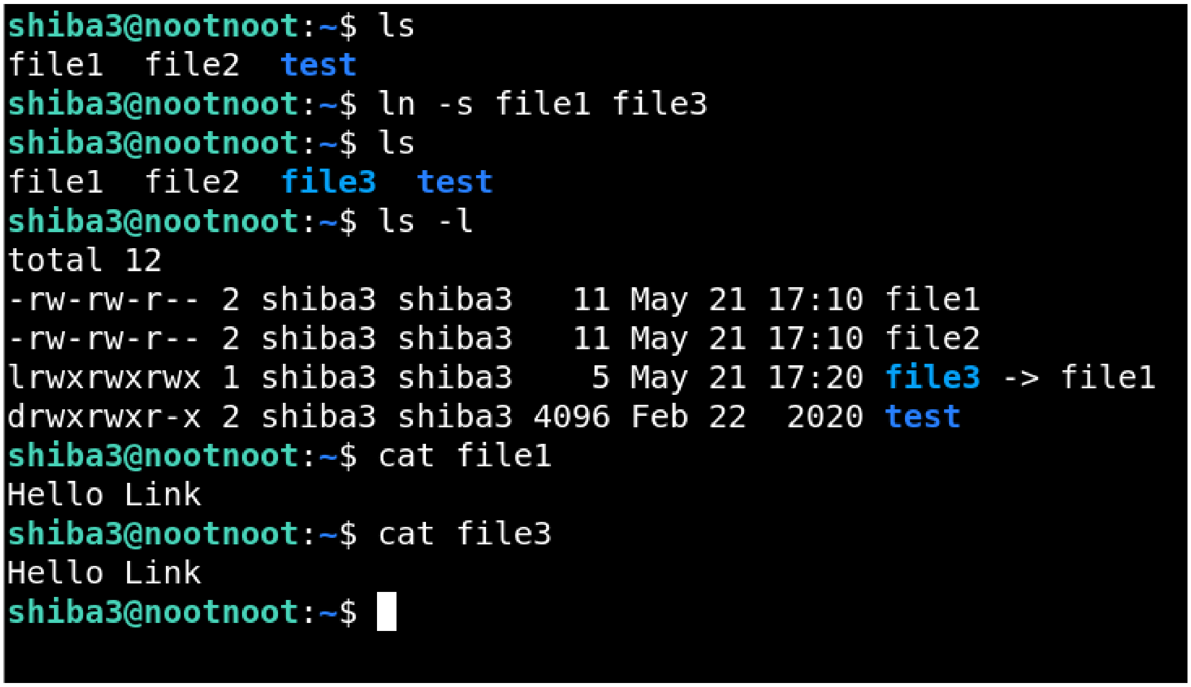
```
shiba3@nootnoot:~$ ls
test
shiba3@nootnoot:~$ touch file1
shiba3@nootnoot:~$ echo "Hello File1" > file1
shiba3@nootnoot:~$ cat file1
Hello File1
shiba3@nootnoot:~$ ln file1 file2
shiba3@nootnoot:~$ ls
file1  file2  test
shiba3@nootnoot:~$ ls -l
total 12
-rw-rw-r-- 2 shiba3 shiba3   12 May 21 17:08 file1
-rw-rw-r-- 2 shiba3 shiba3   12 May 21 17:08 file2
drwxrwxr-x 2 shiba3 shiba3 4096 Feb 22  2020 test
shiba3@nootnoot:~$ cat file2
Hello File1
shiba3@nootnoot:~$ echo "Hello Link" > file2
shiba3@nootnoot:~$ cat file2
Hello Link
shiba3@nootnoot:~$ cat file1
Hello Link
shiba3@nootnoot:~$
```

Figure -29-

Note: As you can see from Figure -29-
Hard link is like a duplicate of the original file but any change in one affect the other

Symbolic link:

- It's a symbol that represents the original file. It's not really a file that contains the same data as the original, it's a way to access the original file.
- Symbolic links are more efficient from cp, cp will create a new copy of the original file that are independent of each other. So creating a copy of a file using cp will need more space in the hardisk(Save disk space)



```
shiba3@nootnoot:~$ ls
file1  file2  test
shiba3@nootnoot:~$ ln -s file1 file3
shiba3@nootnoot:~$ ls
file1  file2  file3  test
shiba3@nootnoot:~$ ls -l
total 12
-rw-rw-r-- 2 shiba3 shiba3   11 May 21 17:10 file1
-rw-rw-r-- 2 shiba3 shiba3   11 May 21 17:10 file2
lrwxrwxrwx 1 shiba3 shiba3    5 May 21 17:20 file3 -> file1
drwxrwxr-x 2 shiba3 shiba3 4096 Feb 22  2020 test
shiba3@nootnoot:~$ cat file1
Hello Link
shiba3@nootnoot:~$ cat file3
Hello Link
shiba3@nootnoot:~$ 
```

Figure -30-

Note: the full permission of file3 doesn't mean that file3 really has full permissions. In reality they have the same permission as the original file (file1)

For example:
To create a hard link between /home/test/testfile to /tmp/test we will execute the following command:

```
~$ ln /home/test/testfile /tmp/test
```

find:

The simple use of find is to list all files and directories in a directory, it’s recursive. But it can’t bypass security permissions, that’s mean if you don’t have the permission that allows you to list files that are owned by a different find will not do magic for you :) .

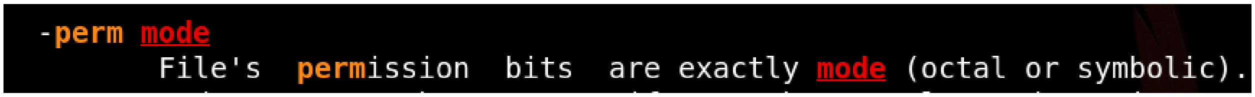
The power of find reside in parameters
For example:
To find files and directories owned by user use the following command:

~$ find <directory> -user <username>

To find files and directories owned by group use the following command:

~$ find <directory> -group <group name>

To find files that have a specific permission we will use the: -perm option

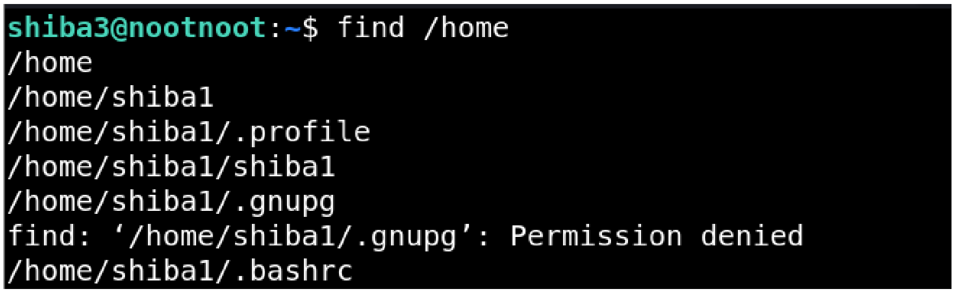


```
  -perm mode
         File's  permission  bits  are exactly mode (octal or symbolic).
```

Figure -31-

To find all the files in /home we will use the following command:

~$ find /home



```
shiba3@nootnoot:~$ find /home
/home
/home/shiba1
/home/shiba1/.profile
/home/shiba1/shiba1
/home/shiba1/.gnupg
find: ‘/home/shiba1/.gnupg’: Permission denied
/home/shiba1/.bashrc
```

Figure -32-

To find all the files owned by paradox on the whole system we will use the following command:

~$ find / -user paradox

Grep:

Grep is used to filter the output for lines that have a specific keyword, it can be used with a file to get lines that have the keyword.

Grep is a great command to search for a specific expression in a large amount of data files, like for example logs files.

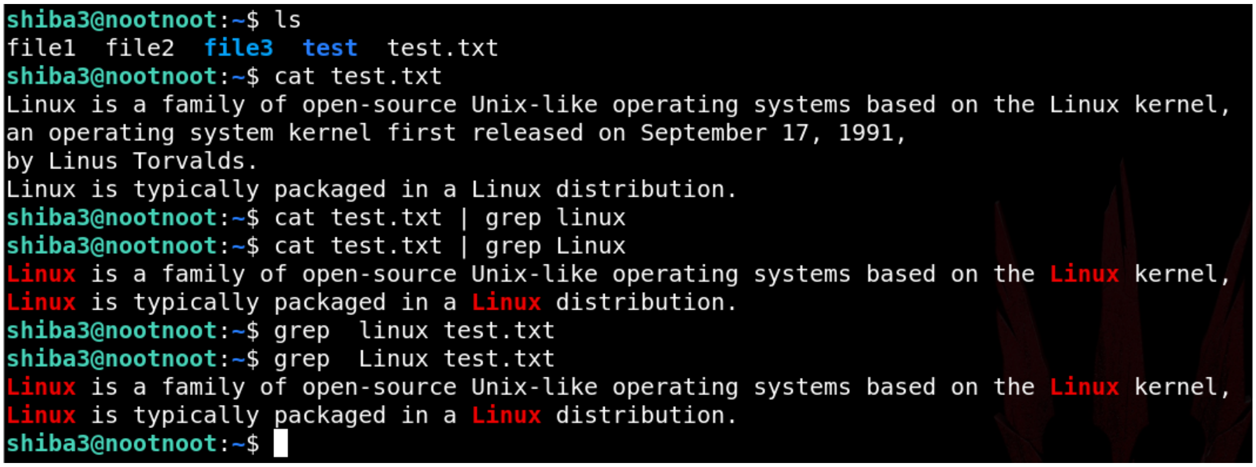
~$ grep <string> <filename>

For example:

~$ grep hello test

Note: grep is case sensitive

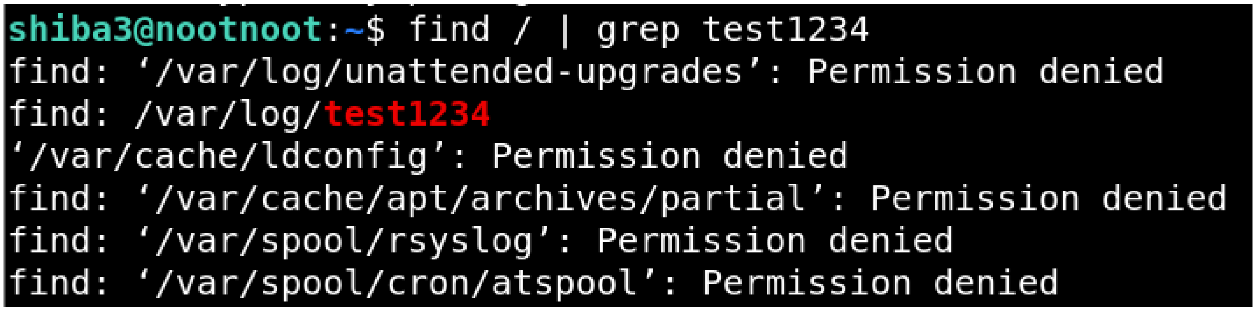
Note: grep can be used with regular expression



```
shiba3@nootnoot:~$ ls
file1  file2  file3  test  test.txt
shiba3@nootnoot:~$ cat test.txt
Linux is a family of open-source Unix-like operating systems based on the Linux kernel,
an operating system kernel first released on September 17, 1991,
by Linus Torvalds.
Linux is typically packaged in a Linux distribution.
shiba3@nootnoot:~$ cat test.txt | grep linux
shiba3@nootnoot:~$ cat test.txt | grep Linux
Linux is a family of open-source Unix-like operating systems based on the Linux kernel,
Linux is typically packaged in a Linux distribution.
shiba3@nootnoot:~$ grep  linux test.txt
shiba3@nootnoot:~$ grep  Linux test.txt
Linux is a family of open-source Unix-like operating systems based on the Linux kernel,
Linux is typically packaged in a Linux distribution.
shiba3@nootnoot:~$ 
```

Figure -33-

~$ find / | grep test1234

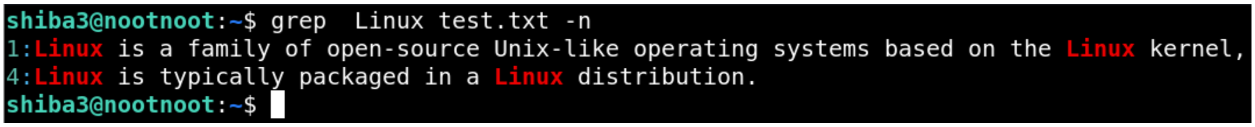


```
shiba3@nootnoot:~$ find / | grep test1234
find: '/var/log/unattended-upgrades': Permission denied
find: /var/log/test1234
'/var/cache/ldconfig': Permission denied
find: '/var/cache/apt/archives/partial': Permission denied
find: '/var/spool/rsyslog': Permission denied
find: '/var/spool/cron/atspool': Permission denied
```

Figure -34-

As with others commands, grep comes with a lot of options that will help you in your life.

For example:
To get the number of the line of matched line, use the: -n option

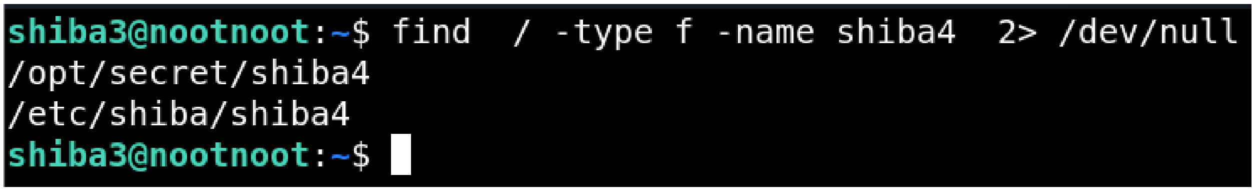


```
shiba3@nootnoot:~$ grep  Linux test.txt -n
1:Linux is a family of open-source Unix-like operating systems based on the Linux kernel,
4:Linux is typically packaged in a Linux distribution.
shiba3@nootnoot:~$ 
```

Figure -35-

To search for the string boop in the file aaaa in the directory /tmp:
~$ grep boop /tmp/aaaa

Binary shiba3:

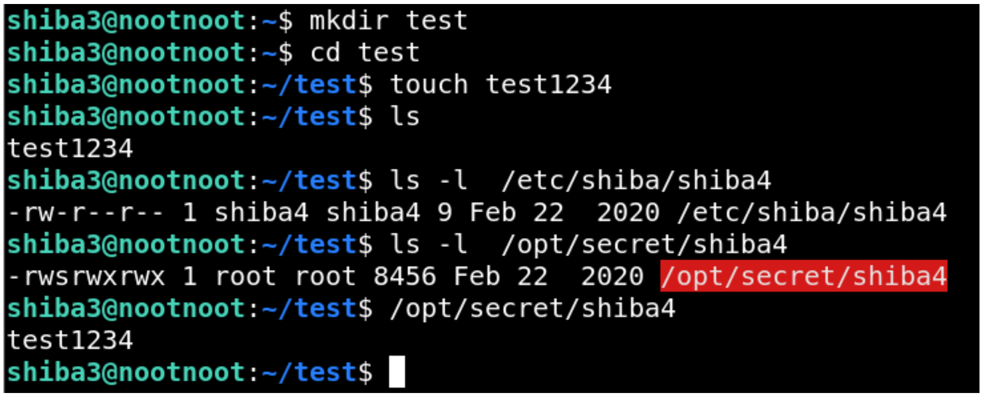


```
shiba3@nootnoot:~$ find  / -type f -name shiba4  2> /dev/null
/opt/secret/shiba4
/etc/shiba/shiba4
shiba3@nootnoot:~$ 
```

Figure -36-

Shiba4 as indicated will check if there is a test directory and inside that directory there is a file named test1234, and this directory is under the home directory of our user:

~$ mkdir test
~$ cd test
~$ touch test1234
~$ /opt/secret/shiba4



```
shiba3@nootnoot:~$ mkdir test
shiba3@nootnoot:~$ cd test
shiba3@nootnoot:~/test$ touch test1234
shiba3@nootnoot:~/test$ ls
test1234
shiba3@nootnoot:~/test$ ls -l  /etc/shiba/shiba4
-rw-r--r-- 1 shiba4 shiba4 9 Feb 22  2020 /etc/shiba/shiba4
shiba3@nootnoot:~/test$ ls -l  /opt/secret/shiba4
-rwsrwxrwx 1 root root 8456 Feb 22  2020 /opt/secret/shiba4
shiba3@nootnoot:~/test$ /opt/secret/shiba4
test1234
shiba3@nootnoot:~/test$ 
```

Figure -37-

Miscellaneous:

sudo:

Sometime you need to run a command as root, because your current privilege not sufficient to execute that command, so you can use sudo “the magic word”

```
~$ sudo <command>
```

The user to be able to use the sudo command need to be in the sudoer file

Note: to add a user to the sudoer file use:

```
~$ adduser <username> sudo
```

For example:
To specify which user you want to run a command as. We will use the “-u” option

To run whoami as jen user:

```
~$ sudo -u jen whoami
   jen
```

To list your current sudo privileges(what commands you can run, who you can run them as etc.) use: the “-l” option

```
~$ sudo -l
```

Adding users and groups:

To create a new user: adduser <username>
To add a new group: addgroup <group name>

Note: you need to run this commands as root

To add a user to a group

```
~$ usermod -a -G <groups separated by commas> <user>
```

For example:
 To add a user test to a group test we will execute:

```
~$ usermod -a -G test test
```

Nano:

nano is a text editor like notepad in windows

```
~$ nano <filename>
```

After writing your data, press ctrl+x to exit from nano (there is some straightforward step like press yes...)

Basic bash scripting:

You can run multiple commands in linux by saving commands you want to run in a file with .sh extension and run the file with bash

Test.sh

```
-----------------------------------------------------
echo “Hello World”
whoami
echo “HI”
-----------------------------------------------------
```

```
~$ ls
Test.sh
~$ bash Test.sh
Hello World
kali
HI
```

You can drop the sh extension, and tell the operating system which interpreter to use, this done by the shebang (#!) at the beginning : #!/bin/bash

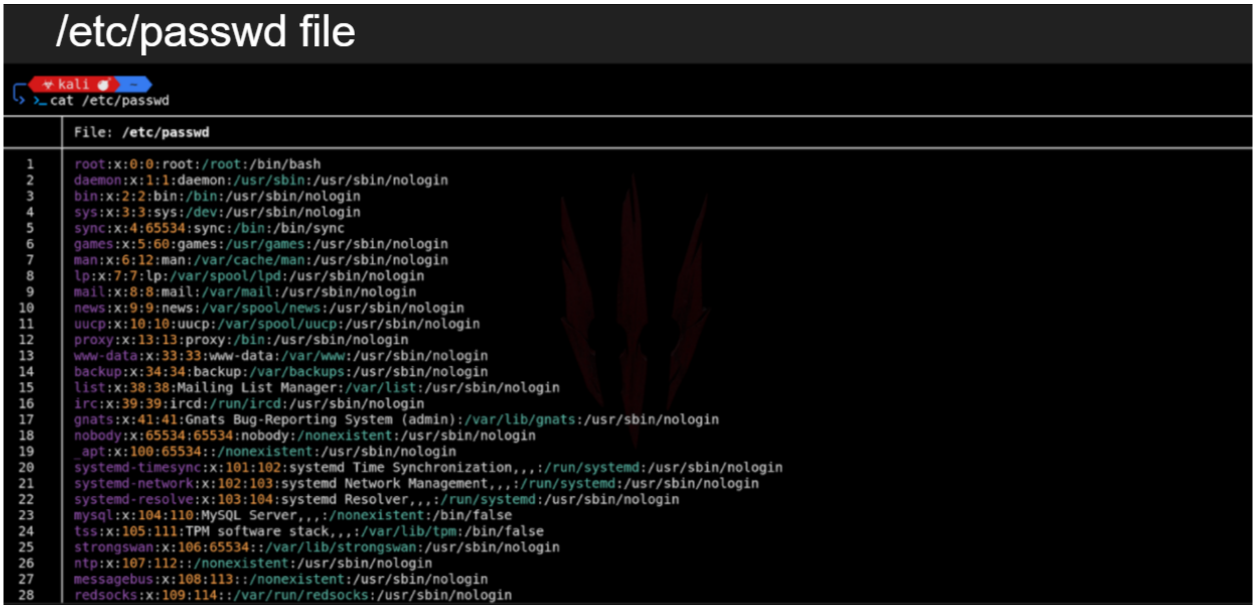
Test

```
 -----------------------------------------------------
#!/bin/bash
echo “Hello World”
whoami
echo “HI”
-----------------------------------------------------
```

```
~$ chmod 777 Test
~$ ./Test
Hello World
kali
HI
```

Important files and directories:

/etc/passwd: contains all users on the systems
It's readable by anyone.

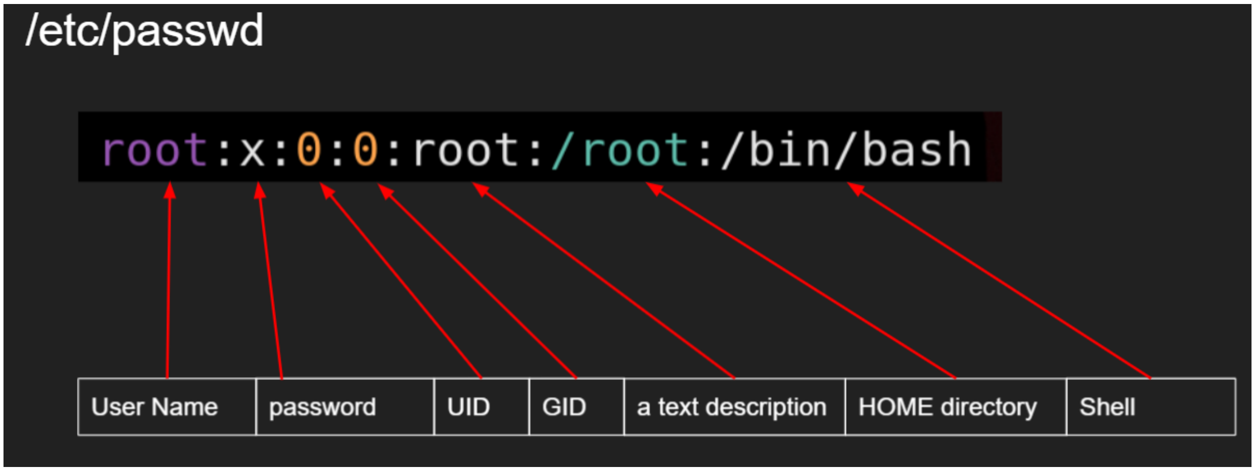


/etc/passwd file

```
kali ~
>_ cat /etc/passwd
```

```
File: /etc/passwd
```

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:101:102:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
systemd-network:x:102:103:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:103:104:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
mysql:x:104:110:MySQL Server,,,:/nonexistent:/bin/false
tss:x:105:111:TPM software stack,,,:/var/lib/tpm:/bin/false
strongswan:x:106:65534::/var/lib/strongswan:/usr/sbin/nologin
ntp:x:107:112::/nonexistent:/usr/sbin/nologin
messagebus:x:108:113::/nonexistent:/usr/sbin/nologin
redsocks:x:109:114::/var/run/redsocks:/usr/sbin/nologin
```

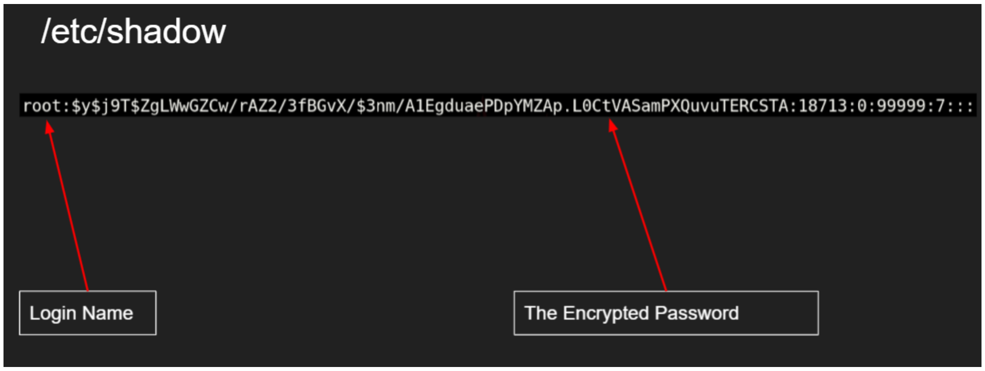


/etc/passwd

```
root:x:0:0:root:/root:/bin/bash
```

Figure -38-

/etc/shadow: contains all user hashed passwords

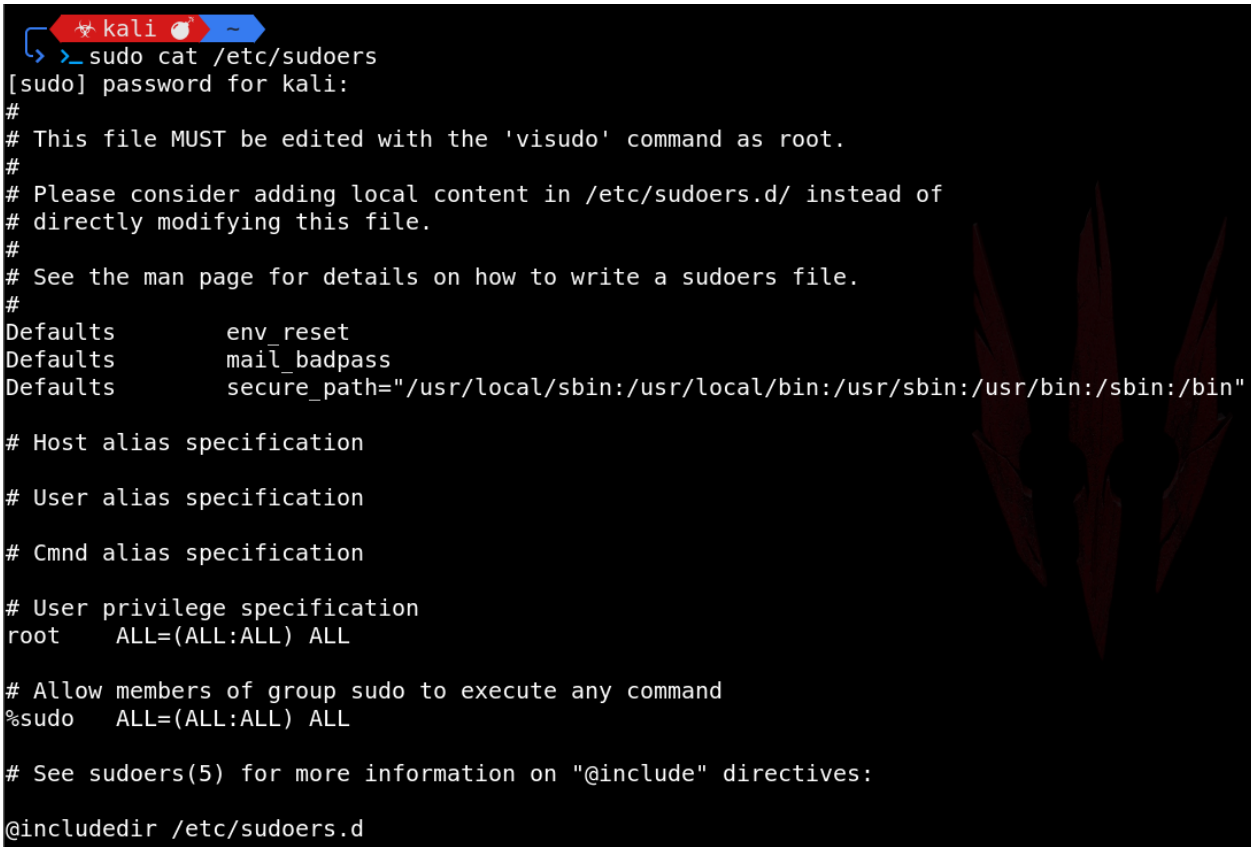


/etc/shadow

```
root:$y$j9T$ZgLWwGZCw/rAZ2/3fBGvX/$3nm/A1EgduaePDpYMZAp.L0CtVASamPXQuvuTERCSTA:18713:0:99999:7:::
```

Figure -39-

/tmp: every file created under this directory will get deleted after shutdown
/etc/sudoers: Used to control the sudo permissions of every user on the system



```
  kali   ~
 >_sudo cat /etc/sudoers
[sudo] password for kali:
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "@include" directives:

@includedir /etc/sudoers.d
```

Figure -40-

/home: it's your user home directory
/root: root user directory
/usr: where all the softwares installed
/bin and /sbin: contains binaries

Environment variable **PATH** is a specific variable used to store the path that tells the system where to look for commands that can be executed anywhere in the system.

To display **PATH** variable use:

~$ echo **$PATH**
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
~$

Installing packages apt:

apt used to download packages, packages may contain source code and different binaries and libraries.

~$ apt install <package name>

For example if you want to install nmap tool:

~$ sudo apt install nmap

We use sudo because only root user can downloads packages

To update and upgrade packages:

~$ apt-get update && apt-get upgrade

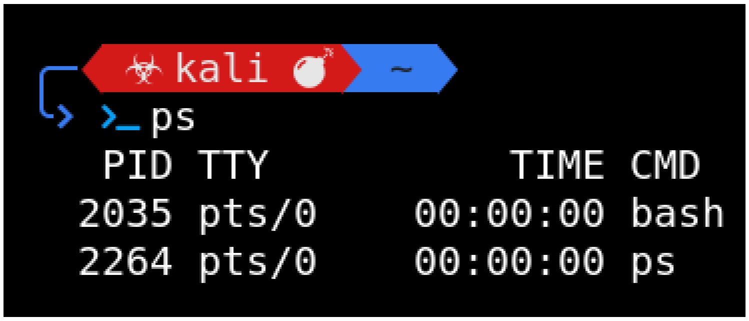
Processes:

Maybe you miss the hero task manager from windows world ;)

As we discuss before a process is in reality a binary or executable that run in the memory

To view user created process use:

~$ ps

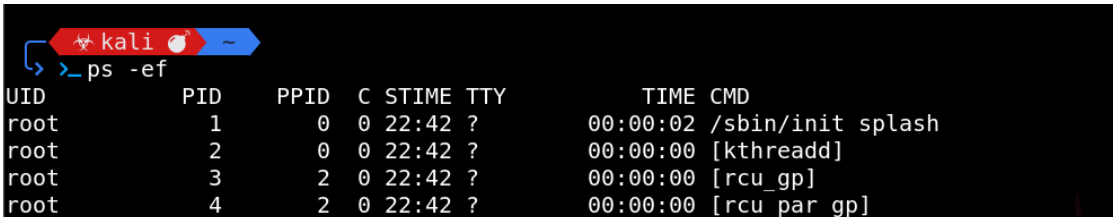


```
kali  ~
ps
  PID TTY          TIME CMD
 2035 pts/0    00:00:00 bash
 2264 pts/0    00:00:00 ps
```

Figure -41-

To view a list of all system processes use:

~$ ps -ef



```
kali  ~
ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 22:42 ?        00:00:02 /sbin/init splash
root           2       0  0 22:42 ?        00:00:00 [kthreadd]
root           3       2  0 22:42 ?        00:00:00 [rcu_gp]
root           4       2  0 22:42 ?        00:00:00 [rcu_par_gp]
```

Figure -42-

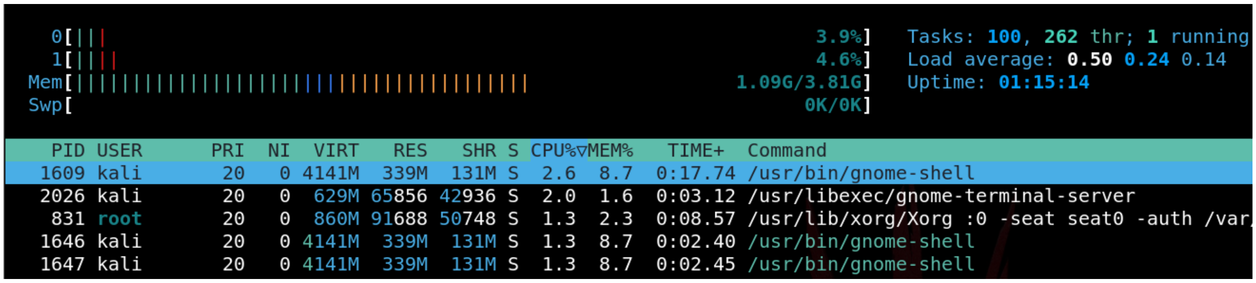
UID is the user ID
PID is the process ID, which is assigned by the operating system, it means a lot if you are going to learn OS not here :)

If you want to end a task in linux like what you do in windows you need “Kill”

~$ kill <PID>

Another way to get the list of processes:

~$ top
Note: you can download softwares based on top command but with a pretty and colorful output like “htop”
~$ sudo apt install htop
~# apt install htop



```
    0[|||                                                           3.9%]   Tasks: 100, 262 thr; 1 running
    1[||||                                                          4.6%]   Load average: 0.50 0.24 0.14
  Mem[||||||||||||||||||||||||||||||||||||||||               1.09G/3.81G]   Uptime: 01:15:14
  Swp[                                                             0K/0K]

    PID USER       PRI  NI  VIRT   RES   SHR S CPU%▽MEM%   TIME+  Command
   1609 kali        20   0 4141M  339M  131M S  2.6  8.7  0:17.74 /usr/bin/gnome-shell
   2026 kali        20   0  629M 65856 42936 S  2.0  1.6  0:03.12 /usr/libexec/gnome-terminal-server
    831 root        20   0  860M 91688 50748 S  1.3  2.3  0:08.57 /usr/lib/xorg/Xorg :0 -seat seat0 -auth /var
   1646 kali        20   0 4141M  339M  131M S  1.3  8.7  0:02.40 /usr/bin/gnome-shell
   1647 kali        20   0 4141M  339M  131M S  1.3  8.7  0:02.45 /usr/bin/gnome-shell
```

Figure -43-

P.S: Check my Linux Privilege Escalation video for more explanation about permissions youtube: https://youtu.be/fvr3q0xanA0

Thanks you for reading <3