Hacktales

HackTales' Linux Crash Course Handbook

Instructor: Daniel Johnson (eJPT,CPTS,PNPT,OSCP)

Introduction

Why linux?

Hackers, both ethical and malicious, favor Linux for its open-source flexibility, powerful command-line capabilities, and vast selection of built-in security tools. Unlike proprietary operating systems, Linux allows complete customization, better privacy controls, and enhanced networking features essential for penetration testing and cybersecurity research. Distributions like Kali Linux and Parrot OS come preloaded with hacking tools, while live-boot options enable anonymous, trace-free operations. Its lightweight nature, strong security model, and active community make Linux the ultimate choice for those seeking control, efficiency, and advanced security exploitation.

Downloading Kali Linux

This guide was created with the intent of educating Red Team interns with the knowledge to run Linux in virtual environments, other methods such as Dual Booting, booting via a bootable device, etc will not be demonstrated within this document but you are encouraged to carefully explore these options at your discretion.

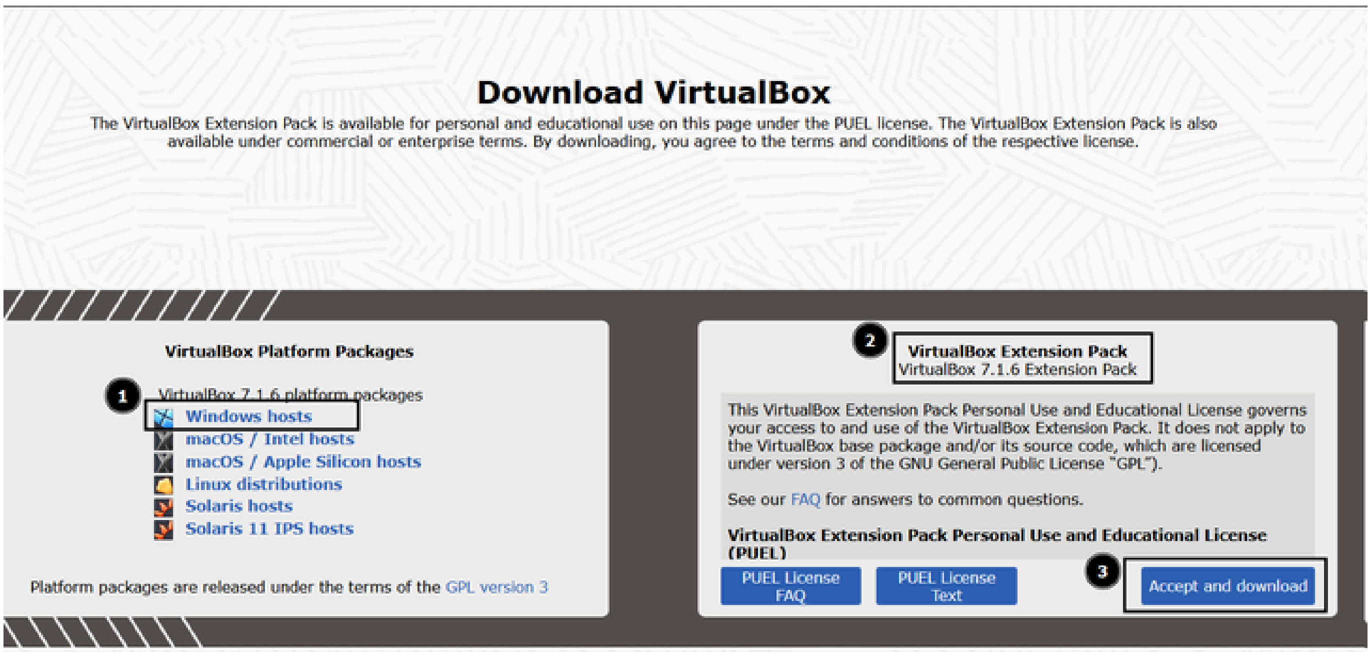
What is a Hypervisor?

A **hypervisor** (also known as a Virtual Machine Monitor, or VMM) is software or firmware that allows multiple virtual machines (VMs) to run on a single physical machine. It creates and manages virtualized environments by allocating resources like CPU, memory, and storage to each VM while keeping them isolated from one another.

They are of 2 types, Type-1 which runs directly on the hardware e.g KVM, Microsoft Hyper-V, etc. and Type-2 which runs as an application on the host operating system e.g VirtualBox, VMWare etc.

Running Kali via VirtualBox

To run Kali via virtualBox, navigate to the official website to download the Virtualbox setup file first at https://www.virtualbox.org/wiki/Downloads. The setup for windows can be downloaded by clicking the button shown in (1) below, the Extension pack should also be downloaded by clicking (3) Accept and download for special functionalities like full screen zoom in while using the Virtual machine.



Download VirtualBox

The VirtualBox Extension Pack is available for personal and educational use on this page under the PUEL license. The VirtualBox Extension Pack is also available under commercial or enterprise terms. By downloading, you agree to the terms and conditions of the respective license.

VirtualBox Platform Packages

1 VirtualBox 7.1.6 platform packages
- Windows hosts
- macOS / Intel hosts
- macOS / Apple Silicon hosts
- Linux distributions
- Solaris hosts
- Solaris 11 IPS hosts

Platform packages are released under the terms of the GPL version 3

2 VirtualBox Extension Pack
VirtualBox 7.1.6 Extension Pack

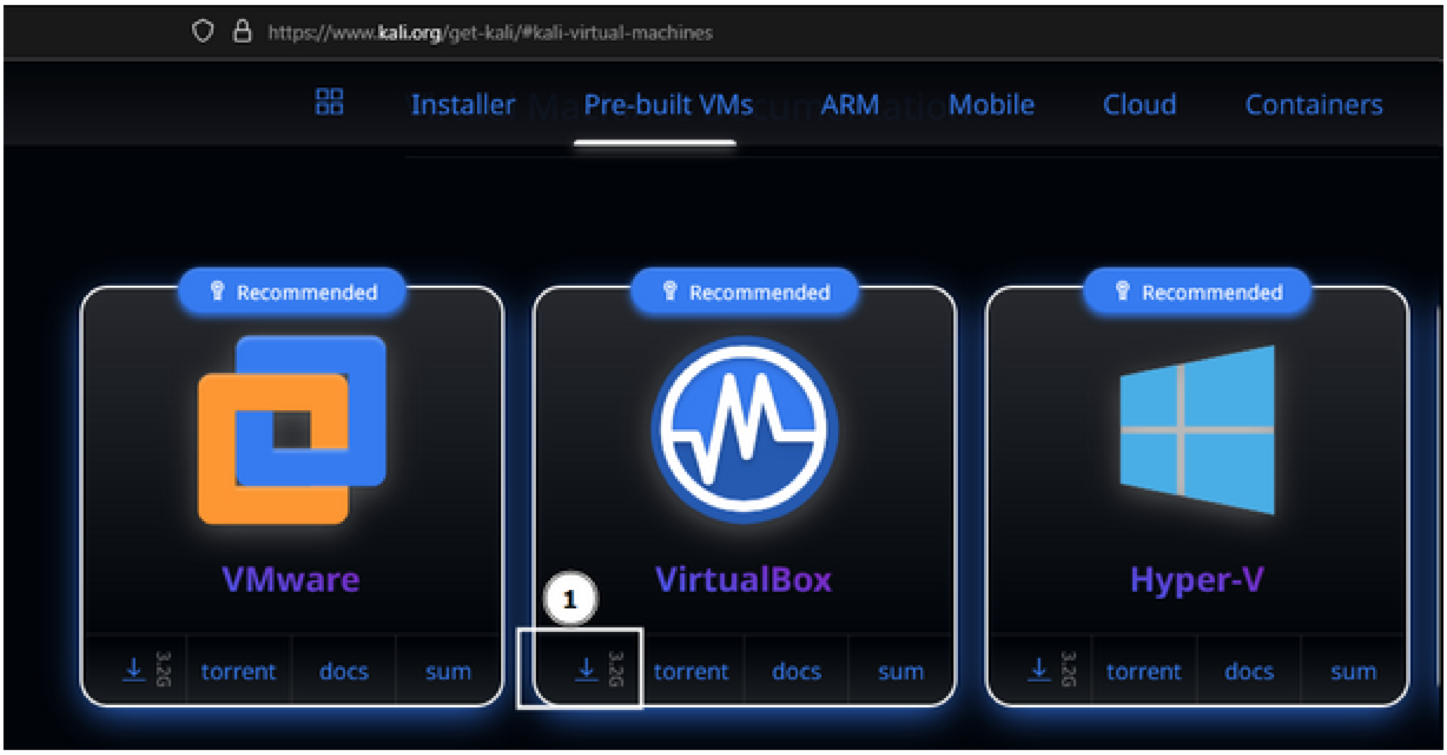
This VirtualBox Extension Pack Personal Use and Educational License governs your access to and use of the VirtualBox Extension Pack. It does not apply to the VirtualBox base package and/or its source code, which are licensed under version 3 of the GNU General Public License "GPL").

See our FAQ for answers to common questions.

VirtualBox Extension Pack Personal Use and Educational License (PUEL)

Figure 1. Downloading Virtualbox

Next download Kali by navigating to **https://www.kali.org/get-kali/#kali-virtual-machines** to download a pre built machine (pre-built in the sense that it's already been pre-configured and we do not have to go through the installation process of setting the language, partition space, filesystem type, user account creation etc) as a result of no user account creation you have a default kali user whose password is kali for this virtual machine, click on the download icon shown in Figure 2 below

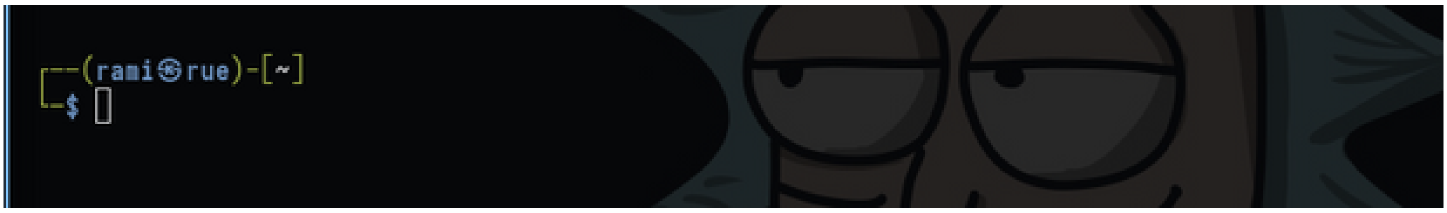

https://www.kali.org/get-kali/#kali-virtual-machines
Installer
Pre-built VMs
ARM
Mobile
Cloud
Containers
Recommended
VMware
3.2G
torrent
docs
sum
Recommended
VirtualBox
1
3.2G
torrent
docs
sum
Recommended
Hyper-V
3.2G
torrent
docs
sum

Now we will set up Kali in virtualbox using the preconfigured virtual machine previously downloaded from the kali website

Linux Filesystem & Navigating Linux

The Linux Shell

A Linux terminal, also called a shell or command line, provides a text-based input/output (I/O) interface between users and the kernel for a computer system. We can think of a shell as a text-based GUI in which we enter commands to perform actions like navigating to other directories, working with files, and obtaining information from the system but with way more capabilities. A shell prompt has the format username@hostname



```
┌──(rami㉿rue)-[~]
└─$ 
```

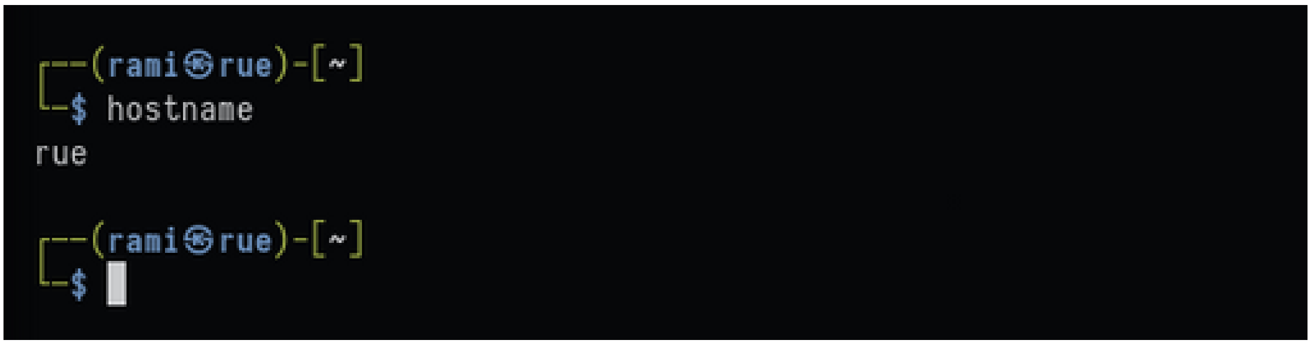
The most commonly used shell in Linux is the Bourne-Again Shell (BASH), and is part of the GNU project. Everything we do through the GUI we can do with the shell. The shell gives us many more possibilities to interact with programs and processes to get information faster. Besides, many processes can be easily automated with smaller or larger scripts that make manual work much easier.

Besides Bash, there also exist other shells like Tcsh/Csh, Ksh, Zsh, Fish shell and others.

SYSTEM INFORMATION

hostname

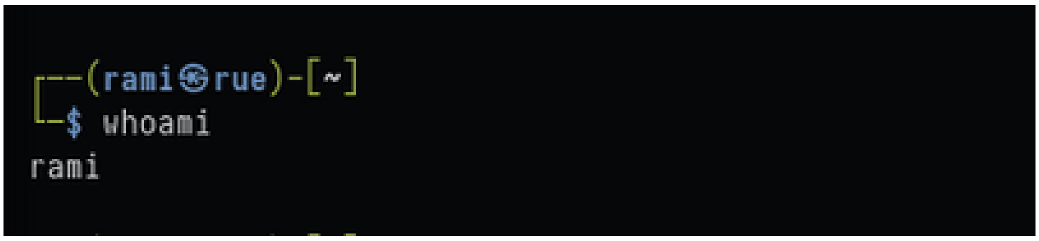
The hostname command is pretty self-explanatory and will just print the name of the computer that we are logged into


```
┌──(rami㉿rue)-[~]
└─$ hostname
rue

┌──(rami㉿rue)-[~]
└─$ █
```

Whoami

This quick and easy command can be used on both Windows and Linux systems to get our current username. During a security assessment, we obtain reverse shell access on a host, and one of the first bits of situational awareness we should do is figuring out what user we are running as.

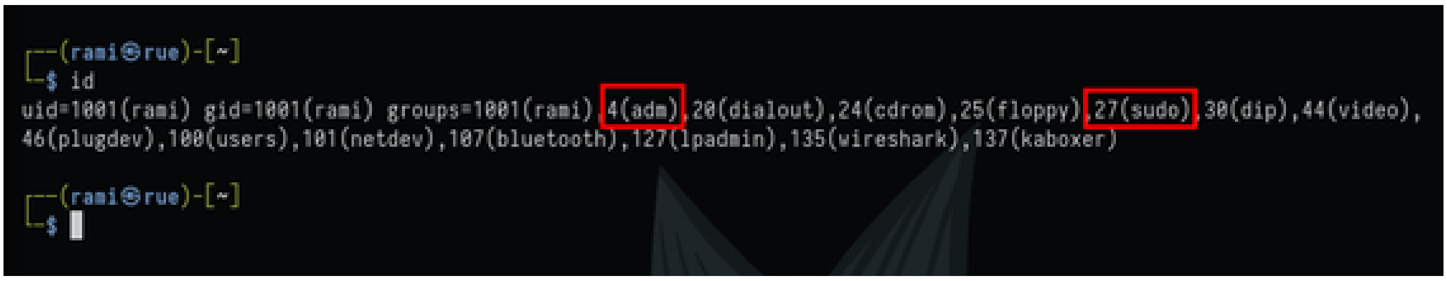
From there, we can figure out if the user has any special privileges/access.


```
┌──(rami㉿rue)-[~]
└─$ whoami
rami
```

id

The id command expands on the whoami command and prints out our effective group membership and IDs. This can be of interest to penetration testers looking to see what access a user may have and sysadmins looking to audit account permissions and group membership. the adm group means that the user can read log files in /var/log and could potentially gain access to sensitive information, membership in the sudo group is of particular interest as this means our user can run some or all commands as the all-powerful root user.

Sudo rights could help us escalate privileges or could be a sign to a sysadmin that they may need to audit permissions and group memberships to remove any access that is not required for a given user to carry out their day-to-day tasks.

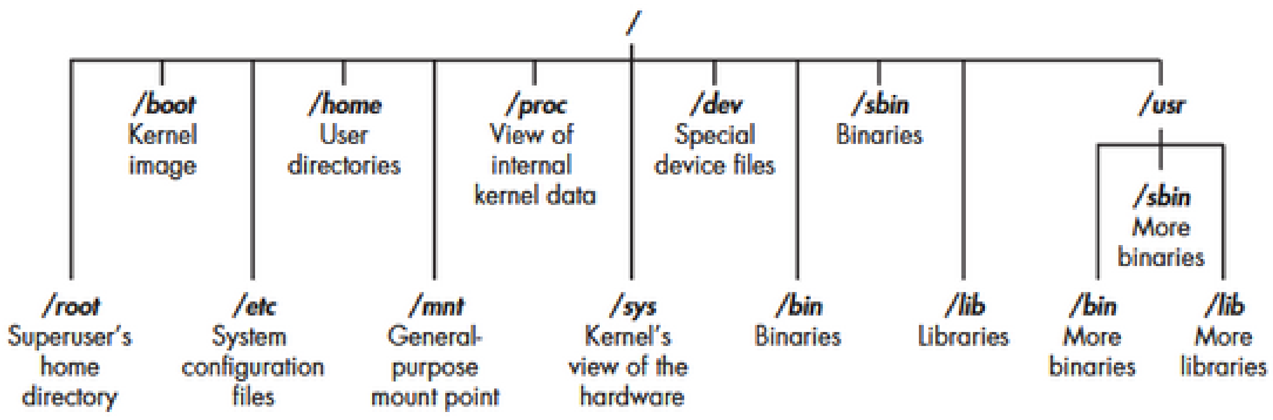

```
┌──(rami㉿rue)-[~]
└─$ id
uid=1001(rami) gid=1001(rami) groups=1001(rami),4(adm),20(dialout),24(cdrom),25(floppy),27(sudo),30(dip),44(video),
46(plugdev),100(users),101(netdev),107(bluetooth),127(lpadmin),135(wireshark),137(kaboxer)

┌──(rami㉿rue)-[~]
└─$ 
```

Command	Description
whoami	Displays current username.
id	Returns users identity
hostname	Sets or prints the name of current host system.
uname	Prints basic information about the operating system name and system hardware.
pwd	Returns working directory name.
ifconfig	The ifconfig utility is used to assign or to view an address to a network interface and/or configure network interface parameters.
ip	Ip is a utility to show or manipulate routing, network devices, interfaces and tunnels.
netstat	Shows network status.
ss	Another utility to investigate sockets.
ps	Shows process status.
who	Displays who is logged in.
env	Prints environment or sets and executes command.
lsblk	Lists block devices.

lsusb	Lists USB devices
lsof	Lists opened files.
lspci	Lists PCI devices.

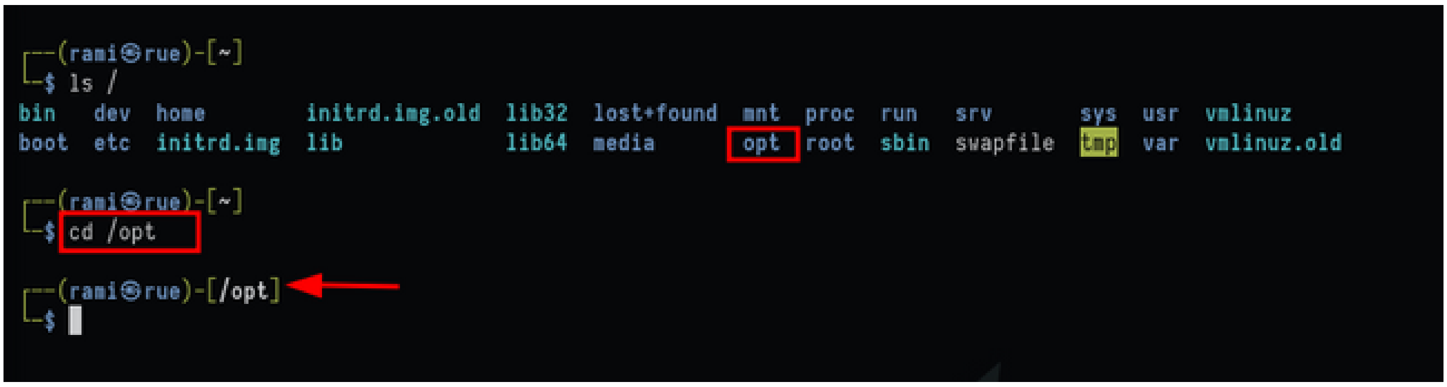
In linux, everything is a file, hence the existence of the Linux FileSystem Hierarchy where each file takes its source from a root directory which can be navigated to via a file explorer

---



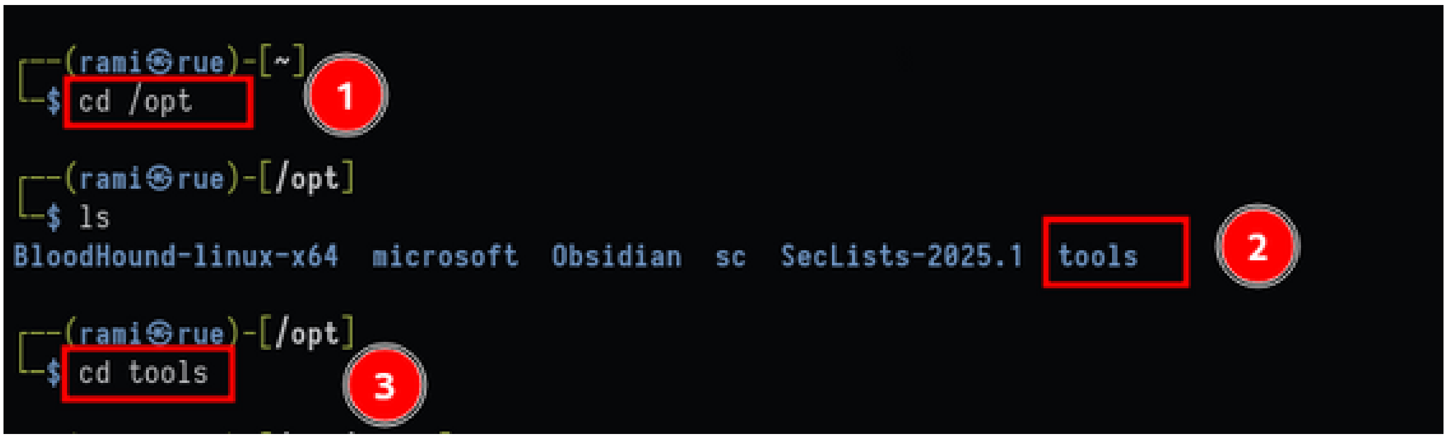
Linux Filesystem Hierarchy

At the very top of the file-system structure is /, which is often referred to as the root of the filesystem, the symbol "~" known as a tilde represents the user home folder in our case is "/home/rami" to change move into a folder, you would use the "cd" command to change directory into that folder specifying either a "relative path" or an "absolute path"



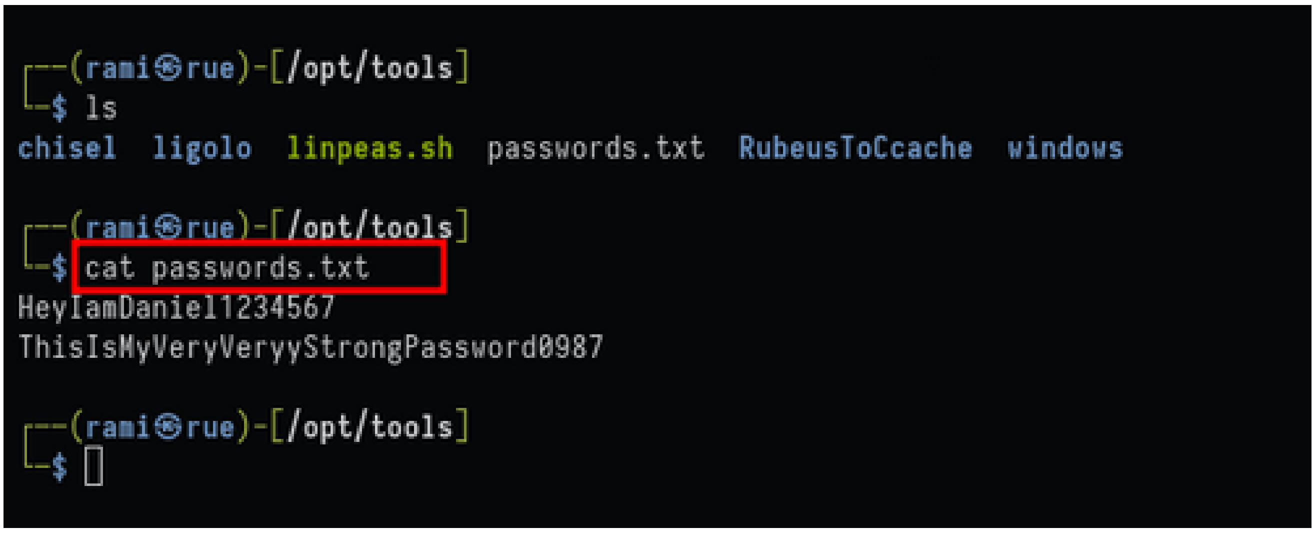
```
┌──(rami㉿rue)-[~]
└─$ ls /
bin   dev  home        initrd.img.old  lib32  lost+found  mnt  proc  run   srv       sys  usr  vmlinuz
boot  etc  initrd.img  lib             lib64  media       opt  root  sbin  swapfile  tmp  var  vmlinuz.old

┌──(rami㉿rue)-[~]
└─$ cd /opt

┌──(rami㉿rue)-[/opt]
└─$ ▌
```

Within the /opt folder, I have a couple files including a "tools" folder, as shown below in (2), to move into the "tools" folder I will once again use the cd command, doing this in this manner is utilizing the Relative path of the folder. It's absolute path would be /tools/opt and to navigate into the folder it would be "cd /opt/tools" rather than "cd tools" while inside of the /opt/ folder.



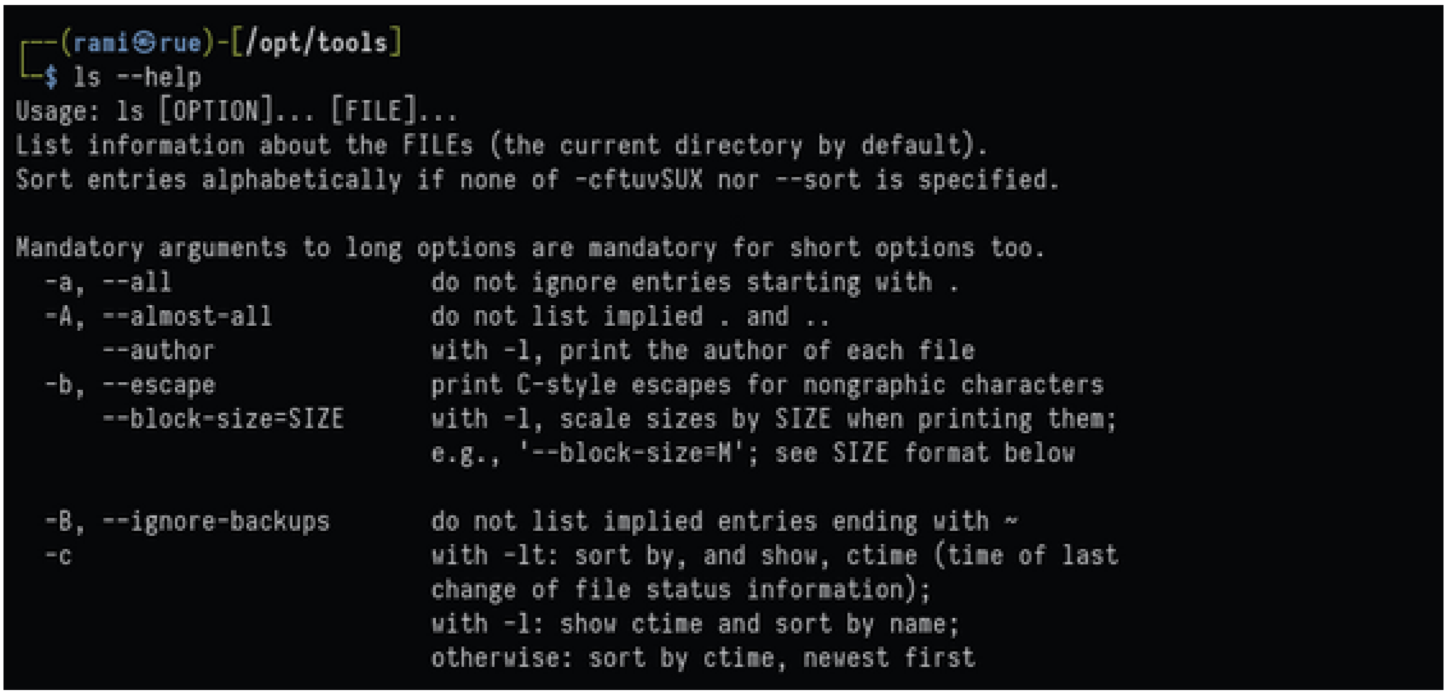
```
┌──(rami㉿rue)-[~]
└─$ cd /opt

┌──(rami㉿rue)-[/opt]
└─$ ls
BloodHound-linux-x64  microsoft  Obsidian  sc  SecLists-2025.1  tools

┌──(rami㉿rue)-[/opt]
└─$ cd tools
```

To list the content of the directory, you can utilize the "ls" command, and to read the content of a textual file you can utilize the "cat" command



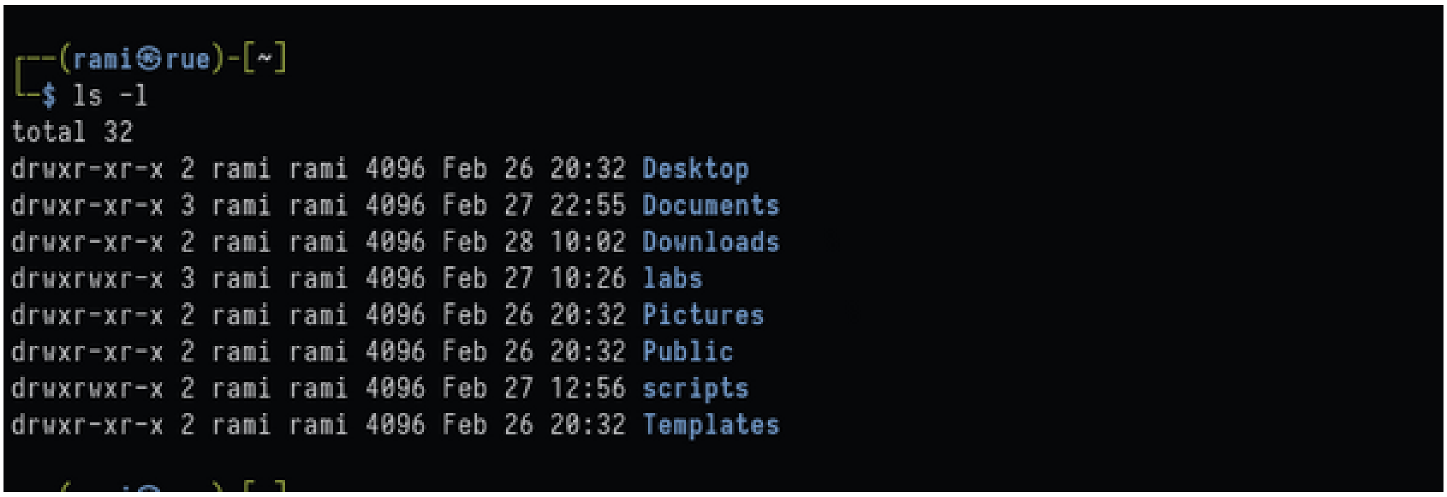
```
┌──(rami㉿rue)-[/opt/tools]
└─$ ls
chisel  ligolo  linpeas.sh  passwords.txt  RubeusToCcache  windows

┌──(rami㉿rue)-[/opt/tools]
└─$ cat passwords.txt
HeyIamDaniel1234567
ThisIsMyVeryyStrongPassword0987

┌──(rami㉿rue)-[/opt/tools]
└─$ ▯
```

In linux, each command has additional “options” and such options can be listed with either the “--help” command or the “man” manual command



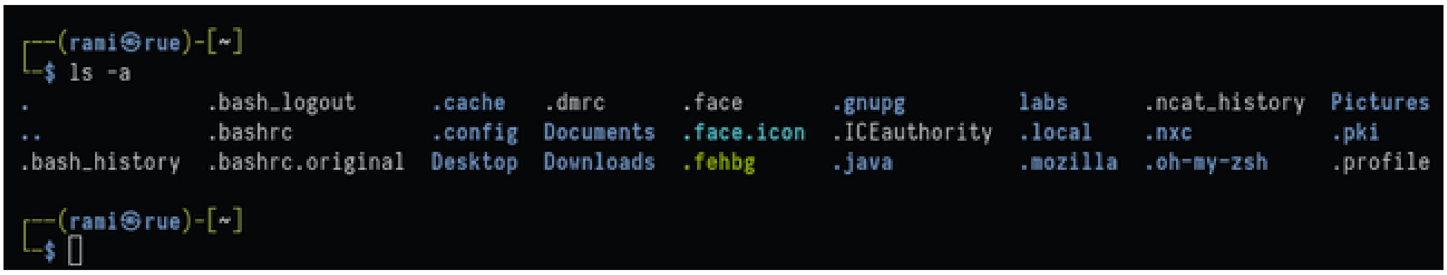
```
┌──(rami㉿rue)-[/opt/tools]
└─$ ls --help
Usage: ls [OPTION]... [FILE]...
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.
  -a, --all                  do not ignore entries starting with .
  -A, --almost-all           do not list implied . and ..
      --author               with -l, print the author of each file
  -b, --escape               print C-style escapes for nongraphic characters
      --block-size=SIZE      with -l, scale sizes by SIZE when printing them;
                             e.g., '--block-size=M'; see SIZE format below

  -B, --ignore-backups       do not list implied entries ending with ~
  -c                         with -lt: sort by, and show, ctime (time of last
                             change of file status information);
                             with -l: show ctime and sort by name;
                             otherwise: sort by ctime, newest first
```

The options have to be appended to the command to give the added functionality, for example if we want to view a list of all files in a directory in a list order we will use the “ls -l” option



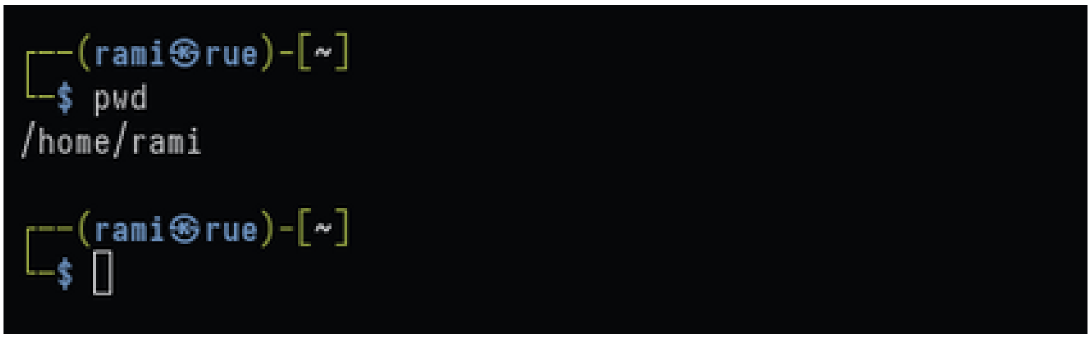
```
┌──(rami㉿rue)-[~]
└─$ ls -l
total 32
drwxr-xr-x 2 rami rami 4096 Feb 26 20:32 Desktop
drwxr-xr-x 3 rami rami 4096 Feb 27 22:55 Documents
drwxr-xr-x 2 rami rami 4096 Feb 28 10:02 Downloads
drwxrwxr-x 3 rami rami 4096 Feb 27 10:26 labs
drwxr-xr-x 2 rami rami 4096 Feb 26 20:32 Pictures
drwxr-xr-x 2 rami rami 4096 Feb 26 20:32 Public
drwxrwxr-x 2 rami rami 4096 Feb 27 12:56 scripts
drwxr-xr-x 2 rami rami 4096 Feb 26 20:32 Templates
```

To view hidden files we will use the “ls -a” option, hidden files in linux begin their names with a period or fullstop, as shown below



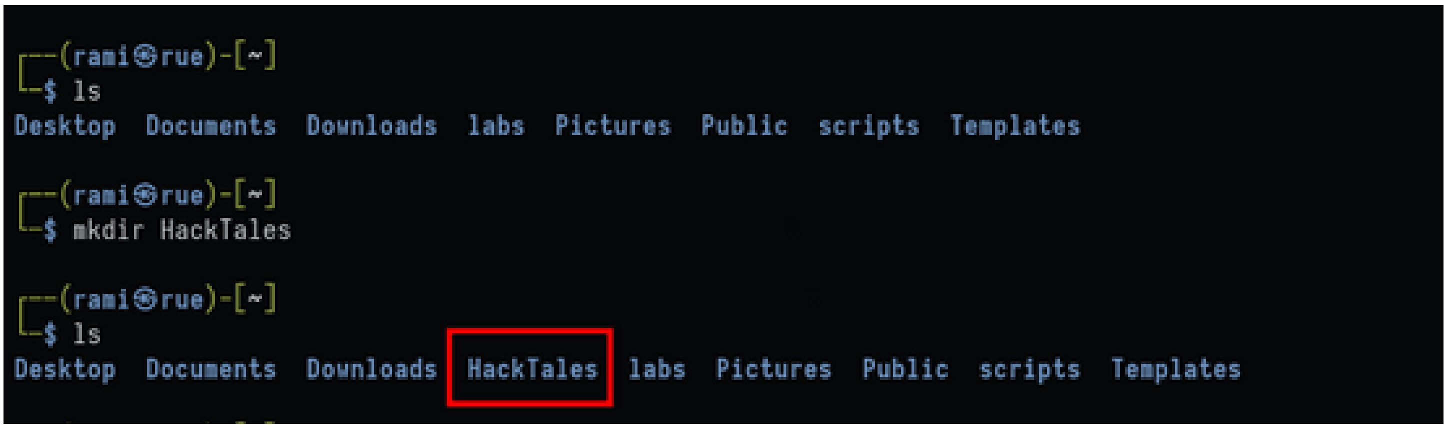
```
┌──(rami㉿rue)-[~]
└─$ ls -a
.              .bash_logout     .cache   .dmrc      .face       .gnupg         labs      .ncat_history  Pictures
..             .bashrc          .config  Documents  .face.icon  .ICEauthority  .local    .nxc           .pki
.bash_history  .bashrc.original Desktop  Downloads  .fehbg      .java          .mozilla  .oh-my-zsh     .profile

┌──(rami㉿rue)-[~]
└─$ ▯
```

Sometimes we might not know where in the filesystem we are, or we would want to be sure, we can use the “pwd” command which prints the path of the current working directory on the filesystem to screen



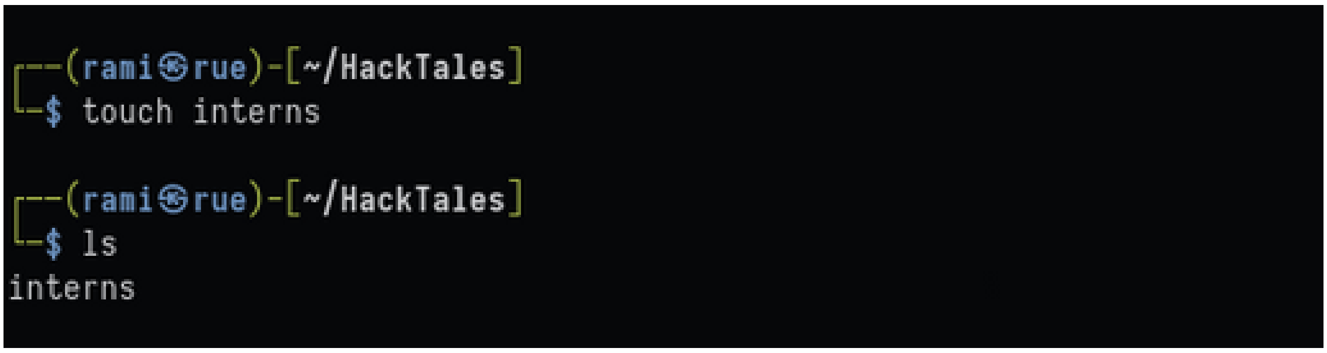
```
┌──(rami㉿rue)-[~]
└─$ pwd
/home/rami

┌──(rami㉿rue)-[~]
└─$ ▯
```

To create folders we can use the “mkdir” command



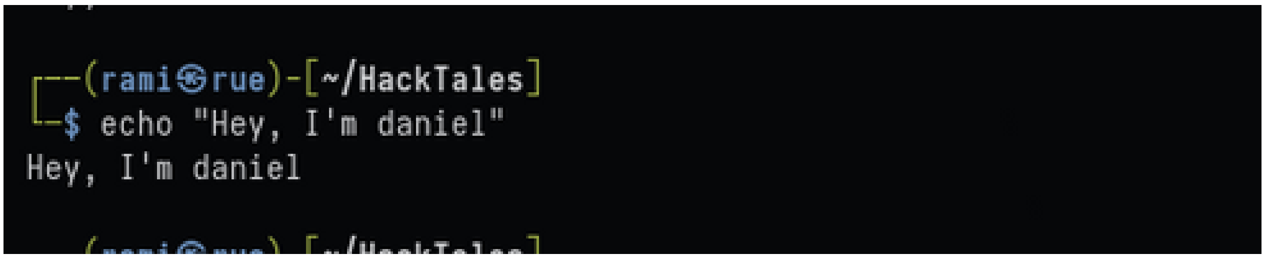
```
┌──(rami㉿rue)-[~]
└─$ ls
Desktop  Documents  Downloads  labs  Pictures  Public  scripts  Templates

┌──(rami㉿rue)-[~]
└─$ mkdir HackTales

┌──(rami㉿rue)-[~]
└─$ ls
Desktop  Documents  Downloads  HackTales  labs  Pictures  Public  scripts  Templates
```

We can then use the “cd” command to move into the directory to create files, to create files we could use the “touch” command which would create an empty text file



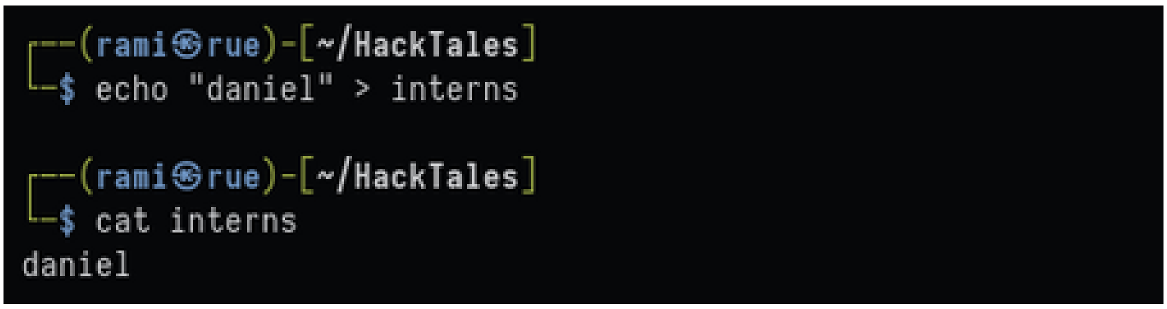
```
┌──(rami㉿rue)-[~/HackTales]
└─$ touch interns

┌──(rami㉿rue)-[~/HackTales]
└─$ ls
interns
```

To put content into our file, we could use redirectors, the “echo” command prints a text to screen just like the “print()” function in python. We can then use a redirector, represented by a greater than sign which passes the content on the left side to the file on the right side.



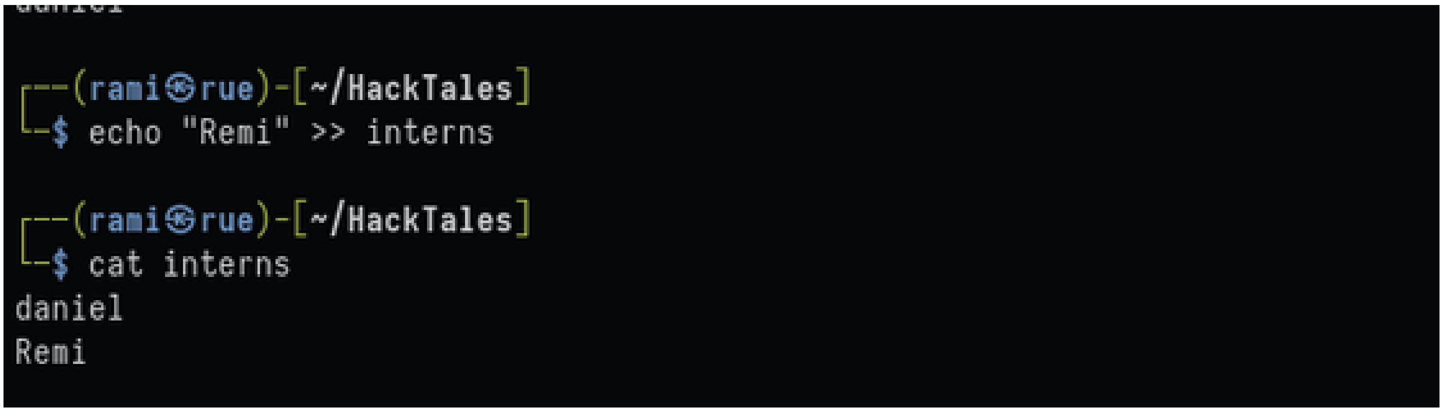
```
┌──(rami㉿rue)-[~/HackTales]
└─$ echo "Hey, I'm daniel"
Hey, I'm daniel
```

To put this content into our file we can use a redirector “>” for that



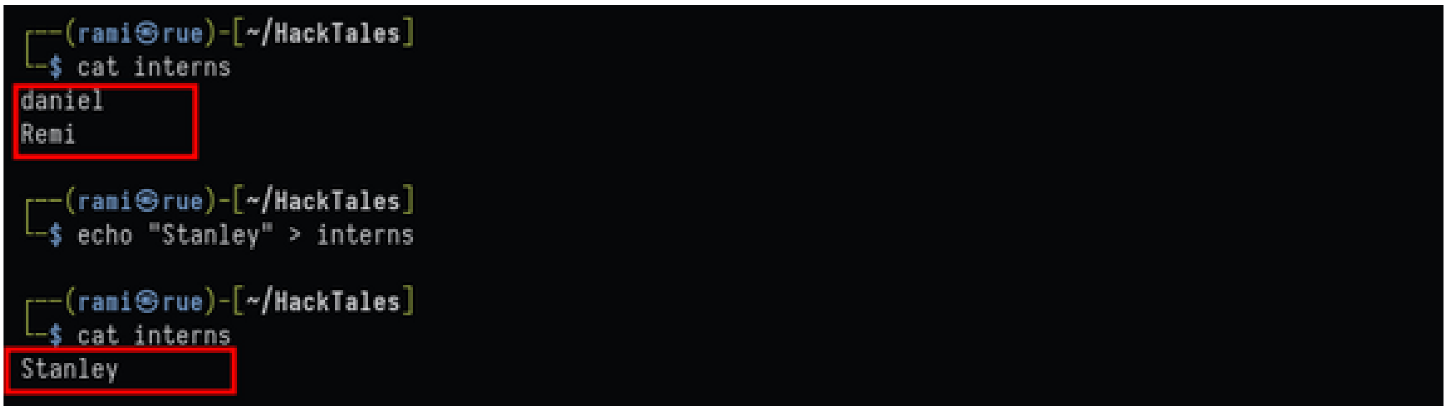
```
┌──(rami㉿rue)-[~/HackTales]
└─$ echo "daniel" > interns

┌──(rami㉿rue)-[~/HackTales]
└─$ cat interns
daniel
```

A single redirector replaces the entire content of a file but a double redirector “>>” appends the content of the file, remember this when performing actions on sensitive configuration files.



```
┌──(rami㉿rue)-[~/HackTales]
└─$ echo "Remi" >> interns

┌──(rami㉿rue)-[~/HackTales]
└─$ cat interns
daniel
Remi
```

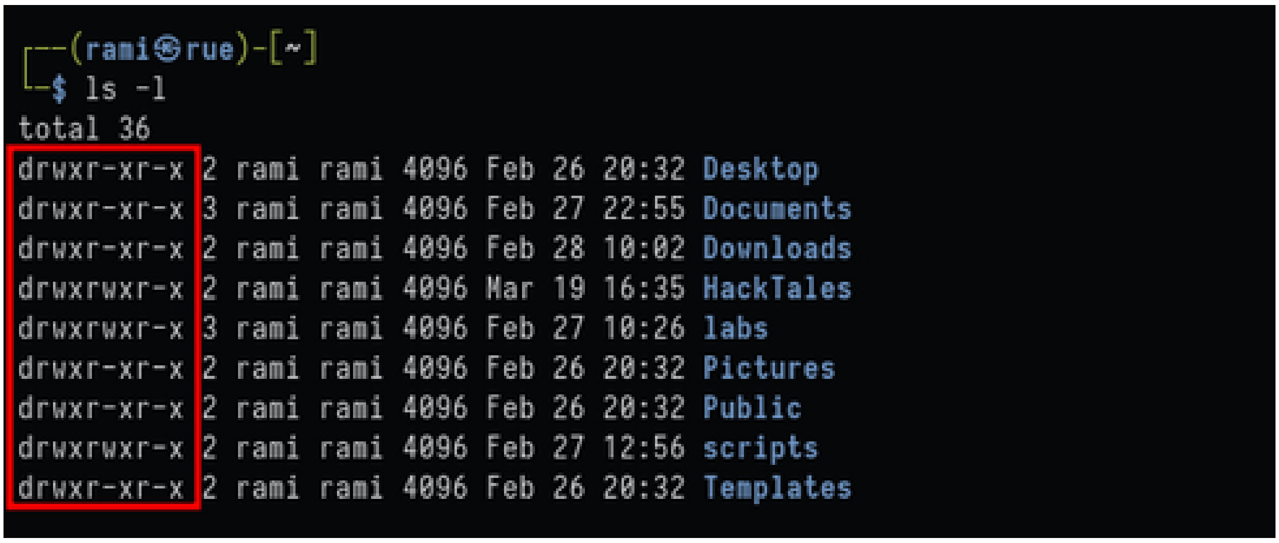
Notice how a single redirector overwrites the content of the file as shown below



```
┌──(rami㉿rue)-[~/HackTales]
└─$ cat interns
daniel
Remi

┌──(rami㉿rue)-[~/HackTales]
└─$ echo "Stanley" > interns

┌──(rami㉿rue)-[~/HackTales]
└─$ cat interns
Stanley
```

LINUX PERMISSIONS

Files & folders have different permissions or file modes. Let's look at an example:



```
┌──(rami㉿rue)-[~]
└─$ ls -l
total 36
drwxr-xr-x 2 rami rami 4096 Feb 26 20:32 Desktop
drwxr-xr-x 3 rami rami 4096 Feb 27 22:55 Documents
drwxr-xr-x 2 rami rami 4096 Feb 28 10:02 Downloads
drwxrwxr-x 2 rami rami 4096 Mar 19 16:35 HackTales
drwxrwxr-x 3 rami rami 4096 Feb 27 10:26 labs
drwxr-xr-x 2 rami rami 4096 Feb 26 20:32 Pictures
drwxr-xr-x 2 rami rami 4096 Feb 26 20:32 Public
drwxrwxr-x 2 rami rami 4096 Feb 27 12:56 scripts
drwxr-xr-x 2 rami rami 4096 Feb 26 20:32 Templates
```

There are four parts to a file's permissions. The first part is the filetype, which is denoted by the first character in the permissions, in our case since we are looking at a directory it shows d for the filetype. Most commonly you will see a - for a regular file.

The next three parts of the file mode are the actual permissions. The permissions are grouped into 3 bits each. The first 3 bits are user permissions, then group permissions and then other permissions. I've added the pipe to make it easier to differentiate**.**

d | rwx | r-x | r-x

Each character represent a different permission:

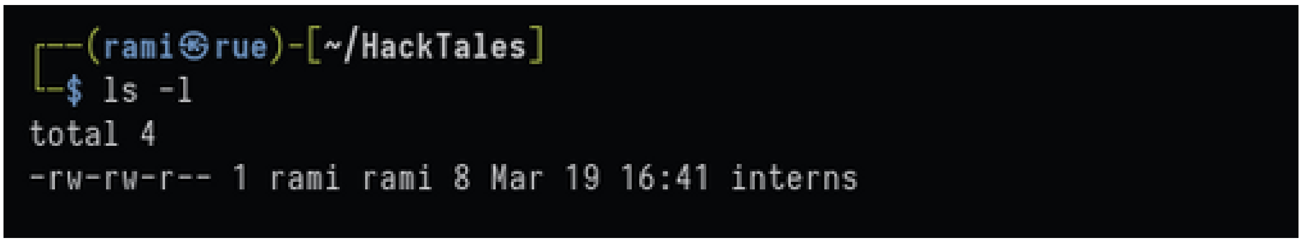
r: readable

w: writable

x: executable (basically an executable program)

-: empty

Hence in the image below, the user “rami” has read and write permissions on the file, the group “rami” as well has the read and write permissions, and every other user on this computer has only read permissions on our “interns” file, meaning they cannot add more interns i.e Write to the file.



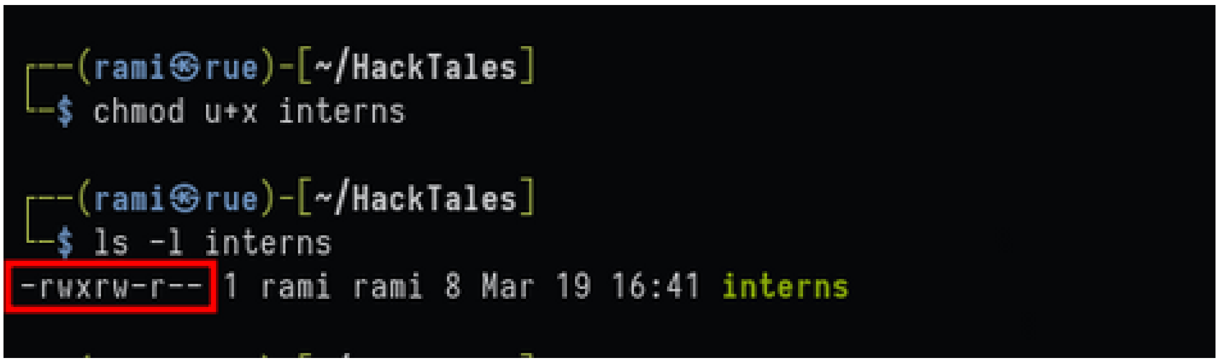
```
┌──(rami㉿rue)-[~/HackTales]
└─$ ls -l
total 4
-rw-rw-r-- 1 rami rami 8 Mar 19 16:41 interns
```

CHANGING PERMISSIONS

Changing permissions can easily be done with the chmod command.

First, pick which permission set you want to change, user, group or other. You can add or remove permissions with a + or -, let's look at some examples.

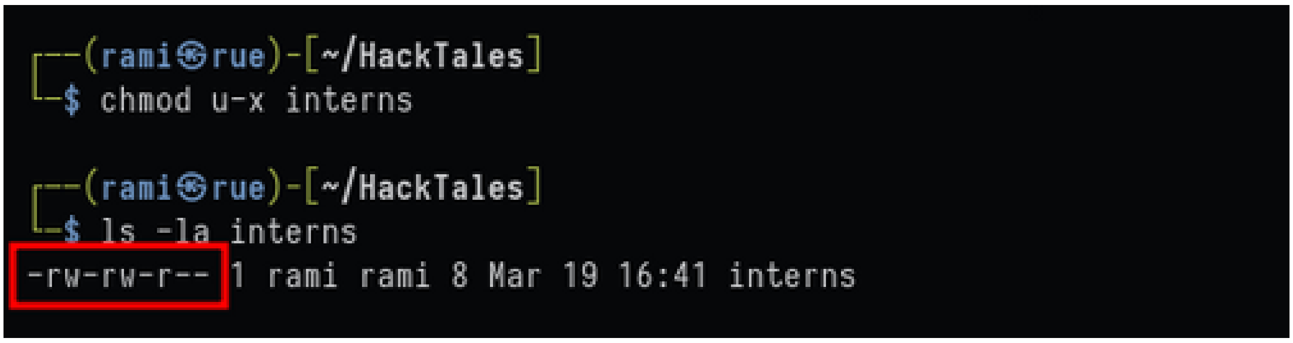
Adding permission bit on a file



```
┌──(rami㉿rue)-[~/HackTales]
└─$ chmod u+x interns

┌──(rami㉿rue)-[~/HackTales]
└─$ ls -l interns
-rwxrw-r-- 1 rami rami 8 Mar 19 16:41 interns
```

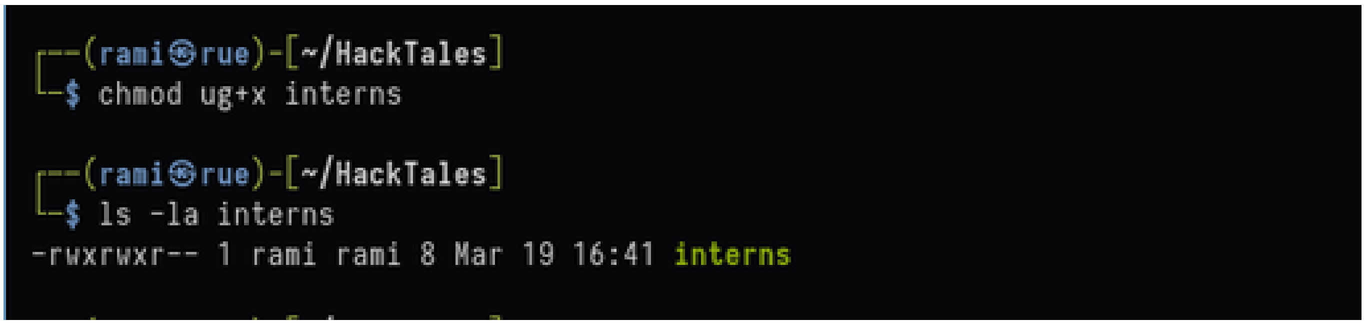
The above command reads like this: change permission on “interns” by adding executable permission bit on the user set. So now the user has executable permission on this file!

Removing permission bit on a file can be done with the subtraction symbol



```
┌──(rami㉿rue)-[~/HackTales]
└─$ chmod u-x interns

┌──(rami㉿rue)-[~/HackTales]
└─$ ls -la interns
-rw-rw-r-- 1 rami rami 8 Mar 19 16:41 interns
```

Adding multiple permission bits on a file can be done by specifying the user (u) and group (g) along with the addition or subtraction symbol to perform the action.



```
┌──(rami㉿rue)-[~/HackTales]
└─$ chmod ug+x interns

┌──(rami㉿rue)-[~/HackTales]
└─$ ls -la interns
-rwxrwxr-- 1 rami rami 8 Mar 19 16:41 interns
```

There is another way to change permissions using numerical format. This method allows you to change permissions all at once. Instead of using r, w, or x to represent permissions, you'll use a numerical representation for a single permission set. So no need to specify the group with g or the user with u.

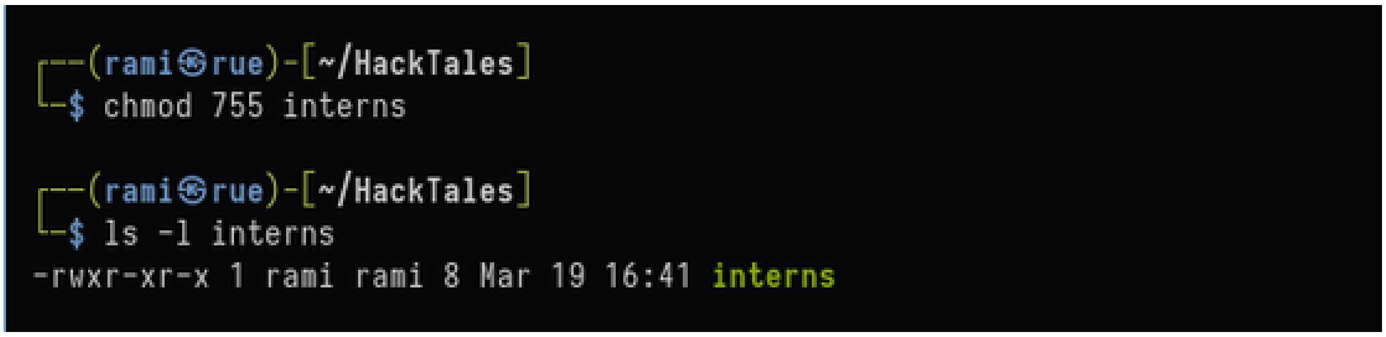
The numerical representations are seen below:

4: read permission

2: write permission

1: execute permission

Let's look at an example:



```
┌──(rami㉿rue)-[~/HackTales]
└─$ chmod 755 interns

┌──(rami㉿rue)-[~/HackTales]
└─$ ls -l interns
-rwxr-xr-x 1 rami rami 8 Mar 19 16:41 interns
```

Let's break this down, so now 755 covers the permissions for all sets. The first number (7) represents user permissions, the second number (5) represents group permissions and the last 5 represents other permissions.

Wait a minute, 7 and 5 weren't listed above, where are we getting these numbers? Remember we are combining all the permissions into one number now, so you'll have to get some math involved.

7 = 4 + 2 + 1, so 7 is the user permissions and it has read, write and execute permissions

5 = 4 + 1, the group has read and execute permissions

5 = 4 +1, and all other users have read and execute permissions

Now try to set a 700, what do you think that would be?