



Introduction to Linux Systems Administration



TABLE OF CONTENTS

1. System Administration Overview: 4
2. The Linux File System: 48
3. Linux File Editors: 113
4. Copying, Moving, and Deleting Files: 153
5. Linux Command Types: 183
6. Hard versus Soft Links: 208
7. Superusers and the Root Login : 239
8. Controlling the Population: 253
9. Piping and I/O Redirection: 317
10. Analyzing and Manipulating Files: 339
11. Find/Locate Files : 369
12. Managing and Identifying Software Packages: 382



TABLE OF CONTENTS

- 13. Controlling Processes: 410
- 14. The Power of Sudo (su and sudo Commands): 437
- 15. Basic Networking: 455
- 16. Shell Scripting Overview: 485
- 17. Controlling Processes: cron and crontab: 539
- 18. System Backups: 553
- 19. Creating Aliases: 574
- 20. File and Disk Management Tools: 593
- 21. LAMP Server Basics: 644
- 22. The Samba File Sharing Facility: 679
- 23. Networked File Systems (NFS): 707

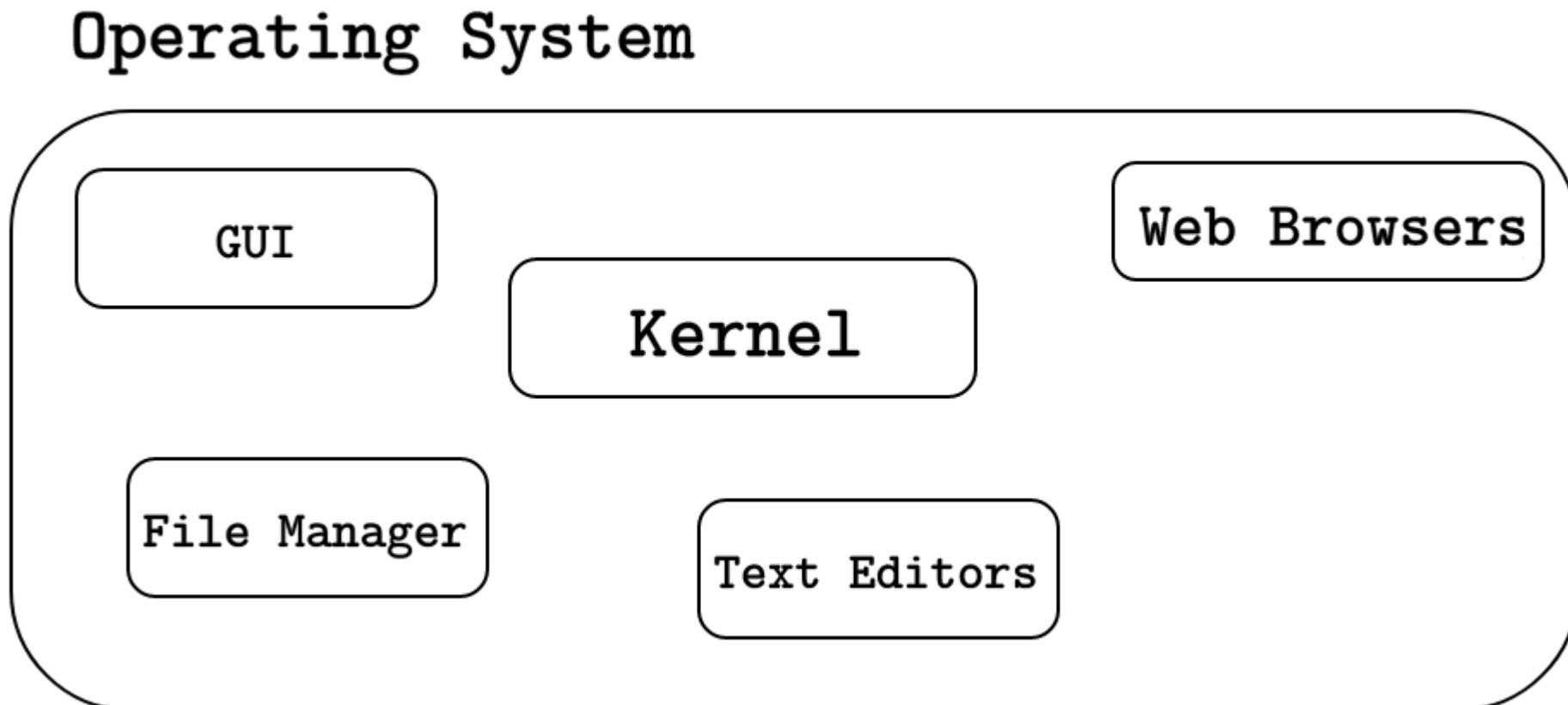


1. System Administration Overview

A blurred background image of a person's hands typing on a laptop keyboard, positioned on the left side of the slide.

System Administration Overview

- Following figure can help you visualize the difference between a kernel and an operating system.

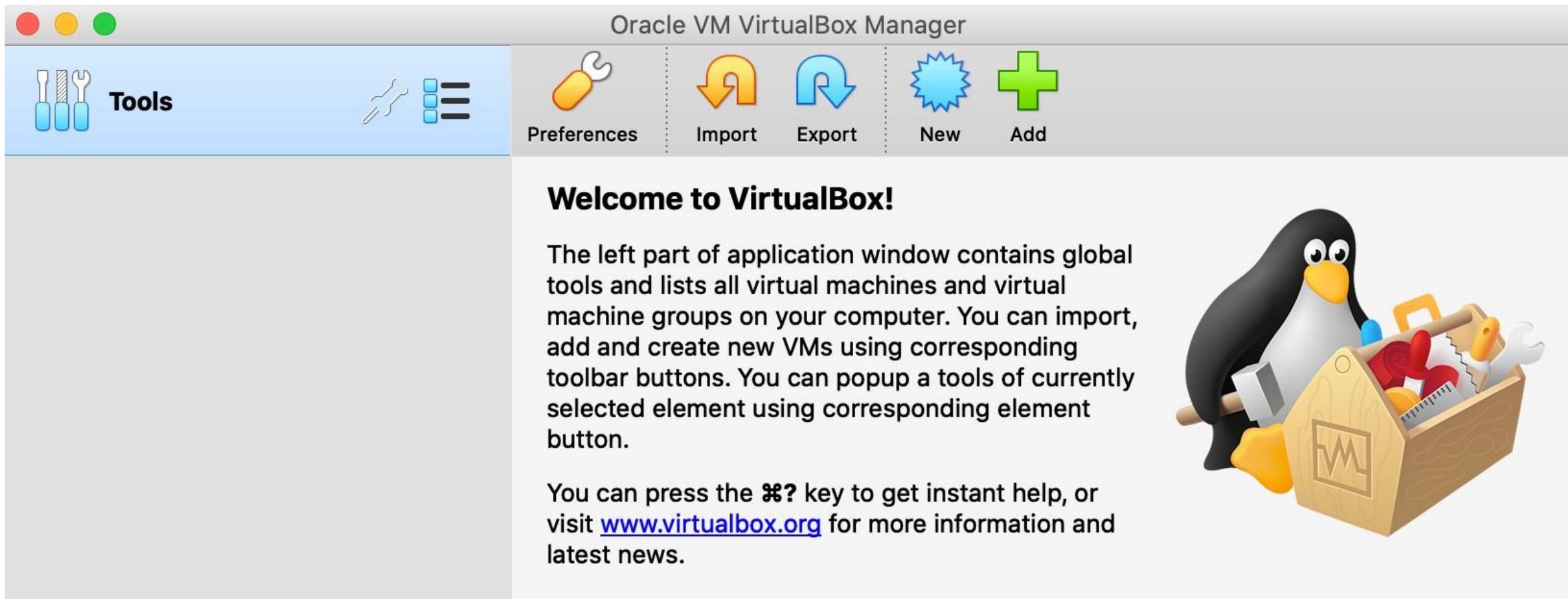


Linux today and the future

- In 1991, Linux was just a little baby. But this baby grew massively, and it became so popular.
- Today, Linux powers over 90% of the world's top supercomputers. And to add to your surprise, you may have been using Linux for years without noticing.
- How? Well, if you ever used an Android smartphone, then you have used Linux, and that's because Android is a Linux distribution! And if you still don't believe me, go to distrowatch.com and search for Android.
- On a more serious matter, the majority of government servers run Linux, and that's why you will see a lot of government technical jobs requiring Linux-skilled individuals.

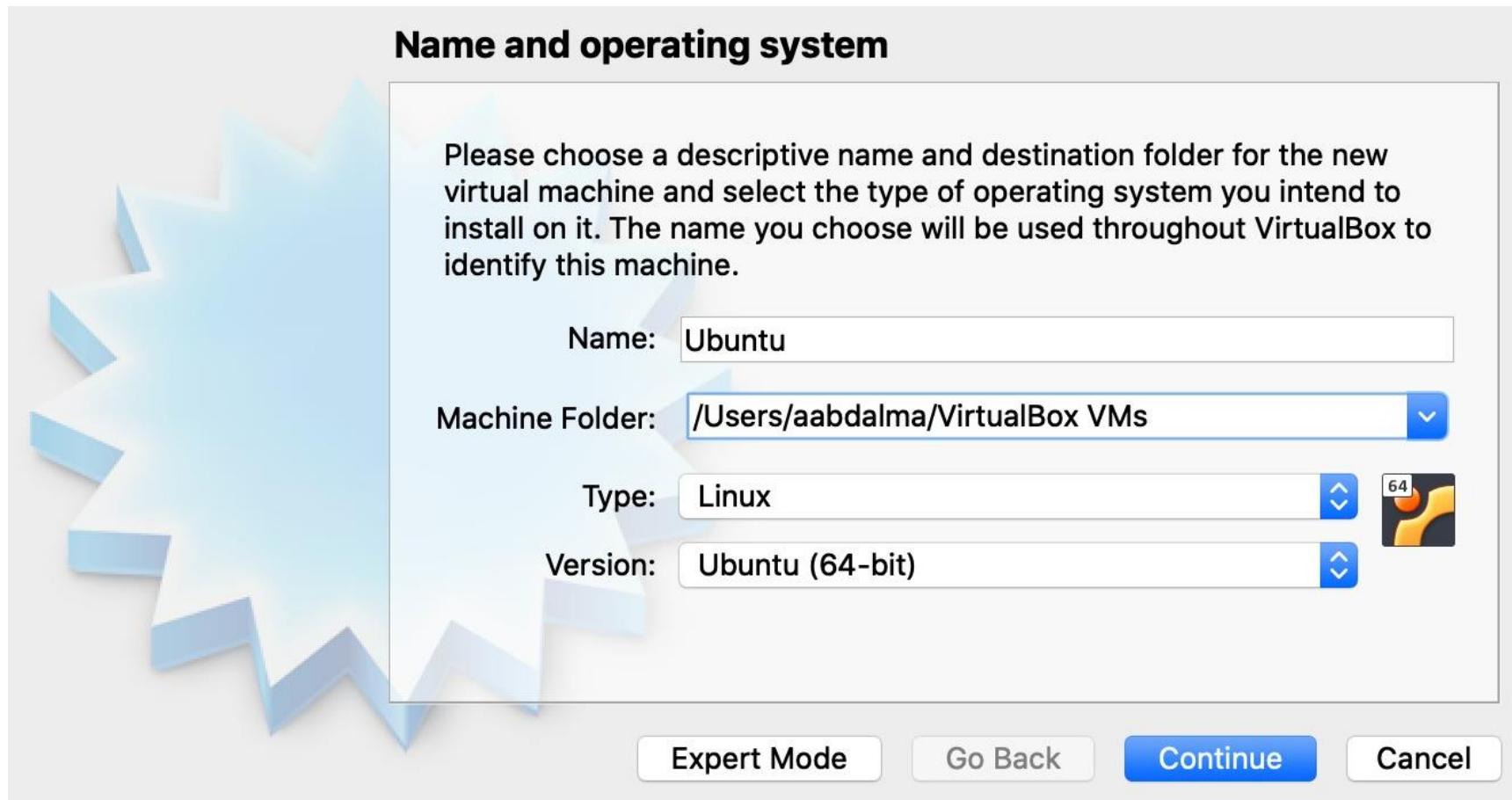
Installing a Linux virtual machine

- When you open VirtualBox, you have to select New from the menu bar.



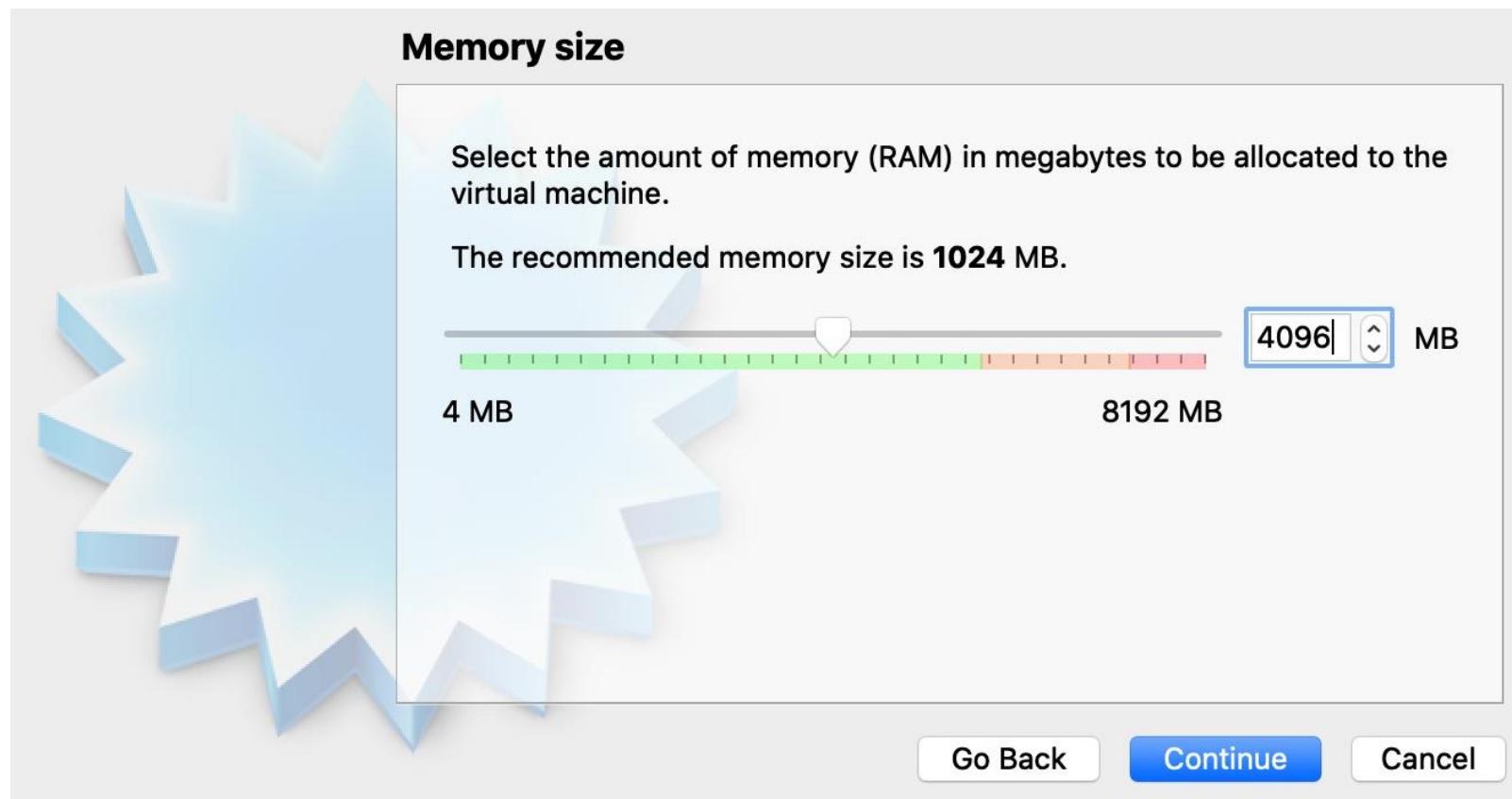
Installing a Linux virtual machine

- you need to choose the name and type of your new virtual machine.



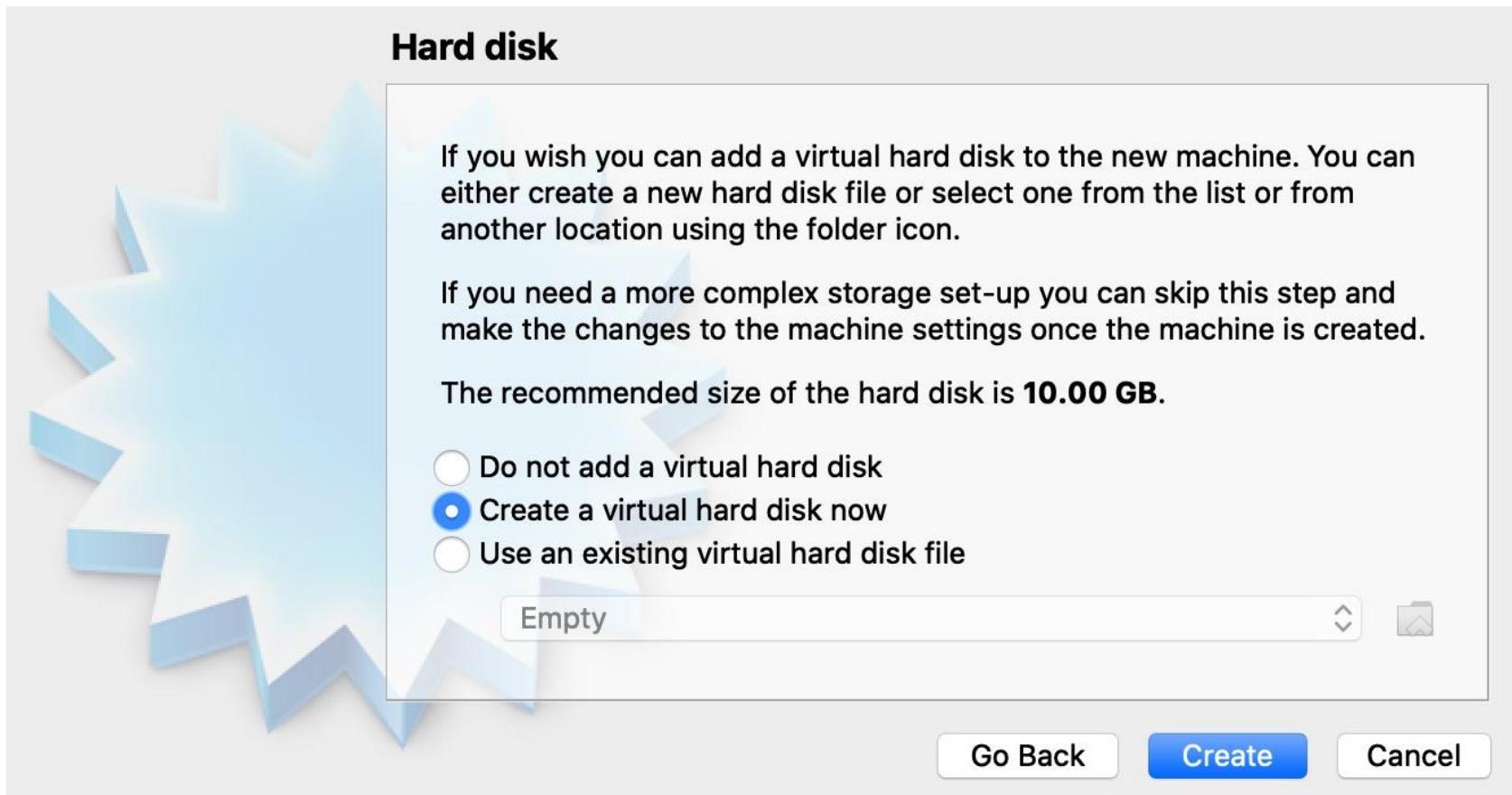
Installing a Linux virtual machine

- I chose to give my virtual machine 4096 MB of memory (RAM), which is equivalent to 4 GB.

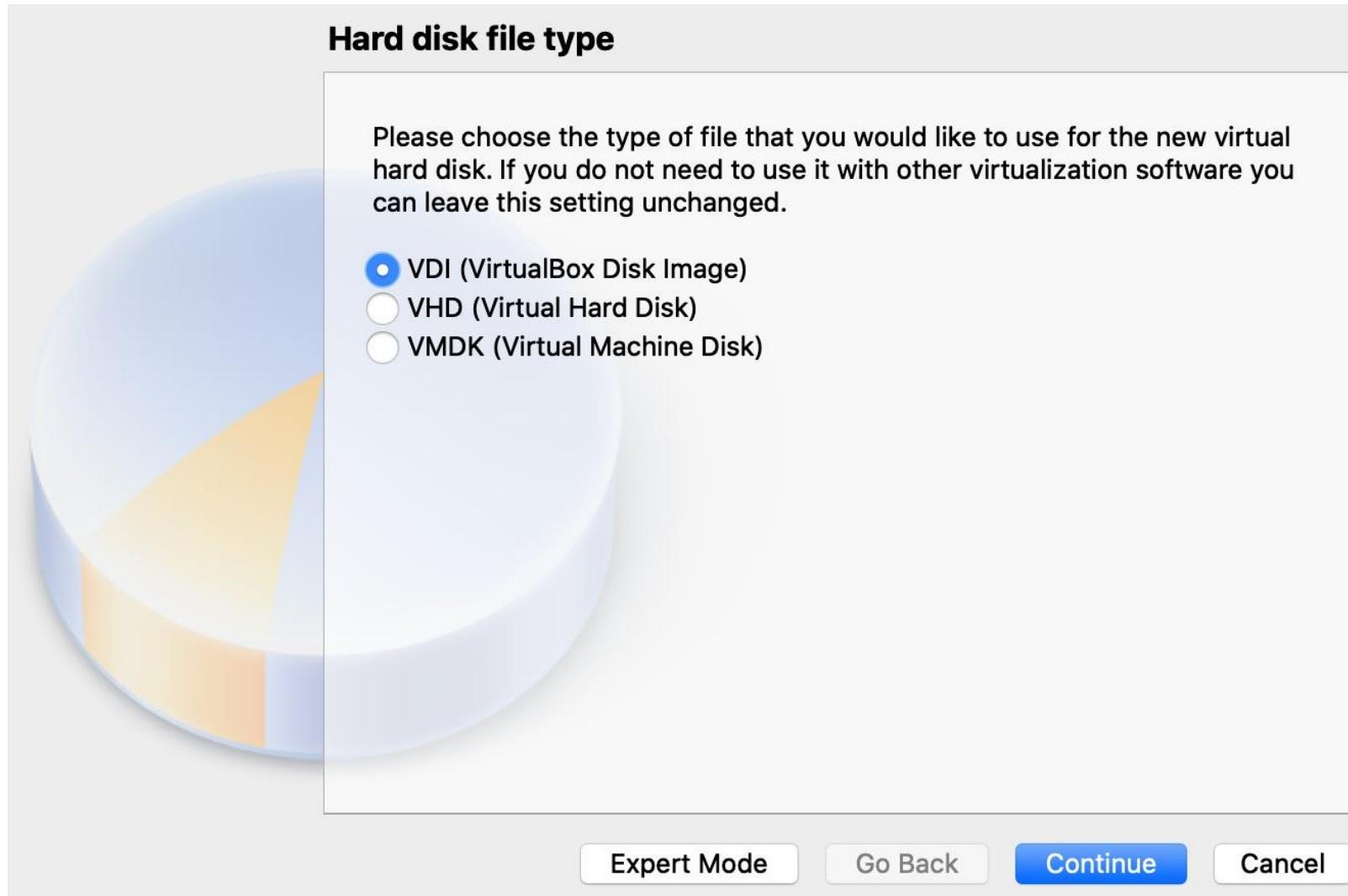


Installing a Linux virtual machine

- Click on Continue and make sure that Create a virtual hard disk now is selected, as shown in the following screenshot, then click on Create.



Installing a Linux virtual machine



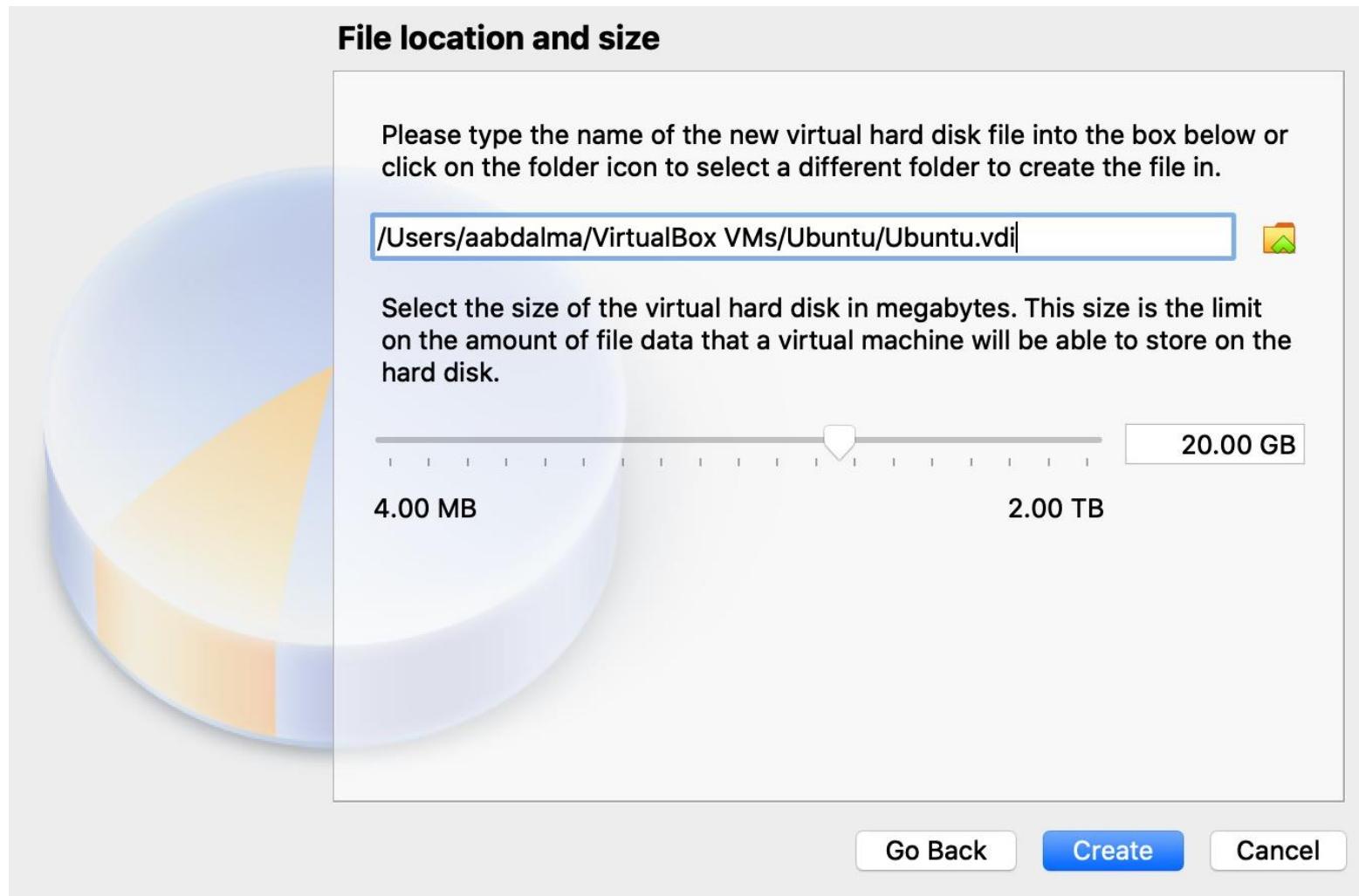
- Choose VDI (VirtualBox Disk Image) as shown in the following screenshot, then click on Continue.

Installing a Linux virtual machine

- Select **Dynamically allocated**, as shown in the following screenshot, then click on Continue.



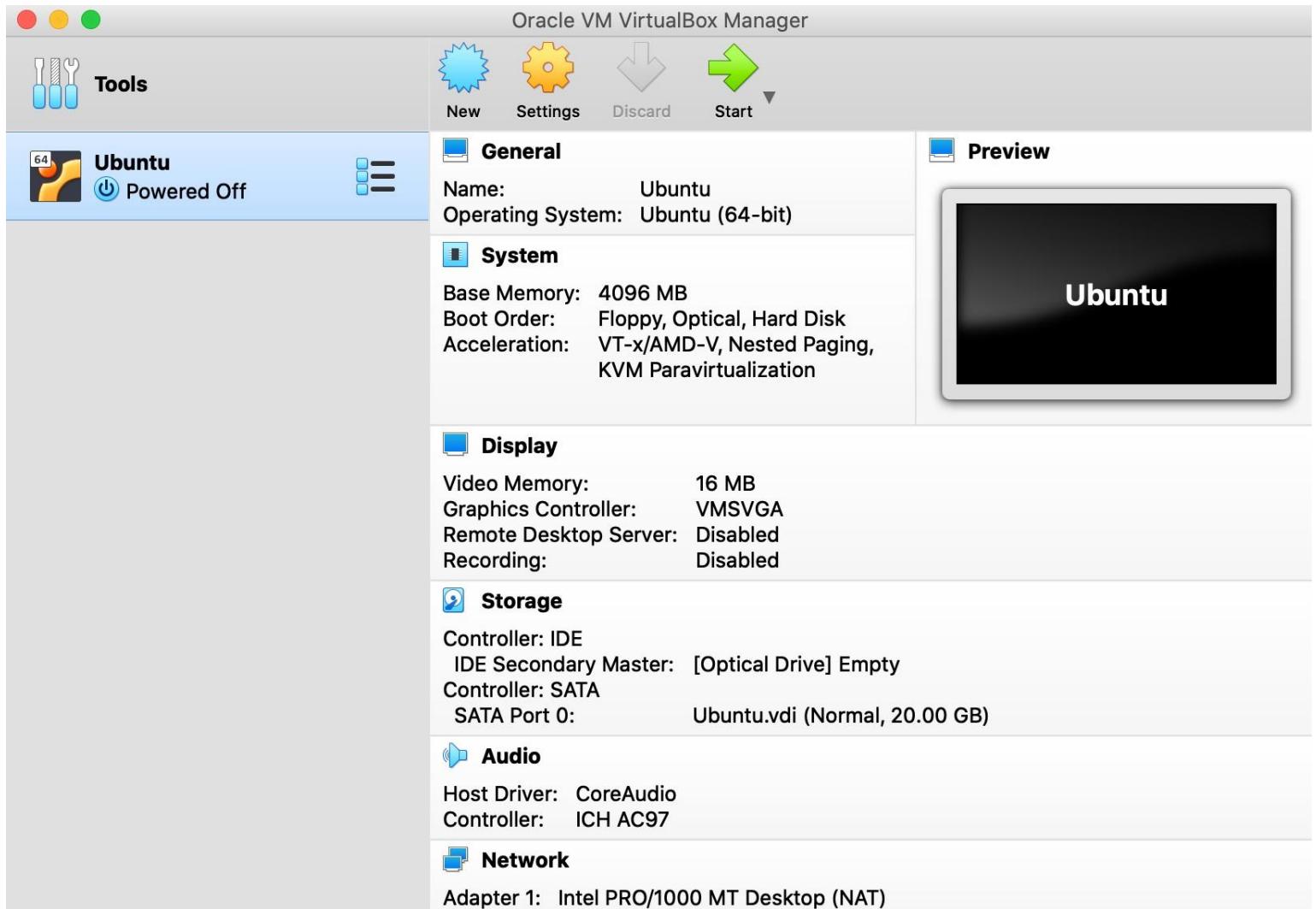
Installing a Linux virtual machine



- I chose 20 GB for my virtual machine.

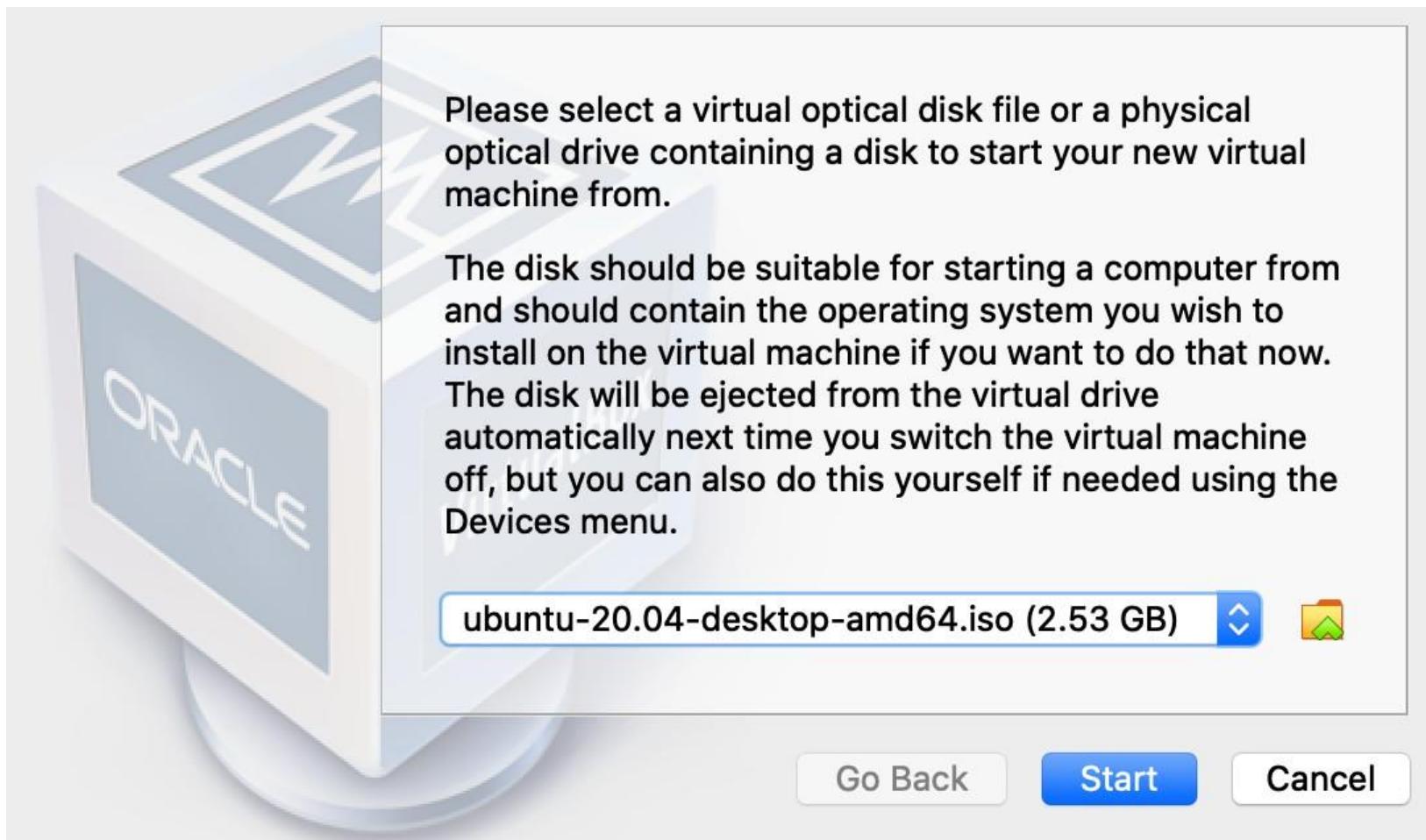
Installing a Linux virtual machine

- After selecting the hard disk size, click on Create to finish creating your virtual machine.

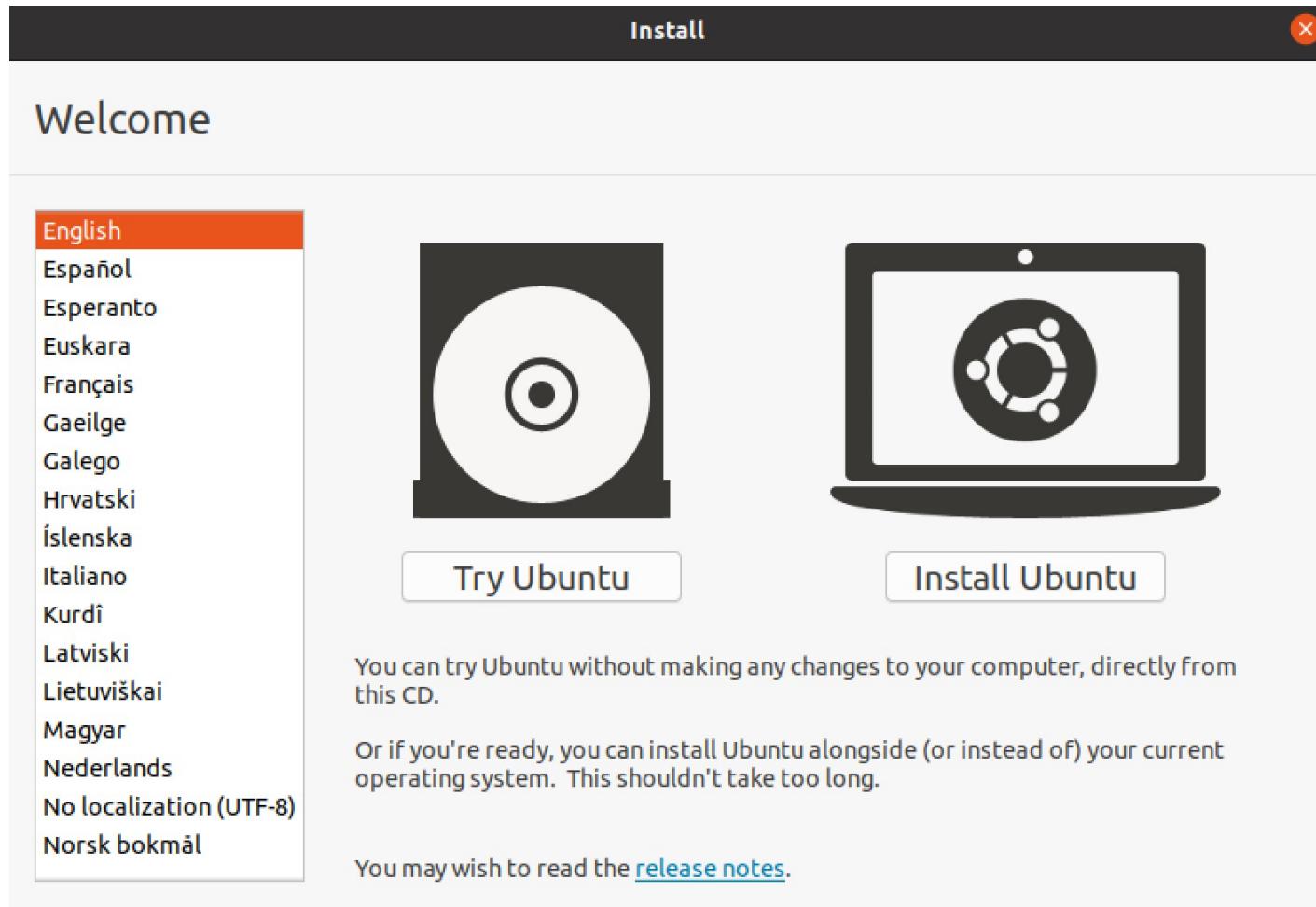


Installing a Linux virtual machine

- You will then have to select a start-up disk, as shown in the following screenshot.



Installing a Linux virtual machine



- Choose the Ubuntu ISO image that you have downloaded and then click on Start to launch the Ubuntu Installer, as shown in the following screenshot.

Installing a Linux virtual machine

- You will eventually come to the step of creating a new user, as shown in the following screenshot.

Install

Who are you?

Your name: Elliot Alderson ✓

Your computer's name: ubuntu-linux ✓
The name it uses when it talks to other computers.

Pick a username: elliot ✓

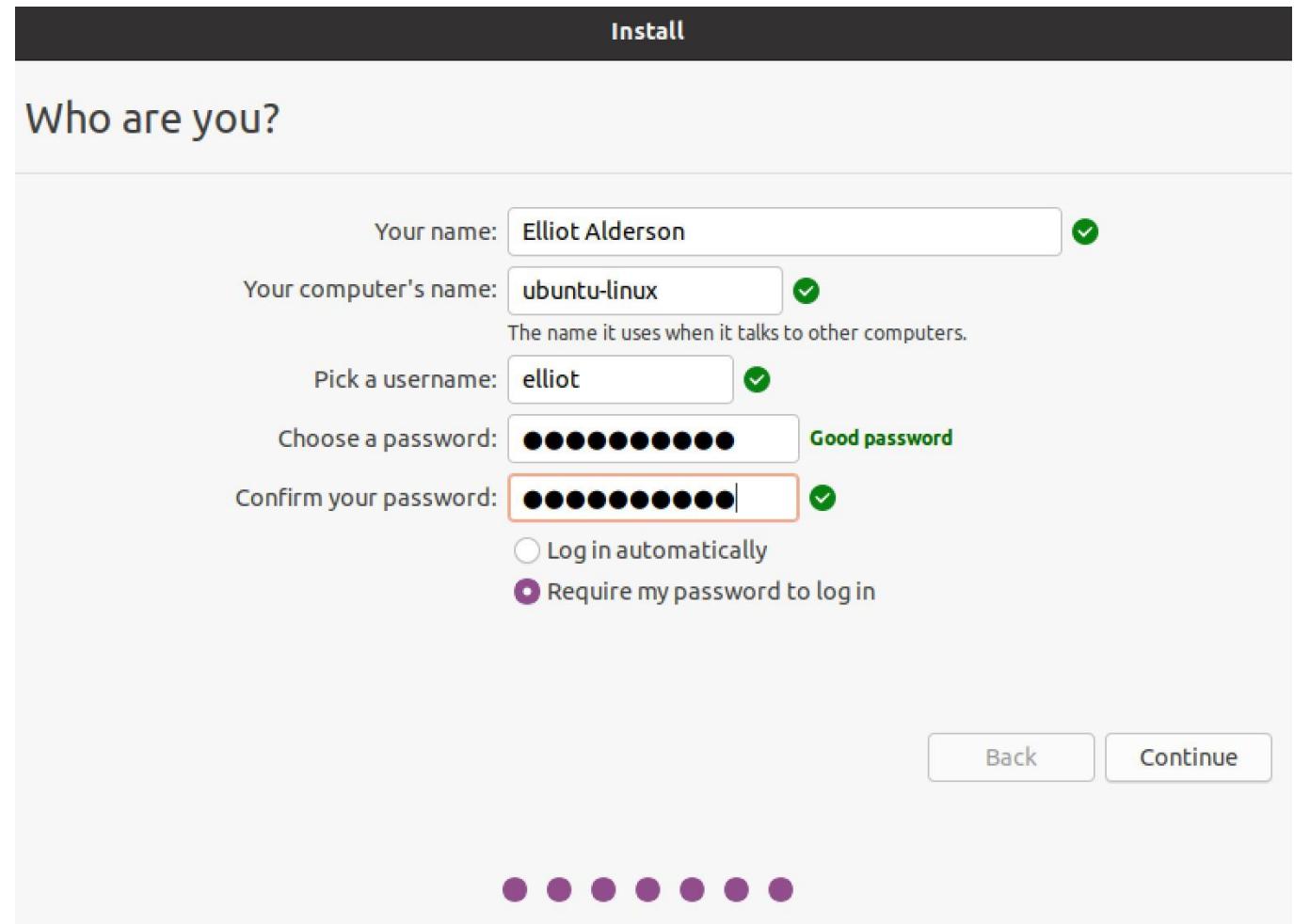
Choose a password: ███████████ Good password

Confirm your password: ███████████ ✓

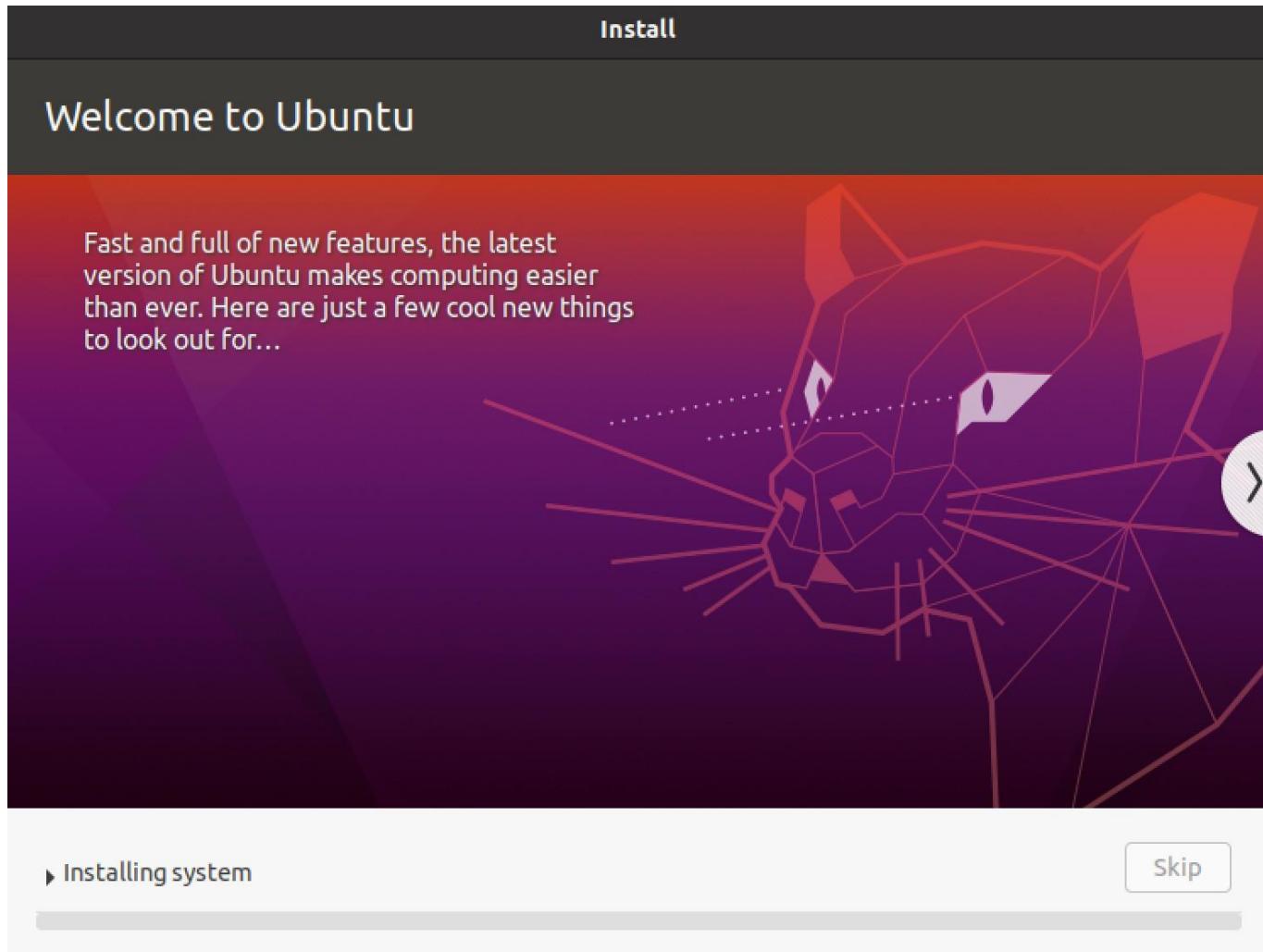
Log in automatically
 Require my password to log in

Back Continue

• • • • • •



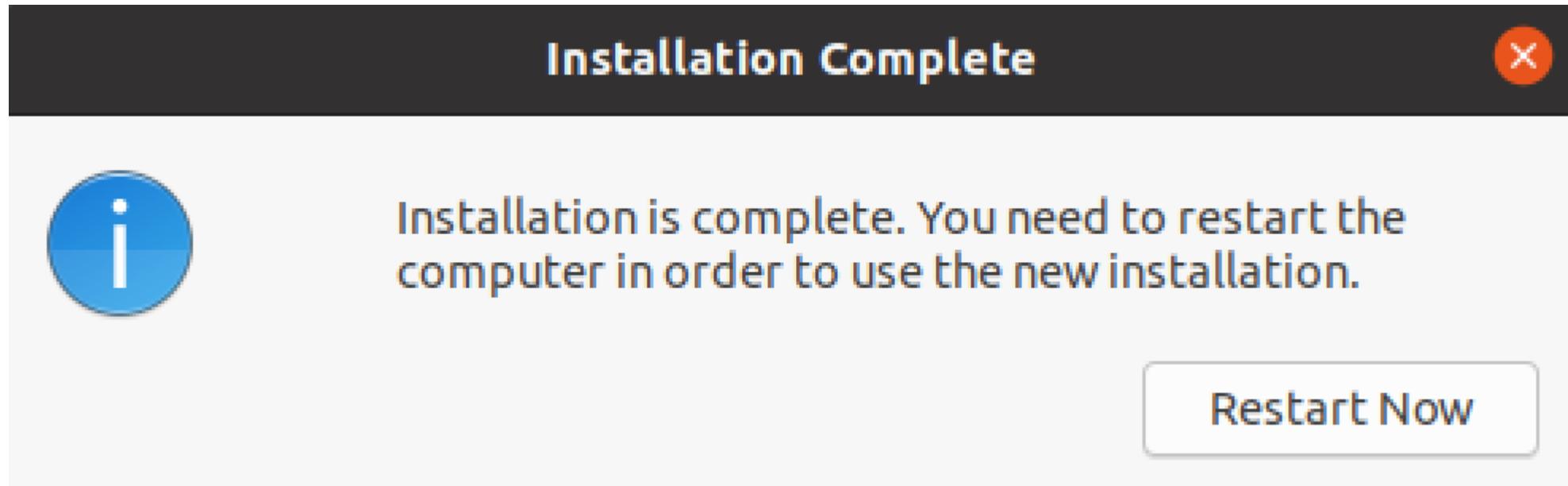
Installing a Linux virtual machine



- You can then click on Continue, and the system installation will begin, as shown in the following screenshot.

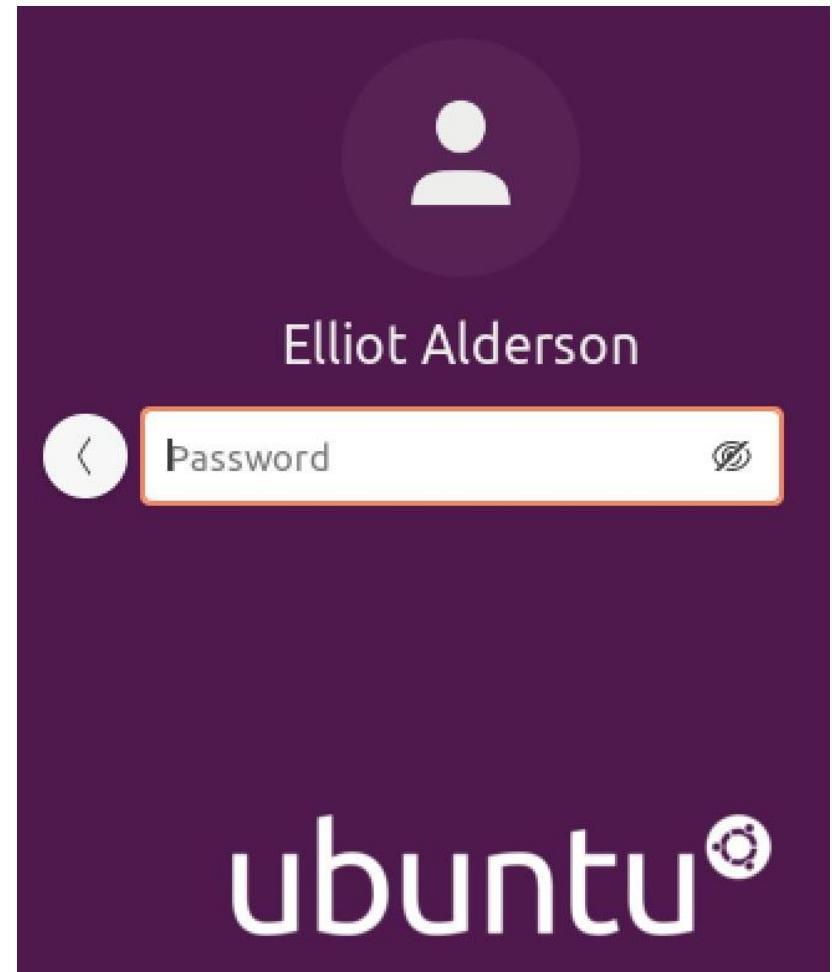
Installing a Linux virtual machine

- You will have to restart your virtual machine when the installation is complete, as shown in the following screenshot.



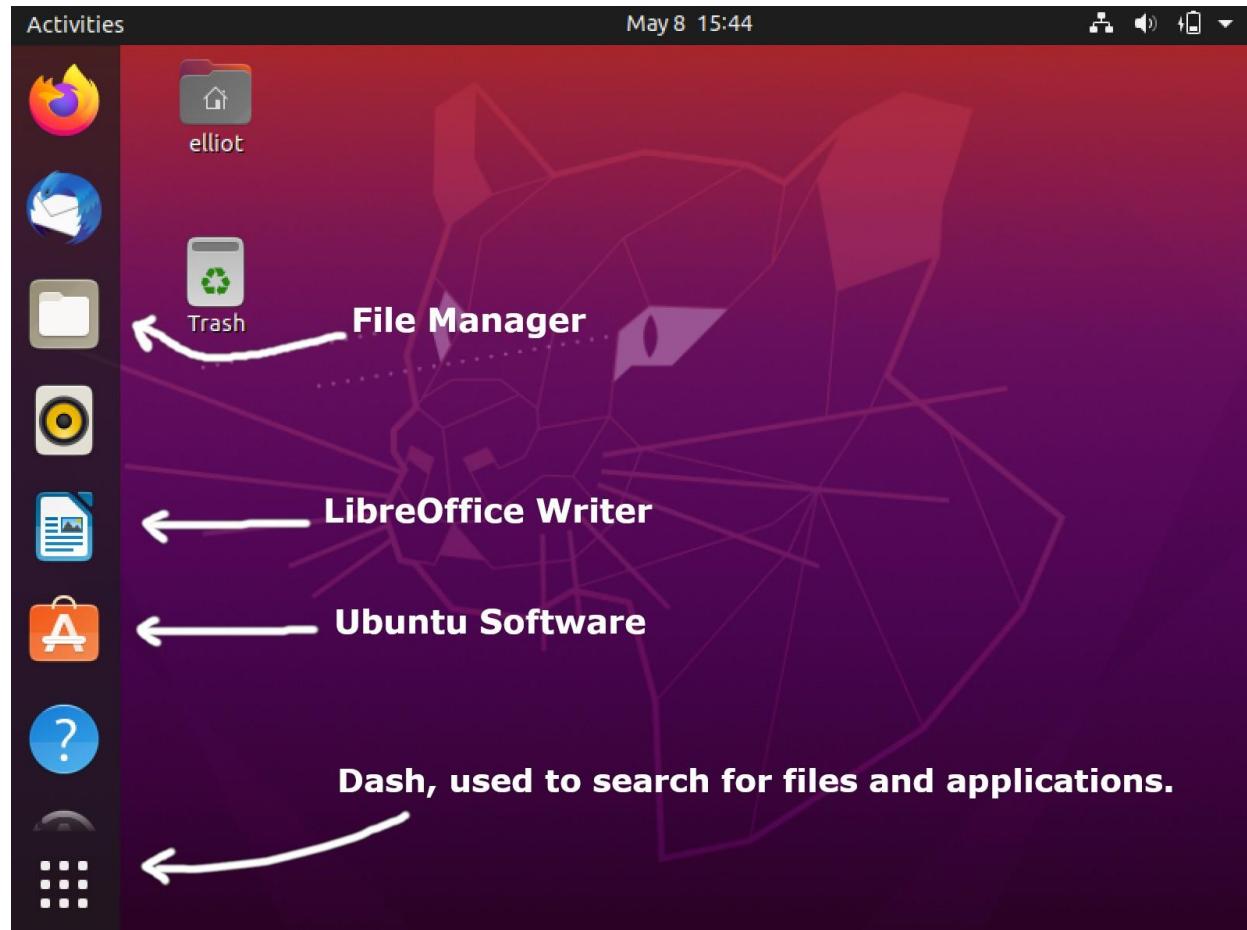
Installing a Linux virtual machine

- Finally, you should see your Sign In screen, as shown in the following screenshot.



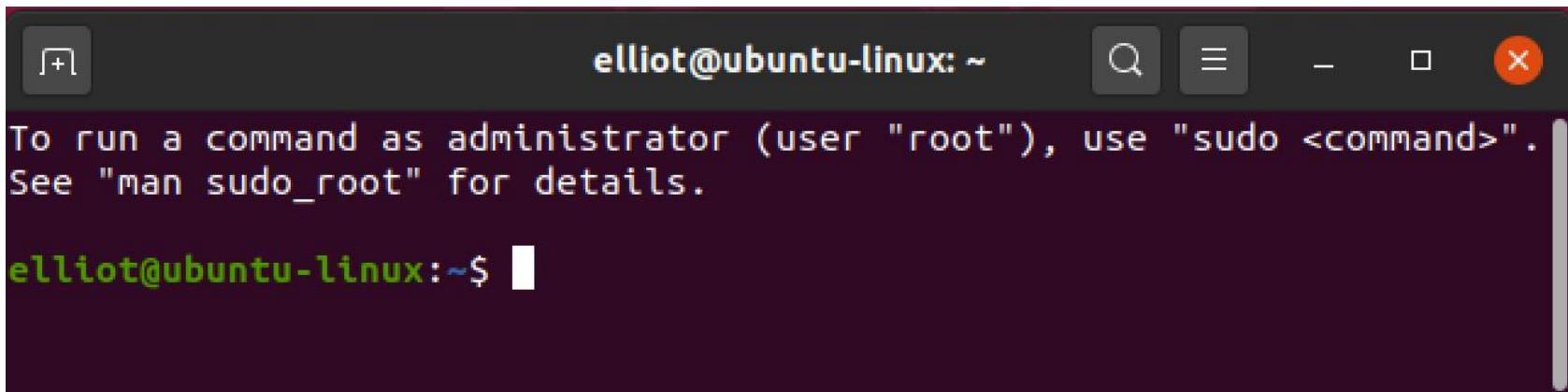
Terminal versus Shell

- The graphical user interface (GUI) is pretty self-explanatory.
- You can easily get around and connect to the internet and open up your web browser.
- All of that is pretty easy, as you can see in the following screenshot.



Terminal versus Shell

- You can also use the shortcut Ctrl+Alt+T to open the Terminal.
- When the Terminal opens, you will see a new window, as shown in the following screenshot.



- It looks kind of similar to the Command Prompt on Microsoft Windows.
Alright, now type date on your Terminal and then hit Enter:

```
elliot@ubuntu-linux:-$ date  
Tue Feb 17 16:39:13 CST 2020
```

Terminal versus Shell

- You shouldn't be confused between the Terminal and the Shell.
- The Terminal is the window you see on your screen where you can type in your commands while the Shell is responsible for executing the commands. That's it, nothing more and nothing less.
- You should also know that if you type any gibberish, you will get a command not found error as shown in the following example:

```
elliot@ubuntu-linux:~$ blabla  
blabla: command not found
```

A few simple commands

- One would usually display the calendar after displaying that date, right? To display the calendar of the current month, you can run the cal command:

```
[elliot@ubuntu-linux:~$ cal
```

```
February 2020
```

Su	Mo	Tu	We	Th	Fr	Sa
----	----	----	----	----	----	----

						1
--	--	--	--	--	--	---

2	3	4	5	6	7	8
---	---	---	---	---	---	---

9	10	11	12	13	14	15
---	----	----	----	----	----	----

16	17	18	19	20	21	22
----	----	----	----	----	----	----

23	24	25	26	27	28	29
----	----	----	----	----	----	----

A few simple commands

- You can also display the calendar of the whole year, for example, to get the full 2022 calendar, you can run:

```
[elliott@ubuntu-linux:~$ cal 2022
          2022
January           February          March
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
                         1       1   2   3   4   5       1   2   3   4   5
2   3   4   5   6   7   8   6   7   8   9   10  11  12   6   7   8   9   10  11  12
9   10  11  12  13  14  15  13  14  15  16  17  18  19  13  14  15  16  17  18  19
16  17  18  19  20  21  22  20  21  22  23  24  25  26  20  21  22  23  24  25  26
23  24  25  26  27  28  29  27  28
30  31

April            May              June
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
                         1   2       1   2   3   4   5   6   7       1   2   3   4
3   4   5   6   7   8   9   8   9   10  11  12  13  14   5   6   7   8   9   10  11
10  11  12  13  14  15  16  15  16  17  18  19  20  21  12  13  14  15  16  17  18
17  18  19  20  21  22  23  22  23  24  25  26  27  28  19  20  21  22  23  24  25
24  25  26  27  28  29  30  29  30  31
26  27  28  29  30

July             August            September
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
                         1   2       1   2   3   4   5   6       1   2   3
3   4   5   6   7   8   9   7   8   9   10  11  12  13   4   5   6   7   8   9   10
10  11  12  13  14  15  16  14  15  16  17  18  19  20  11  12  13  14  15  16  17
17  18  19  20  21  22  23  21  22  23  24  25  26  27  18  19  20  21  22  23  24
24  25  26  27  28  29  30  28  29  30  31
31

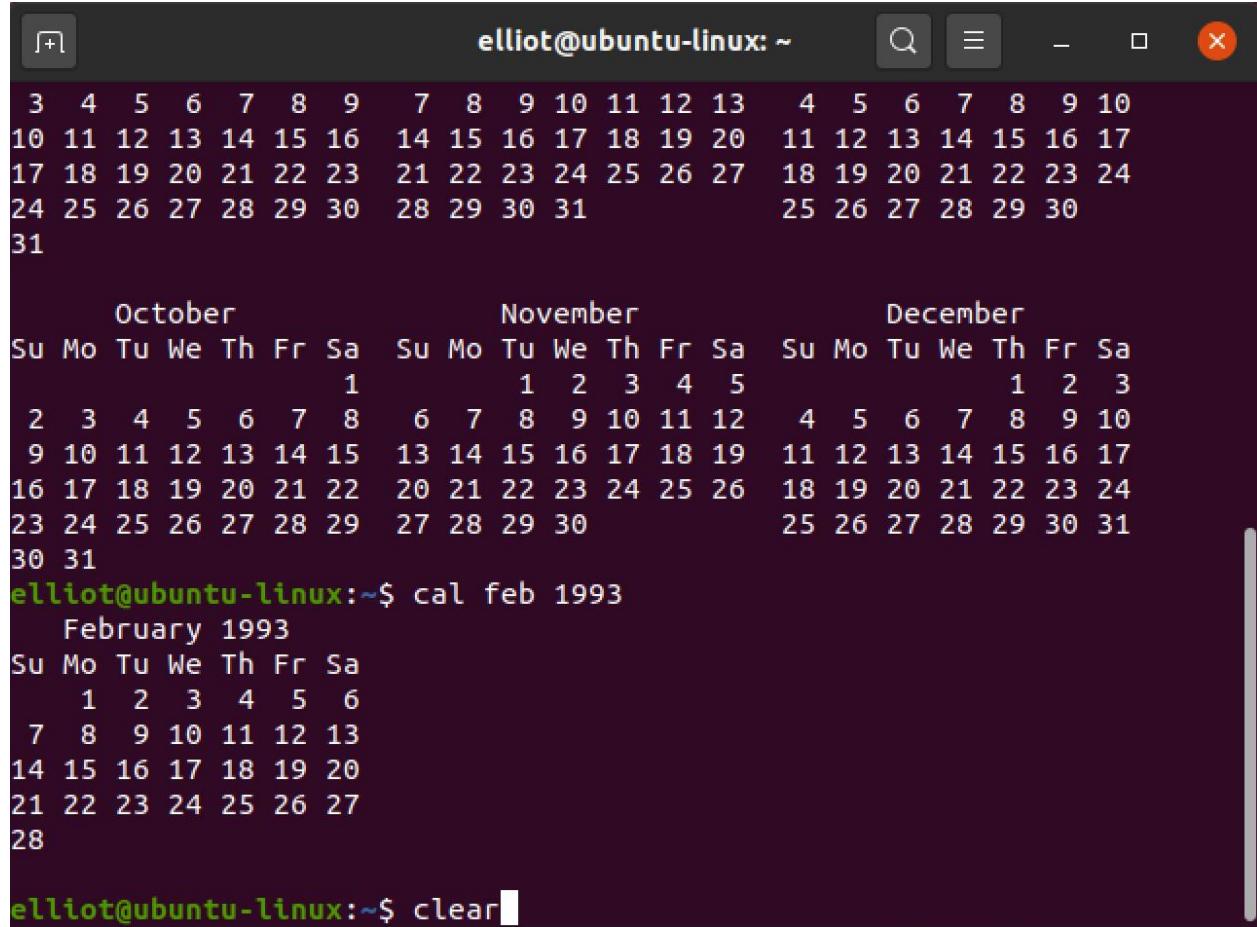
October          November          December
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
                         1       1   2   3   4   5       1   2   3
2   3   4   5   6   7   8   6   7   8   9   10  11  12   4   5   6   7   8   9   10
9   10  11  12  13  14  15  13  14  15  16  17  18  19  11  12  13  14  15  16  17
16  17  18  19  20  21  22  20  21  22  23  24  25  26  18  19  20  21  22  23  24
23  24  25  26  27  28  29  27  28  29  30
30  31
```

A few simple commands

- You can also specify a month, for example, to display the calendar of February 1993, you can run the command:

```
[elliot@ubuntu-linux:~$ cal feb 1993
          February 1993
Su Mo Tu We Th Fr Sa
      1  2  3  4  5  6
  7  8  9  10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28
```

A few simple commands



The terminal window shows the following output:

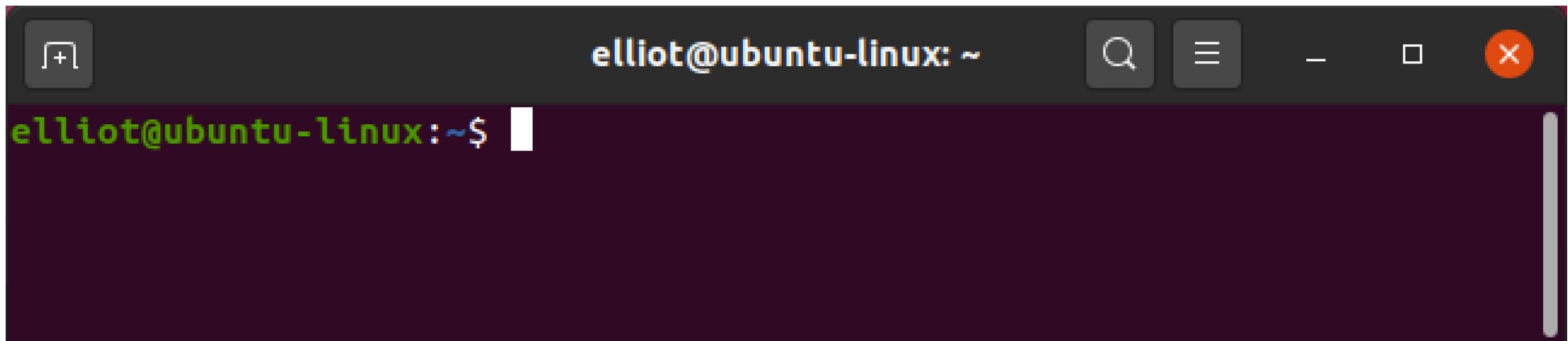
```
elliot@ubuntu-linux:~$ cal feb 1993
February 1993
Su Mo Tu We Th Fr Sa
      1   2   3   4   5   6
   7   8   9   10  11  12  13
  14  15  16  17  18  19  20
  21  22  23  24  25  26  27
  28  29  30  31

elliot@ubuntu-linux:~$ clear
```

- You now have a lot of output on your Terminal.
- You can run the clear command to clear the Terminal screen:

A few simple commands

- This is how your Terminal will look after running the clear command:



A few simple commands

- You can use the lscpu command, which is short for List CPU, to display your CPU architecture information:

```
elliot@ubuntu-linux:~$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                1
On-line CPU(s) list:  0
Thread(s) per core:   1
Core(s) per socket:   1
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 61
Model name:            Intel(R) Core(TM) i5-5300U CPU @ 2.30GHz Stepping: 4
CPU MHz:               2294.678
BogoMIPS:              4589.35
Hypervisor vendor:    KVM
Virtualization type:  full
L1i cache:             32K
L1d cache:             32K
L2 cache:              256K
L3 cache:              3072K
NUMA node0 CPU(s):    0
Flags:                 fpu vme de pse tsc msr pae mce cx8 apic sep mtrr
```

A few simple commands

```
elliot@ubuntu-linux:~$ uptime  
18:48:04 up 4 days, 4:02, 1 user, load average: 0.98, 2.12, 3.43
```

You might be intimidated by the output of the `uptime` command, but don't worry, the following table breaks down the output for you.

18:48:04	The first thing you see in the output is the current time.
up 4 days, 4:02	This is basically saying that the system has been up and running for 4 days, 4 hours, and 2 minutes.
1 user	Only one user is currently logged in.
load average: 0.98, 2.12, 3.43	The system load averages for the past 1, 5, and 15 minutes.

A few simple commands

- The three numbers that you see at the very end of the uptime command output are the load averages over 1, 5, and 15 minutes respectively.
- For example, if your load averages values are:

load average: 2.00, 4.00, 6.00

Then these three numbers represent the following:

- 2.00 --+: The load average over the last minute.
- 4.00 --+: The load average over the last five minutes.
- 6.00 --+: The load average over the last fifteen minutes.

A few simple commands

- For instance, load averages of:

load average: 1.00, 3.00, 7.00

- Shows that the system load is decreasing over time.
- On the other hand, load averages of:

load average: 5.00, 3.00, 2.00

A few simple commands

- You can run the reboot command to restart your system:

```
elliot@ubuntu-linux:-$ reboot
```

- You can run the pwd command to print the name of your current working directory:

```
elliot@ubuntu-linux:-$ pwd  
/home/elliot
```

- The current working directory is the directory in which a user is working at a given time, By default, when you log into your Linux system, your current working directory is set to your home directory:

```
/home/your_username
```

A few simple commands

- You can run the ls command to list the contents of your current working directory:

```
elliot@ubuntu-linux:~$ ls
```

```
Desktop Documents Downloads Music Pictures Public Videos
```

- If you want to change your password, you can run the passwd command:

```
elliot@ubuntu-linux:~$ passwd
```

```
Changing password for elliot.
```

```
(current) UNIX password:
```

```
Enter new UNIX password:
```

```
Retype new UNIX password:
```

```
passwd: password updated successfully
```

A few simple commands

- You can use the hostname command to display your system's hostname:

```
elliot@ubuntu-linux:-$ hostname  
ubuntu-linux
```

- You can use the free command to display the amount of free and used memory on your system:

```
elliot@ubuntu-linux:-$ free  
              total    used    free   shared  buff/cache available  
Mem:  4039732  1838532  574864  71900   1626336  1848444  
Swap: 969960      0  969960
```

A few simple commands

- By default, the free command displays the output in kilobytes, but only aliens will make sense out of this output.
- You can get an output that makes sense to us humans by running the free command with the -h option:

```
elliot@ubuntu-linux:~$ free -h
```

	total	used	free	shared	buff/cache	available
Mem:	3.9G	1.8G	516M	67M	1.6G	1.7G
Swap:	947M	0B	947M			

A few simple commands

- On the other hand, if you are using the full name of the command option, then you need to use a double hyphen:

```
elliot@ubuntu-linux:~$ free --human
```

	total	used	free	shared	buff/cache	available
Mem:	3.9G	1.8G	516M	67M	1.6G	1.7G
Swap:	947M	0B	947M			

A few simple commands

- You can use the df command to display the amount of disk space available on your system:

```
elliot@ubuntu-linux:~$ df
```

Filesystem	1K-blocks	Used	Available	Use%	Mounted on
udev	1989608	0	1989608	0%	/dev
tmpfs	403976	1564	402412	1%	/run
/dev/sda1	20509264	6998972	12445436	36%	/
tmpfs	2019864	53844	1966020	3%	/dev/shm
tmpfs	5120	4	5116	1%	/run/lock
tmpfs	2019864	0	2019864	0%	/sys/fs/cgroup
/dev/loop0	91648	91648	0	100%	/snap/core/6130
tmpfs	403972	28	403944	1%	/run/user/121
tmpfs	403972	48	403924	1%	/run/user/1000

A few simple commands

- Again you may want to use the human-readable option -h to display a nicer format:

```
elliot@ubuntu-linux:~$ df -h
```

Filesystem	Size	Used	Avail	Use%	Mounted on
udev	1.9G	0	1.9G	0%	/dev
tmpfs	395M	1.6M	393M	1%	/run
/dev/sda1	20G	6.7G	12G	36%	/
tmpfs	2.0G	57M	1.9G	3%	/dev/shm
tmpfs	5.0M	4.0K	5.0M	1%	/run/lock
tmpfs	2.0G	0	2.0G	0%	/sys/fs/cgroup
/dev/loop0	90M	90M	0	100%	/snap/core/6130
tmpfs	395M	28K	395M	1%	/run/user/121
tmpfs	395M	48K	395M	1%	/run/user/1000

A few simple commands

- The echo command is another very useful command; it allows you to print a line of text on your Terminal.
- For example, if you want to display the line Cats are better than Dogs! on your Terminal, then you can run:

```
elliot@ubuntu-linux:~$ echo Cats are better than Dogs!  
Cats are better than Dogs!
```

A few simple commands

- Let's run the history command and see what we get here:

```
elliott@ubuntu-linux:~$ history  
1 date  
2 blabla  
3 cal  
4 cal 2022  
5 cal feb 1993  
6 clear  
7 lscpu  
8 uptime  
9 reboot  
10 pwd  
11 ls  
12 passwd  
13 hostname  
14 free  
15 free -h  
16 free --human  
17 df  
18 df -h  
19 echo Cats are better than Dogs!  
20 history
```

A few simple commands

- On my history list, the lscpu command is number 7, so if I want to rerun lscpu, all I need to do is run !7:

```
elliot@ubuntu-linux:~$ !7
lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                1
On-line CPU(s) list:  0
Thread(s) per core:   1
Core(s) per socket:   1
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 61
Model name:            Intel(R) Core(TM) i5-5300U CPU @ 2.30GHz
Stepping:               4
CPU MHz:               2294.678
BogoMIPS:              4589.35
Hypervisor vendor:    KVM
Virtualization type:  full
L1 cache:              32K
L2 cache:              256K
L3 cache:              3072K
NUMA node0 CPU(s):    0
Flags:                 fpu vme de pse tsc msr pae mce cx8 apic sep mtrr
```

A few simple commands

- You can use the uname command to display your system's kernel information.
- When you run the uname command without any options, then it will print just the kernel name:

```
elliot@ubuntu-linux:-$ uname  
Linux
```

- You can use the -v option to print the current kernel version information:

```
elliot@ubuntu-linux:-$ uname -v  
#33-Ubuntu SMP Wed Apr 29 14:32:27 UTC 2020
```

A few simple commands

- You can also use the -r option to print the current kernel release information:

```
elliot@ubuntu-linux:-$ uname -r  
5.4.0-29-generic
```

- You can also use the -a option to print all the information of your current kernel at once:

```
elliot@ubuntu-linux:-$ uname -a  
Linux ubuntu-linux 5.4.0-29-generic #33-Ubuntu SMP  
Wed Apr 29 14:32:27 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
```

A few simple commands

- You can also run the `lsb_release -a` command to display the Ubuntu version you are currently running:

```
elliot@ubuntu-linux:-$ lsb_release -a
```

No LSB modules are available.

Distributor ID: Ubuntu

Description: Ubuntu 20.04 LTS

Release: 20.04

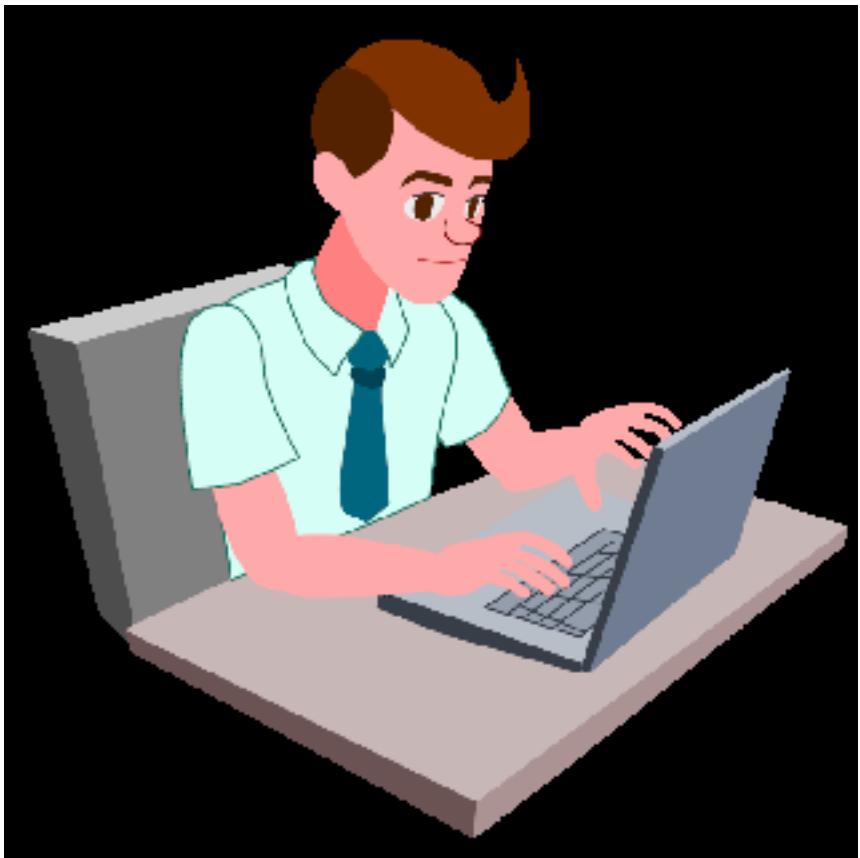
Codename: focal

- Finally, the last command you are going to learn in this lesson is the `exit` command, which terminates your current Terminal session:

```
elliot@ubuntu-linux:-$ exit
```

Knowledge check

- For the following exercises, open up your Terminal and try to solve the following tasks:
 1. Display the whole calendar for the year 2023.
 2. Display the memory information of your system in a human-readable format.
 3. Display the contents of your home directory.
 4. Change your current user password.
 5. Print the line "Mr. Robot is an awesome TV show!" on your Terminal.



"Complete Lab"

2. The Linux File System

The Linux filesystem

- You can use the cd command followed by a forward slash to get to the root:

```
elliot@ubuntu-linux:~$ cd /
```

- The cd command is short for Change Directory and is one of the most used commands in Linux.
- You can't move around in Linux without it. It's like your limbs (arms and legs), can you climb a tree without your limbs?
- The forward slash character represents the root directory.
- Now to make sure you're at the root directory, you can run pwd:

```
elliot@ubuntu-linux:~$ pwd
```

The Linux filesystem

- Alright, while we are still at the root directory, let's see what's in there! Run the ls command to view the contents of the current directory:

```
elliott@ubuntu-linux:/$ ls  
bin etc lib proc tmp var boot  
dev home opt root sbin usr
```

The Linux filesystem

- To have a better view of the contents, you can use the long listing -l option with the ls command:

```
elliott@ubuntu-linux:/$ ls -l
drwxr-xr-x  2 root root          4096 Dec 28 15:36 bin
drwxr-xr-x 125 root root        12288 Jan  1 11:01 etc
drwxr-xr-x  21 root root         4096 Dec 26 23:52 lib
dr-xr-xr-x 227 root root          0 Jan   3 02:33 proc
drwxrwxrwt  15 root root         4096 Jan   3 02:35 tmp
drwxr-xr-x  14 root root         4096 Jul 24 21:14 var
drwxr-xr-x  3 root root         4096 Dec 29 07:17 boot
drwxr-xr-x  18 root root        4000 Jan   3 02:33 dev
drwxr-xr-x  3 root root         4096 Dec 26 23:47 home
drwxr-xr-x  3 root root         4096 Dec 27 15:07 opt
drwx-----  4 root root         4096 Dec 29 09:39 root
drwxr-xr-x  2 root root        12288 Dec 28 15:36 sbin
drwxr-xr-x 10 root root         4096 Jul 24 21:03 usr
```

The Linux filesystem

- we focus on the first letter in the first column of the output.
- Take a look at the first column of the output:

drwxr-xr-x

drwxr-xr-x

drwxr-xr-x

drwxr-xr-x

.

.

.

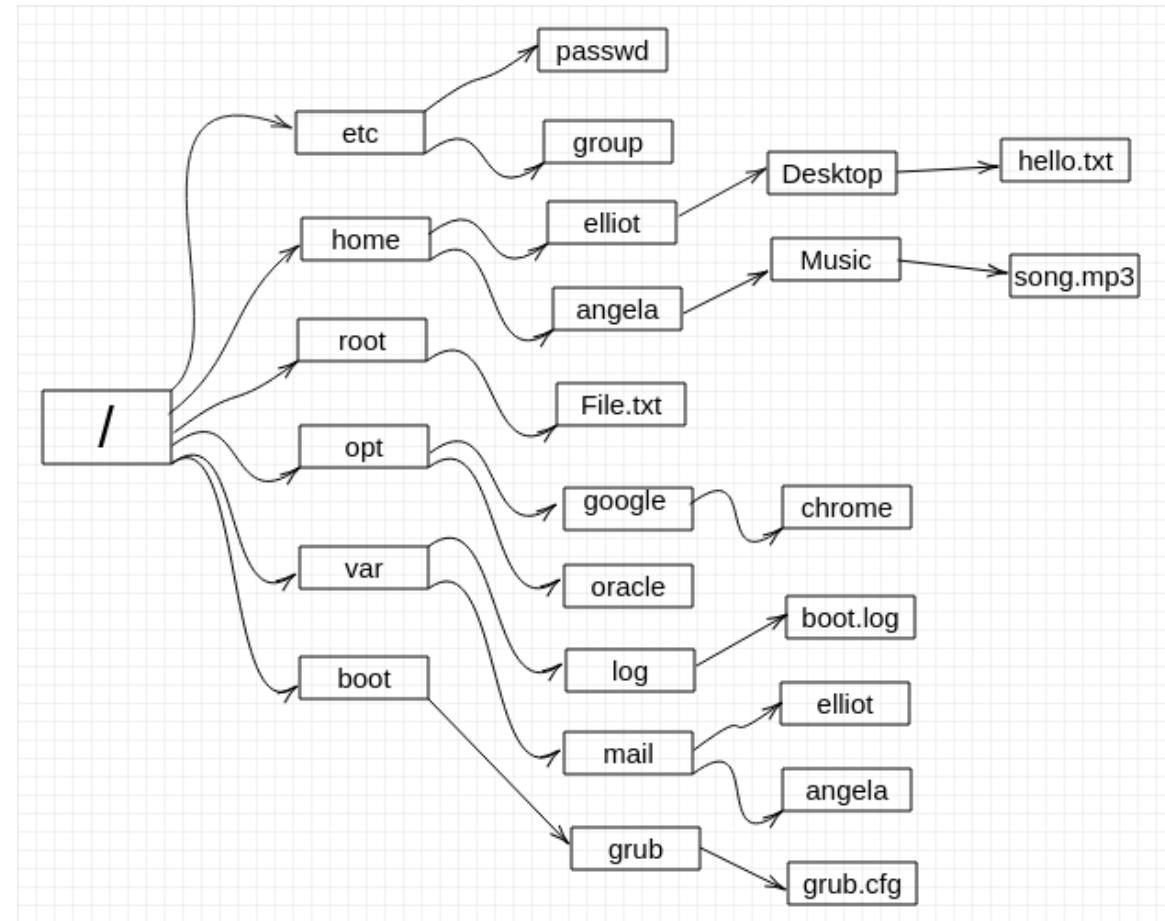
.

The Linux filesystem

- You can also run the `man hier` command to read more about the Linux filesystem hierarchy:

```
elliott@ubuntu-linux:/$ man hier
```

- Alright, now let's do further climbing on the Linux directory tree.
- Take a look at figure 1, and you will understand why we choose a tree to describe the structure of the Linux filesystem.



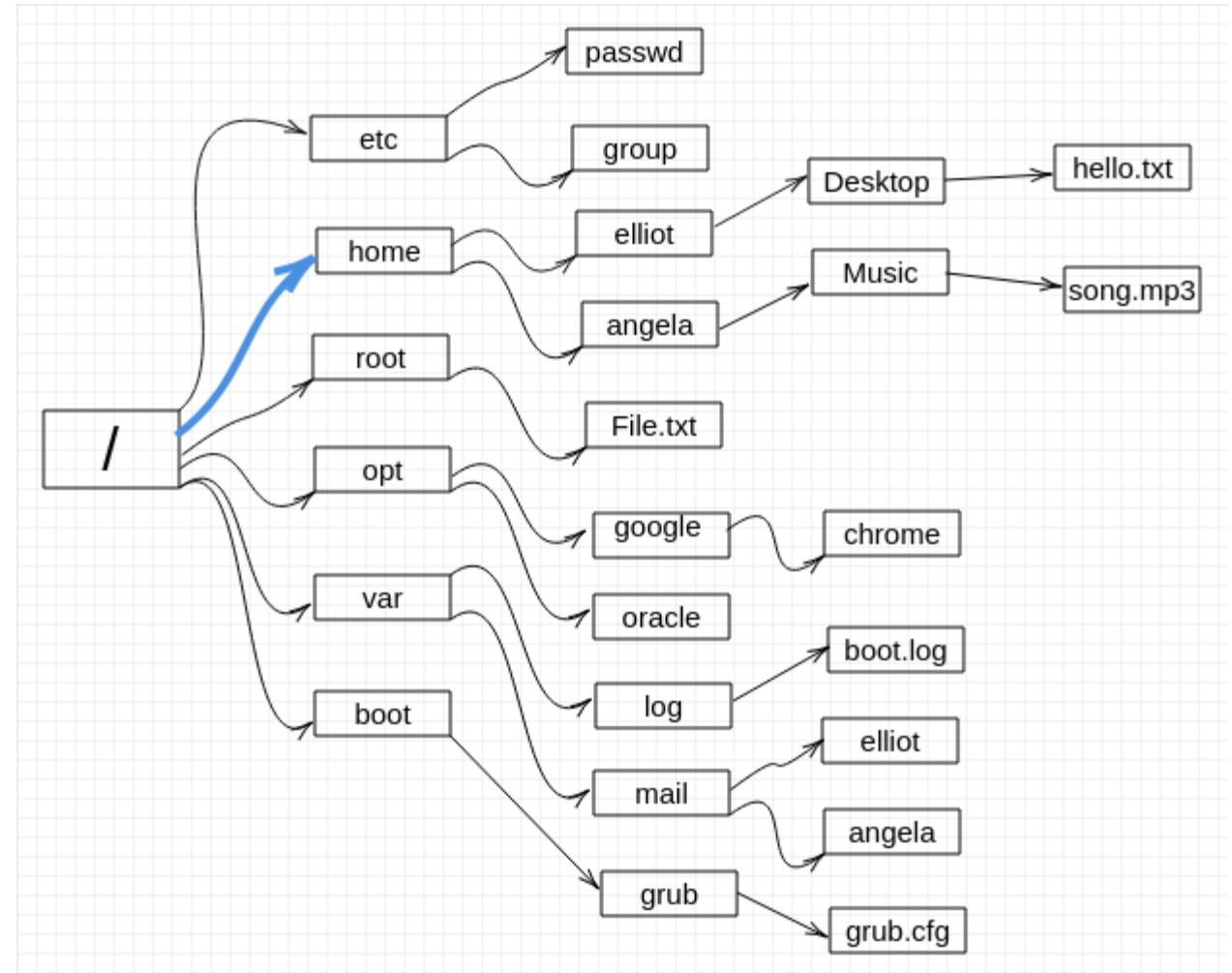
Navigating through the directory tree

- Alright, let's do more climbing.
- For example, let's climb to the /home directory to see how many users we have on the system.
- You can do that by simply running the cd /home command:

```
elliot@ubuntu-linux:~$ cd /home  
elliot@ubuntu-linux:/home$
```

Navigating through the directory tree

- Notice how your command prompt changes as it's now showing that you are at the home directory.



Navigating through the directory tree

- Now let's run ls to view the contents of the /home directory:

```
elliot@ubuntu-linux:/home$ ls  
angela elliot
```

- If you want proof that you are currently at the /home directory, you can run the pwd command:

```
elliot@ubuntu-linux:/home$ pwd  
/home
```

Navigating through the directory tree

- Sure enough! We are at the /home directory.
- Now let's climb to the home directory of user elliot.
- Now, believe it or not, there are two ways to navigate to elliot's home directory, You can simply run the cd elliot command:

```
elliot@ubuntu-linux:/home$ cd elliot
```

```
elliot@ubuntu-linux:~$ pwd
```

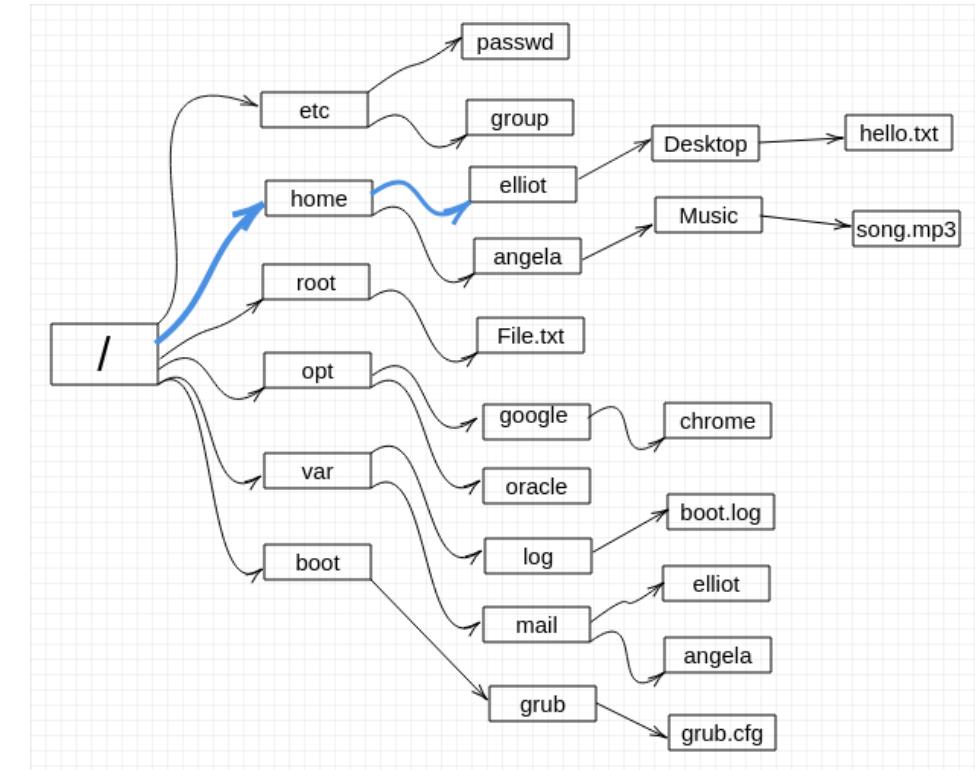
```
/home/elliot
```

Navigating through the directory tree

- Or you can run the cd /home/elliot command:

```
elliot@ubuntu-linux:/home$ cd /home/elliot
```

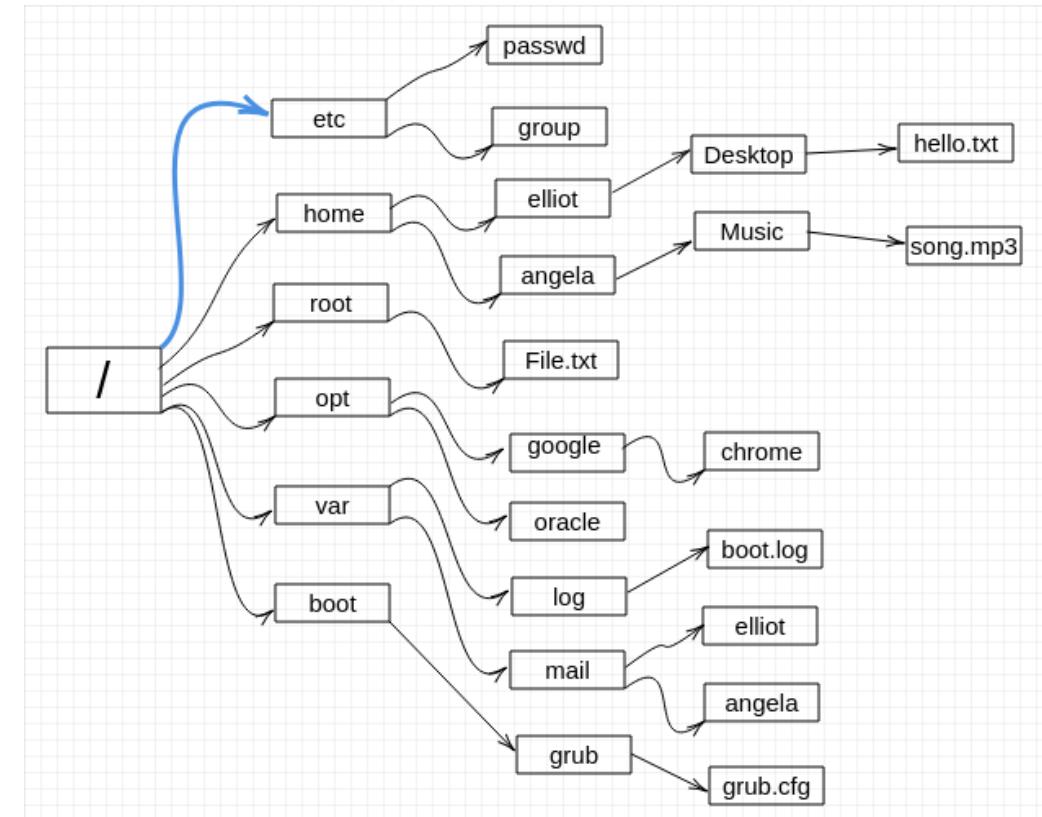
```
elliot@ubuntu-linux:~$ pwd  
/home/elliot
```



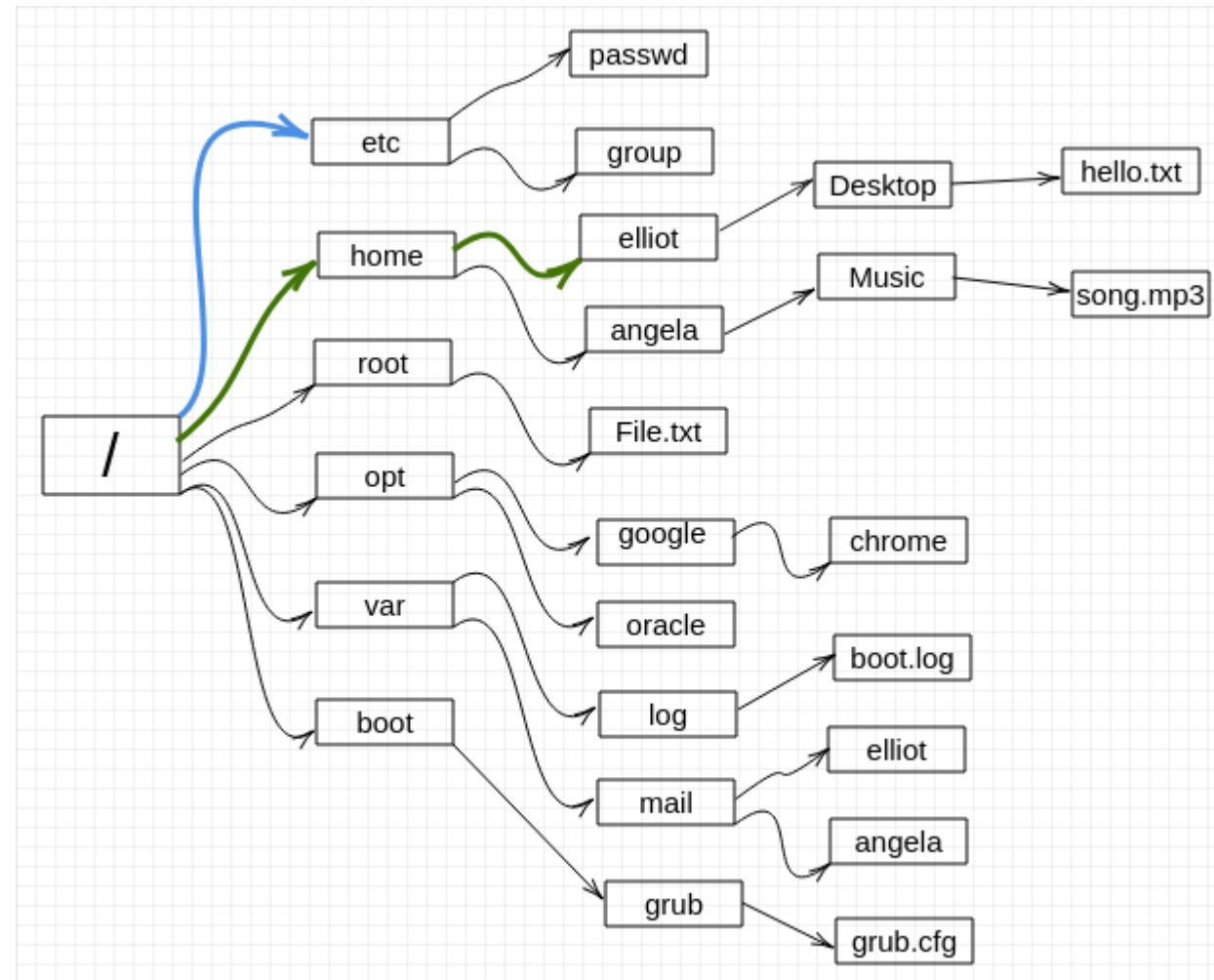
Navigating through the directory tree

- In other situations, we would be forced to use the full path (absolute path) /home/elliot to reach our destination.
- To demonstrate, let's first change to the /etc directory:

```
elliott@ubuntu-linux:~$ cd /etc  
elliott@ubuntu-linux:/etc$ pwd  
/etc
```



Navigating through the directory tree



Navigating through the directory tree

- To get to elliot's home directory, we can no longer use a short path (relative path) by running the cd elliot command:

```
elliot@ubuntu-linux:/etc$ cd elliot
```

```
bash: cd: elliot: No such file or directory
```

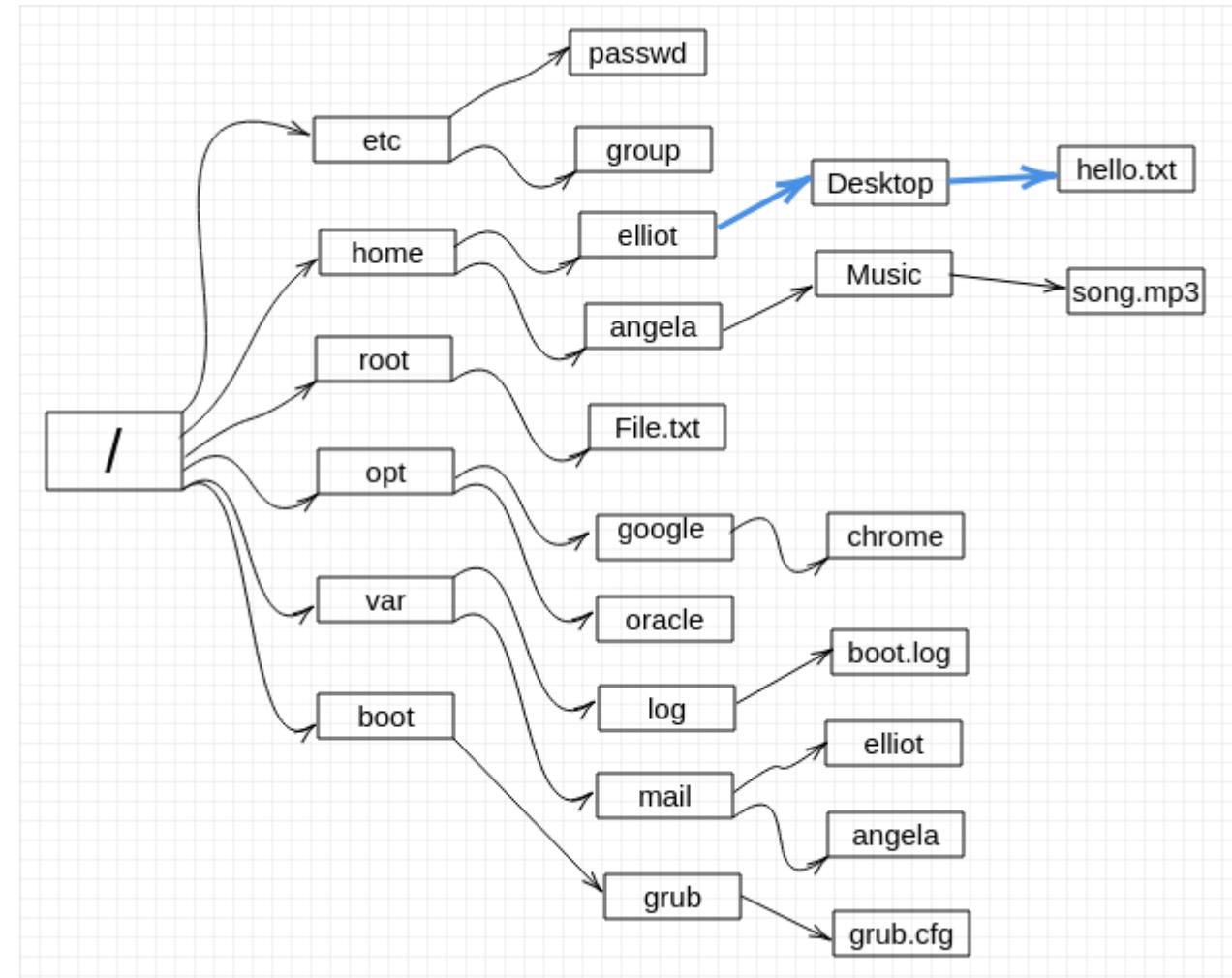
- As you can see, the Shell got mad and returned an error bash: cd: elliot: No such file or directory.
- In this case, we have to use the full path (absolute path)/home/elliot:

```
elliot@ubuntu-linux:/etc$ cd /home/elliot
```

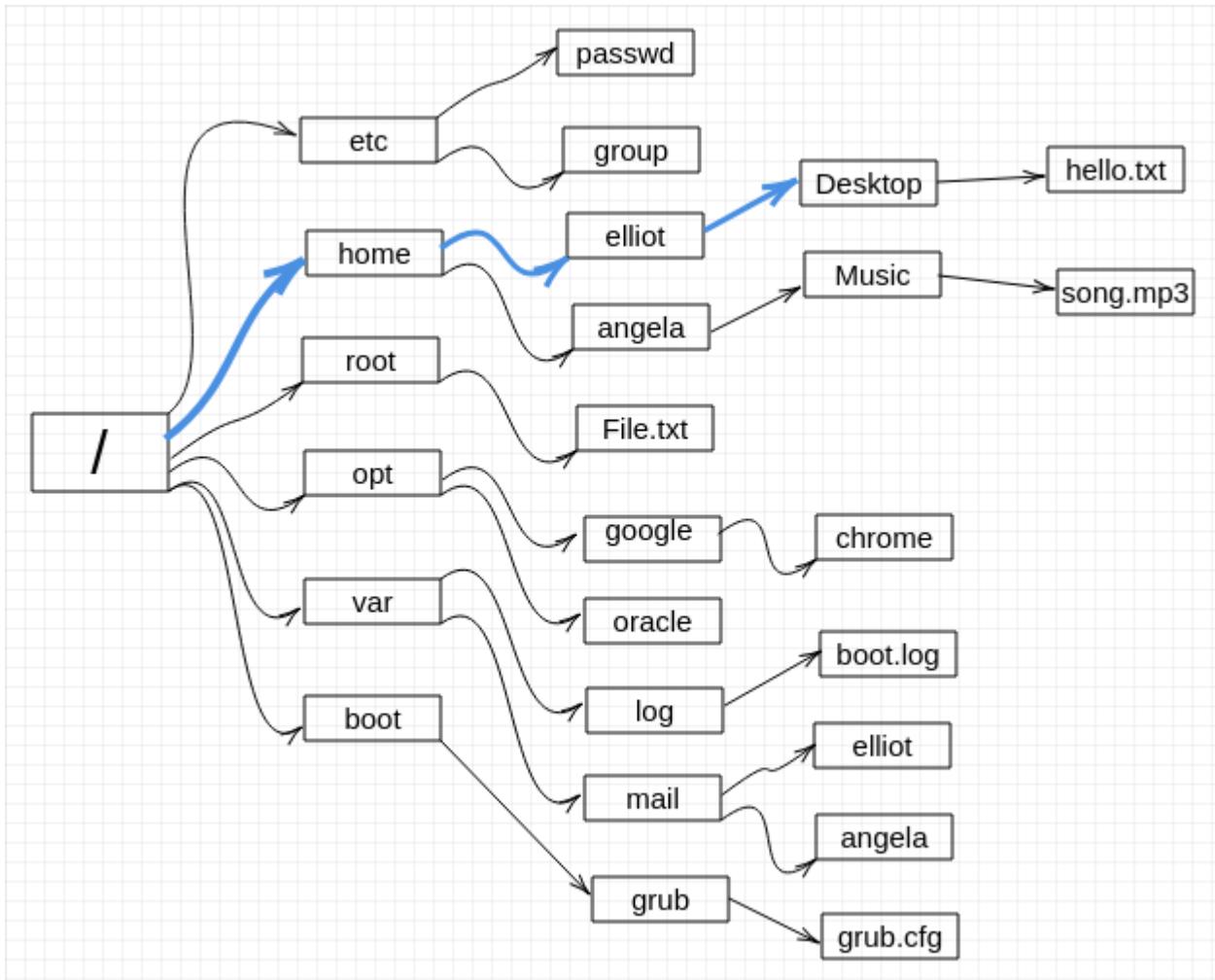
```
elliot@ubuntu-linux:~$ pwd  
/home/elliot
```

Navigating through the directory tree

- The following diagram shows you the relative path Desktop/hello.txt and will only work if your current working directory is /home/elliot.



Navigating through the directory tree



- The following image shows you the absolute path /home/elliot/Desktop and will always work regardless of your current working directory.

Navigating through the directory tree

- Now let's climb to Elliot's Desktop directory to see what he has there.
- We will use an absolute path:

```
elliot@ubuntu-linux:$ cd /home/elliott/Desktop
```

```
elliot@ubuntu-linux:~/Desktop$ pwd
```

```
/home/elliott/Desktop
```

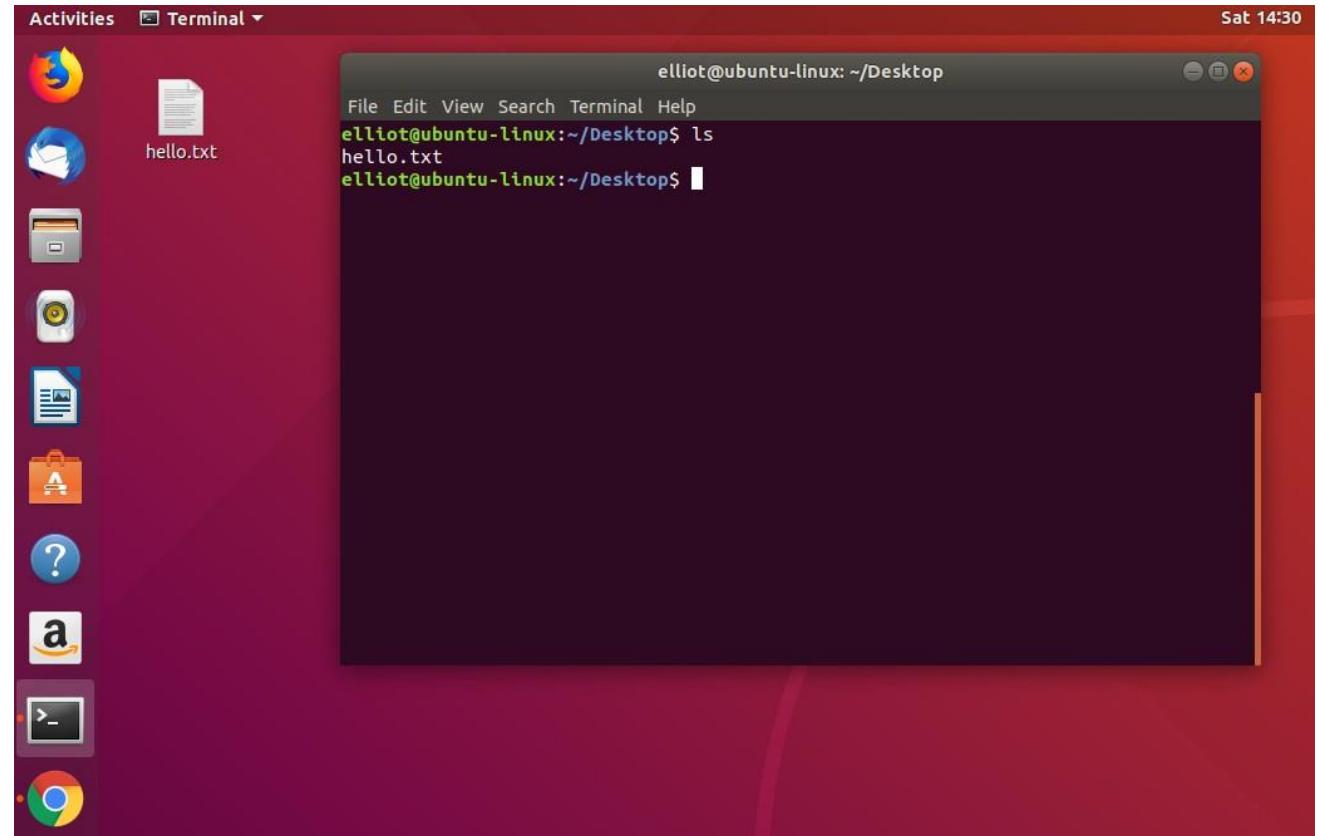
- We follow it with a pwd to confirm that we are indeed in the desired directory. Now let's run ls to view the contents of Elliot's desktop:

```
elliot@ubuntu-linux:~/Desktop$ ls
```

```
hello.txt
```

Navigating through the directory tree

- Notice that the file hello.txt is on Elliot's desktop, so we can actually see it right there on the desktop.



Navigating through the directory tree

- As you can see in the preceding image, there is a file named hello.txt on Elliot's desktop.
- You can use the cat command to view the contents of a text file:

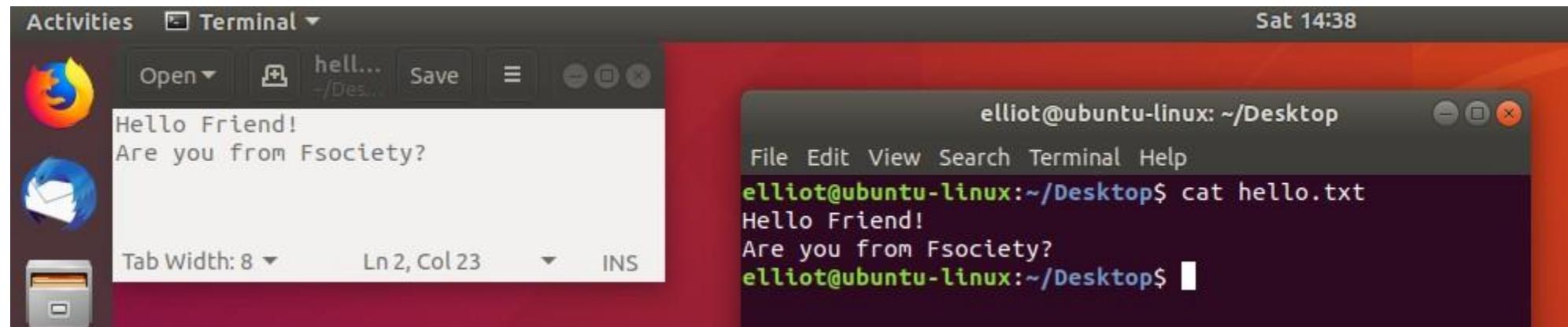
```
elliot@ubuntu-linux:~/Desktop$ cat hello.txt
```

Hello Friend!

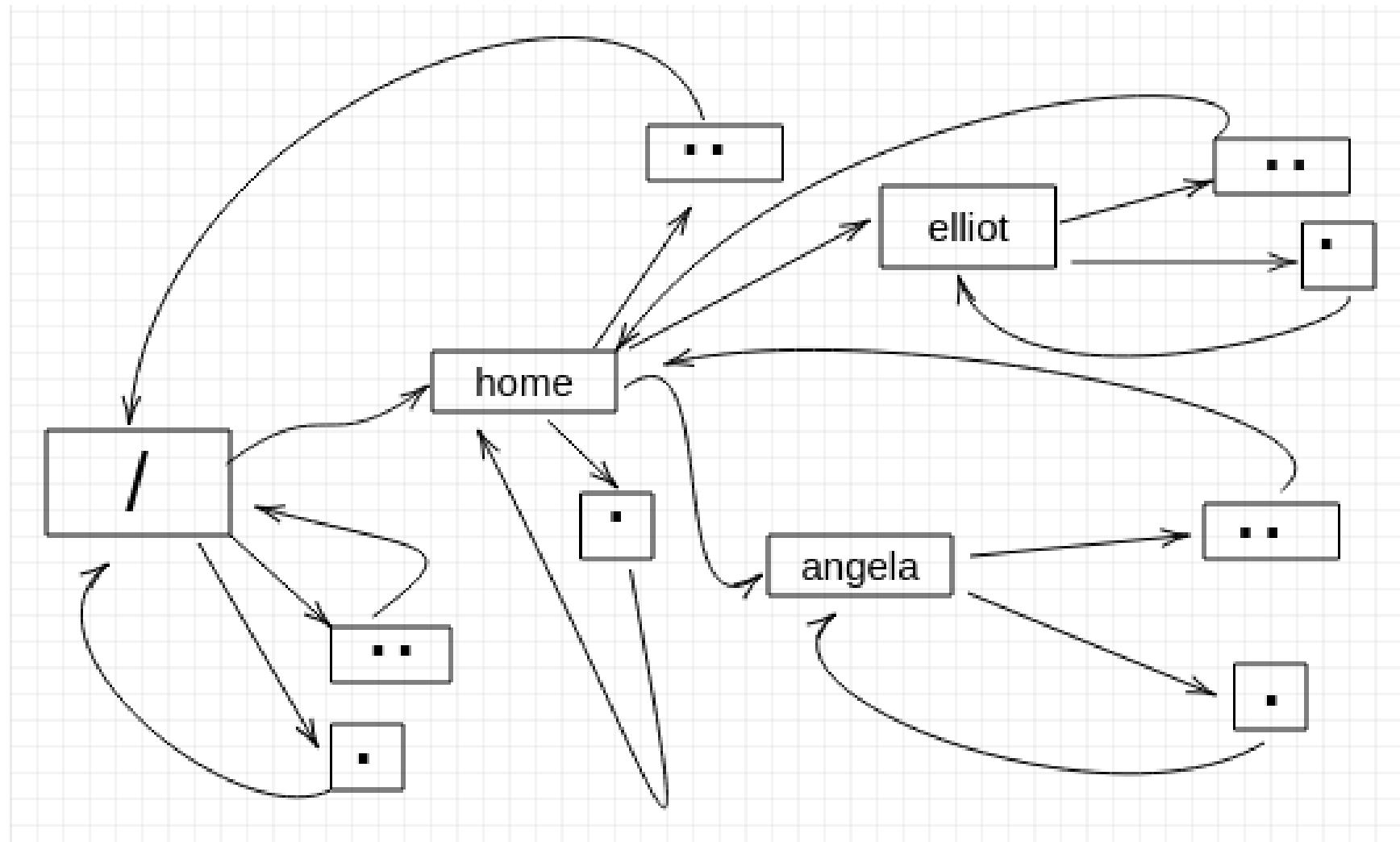
Are you from fsociety?

Navigating through the directory tree

- If you open the file hello.txt on the desktop, you will see the same contents, of course, as you can see in the following screenshot.



Parent and current directories



Parent and current directories

- It's easy to understand both directories by going through a few examples.

To demonstrate, let's first change to /home/elliot so that it becomes our current working directory:

```
elliot@ubuntu-linux:~/Desktop$ cd /home/elliot
```

```
elliot@ubuntu-linux:~$ pwd  
/home/elliot
```

- Now run the cd . command:

```
elliot@ubuntu-linux:~$ cd .  
elliot@ubuntu-linux:~$ pwd  
/home/elliot
```

Parent and current directories

- Now run the cd .. command:

```
elliott@ubuntu-linux:~$ cd ..  
elliott@ubuntu-linux:/home$ pwd  
/home
```

- We moved back one directory! In other words, we changed to the parent directory of /home/elliott, which is /home.
- Let's run another cd ..:

```
elliott@ubuntu-linux:/home$ cd ..  
elliott@ubuntu-linux:$ pwd  
/
```

Parent and current directories

- Indeed we keep going back, and now we are at the root of our directory tree.
- Well, let's run cd .. one more time:

```
elliot@ubuntu-linux:$ cd ..
```

```
elliot@ubuntu-linux:$ pwd
```

```
/
```

Parent and current directories

- You can also insert the directory separator cd ../../ to move back two directories at once:

```
elliott@ubuntu-linux:~$ pwd  
/home/elliott  
elliott@ubuntu-linux:~$ cd ../../  
elliott@ubuntu-linux:$ pwd  
/
```

Go back home!

- Let's change to the /var/log directory:

```
elliott@ubuntu-linux:~$ cd /var/log  
elliott@ubuntu-linux:/var/log$ pwd  
/var/log
```

- You can now run the cd ~ command to go to your home directory:

```
elliott@ubuntu-linux:/var/log$ cd ~  
elliott@ubuntu-linux:~$ pwd  
/home/elliott
```

Go back home!

- WOW! Let's do it again, but this time, we switch to user angela.
- In case you don't know, the character is called tilde and should be located next to your number 1 key on your keyboard:

```
elliott@ubuntu-linux:~$ whoami
```

```
elliott
```

```
elliott@ubuntu-linux:~$ su angela
```

```
Password:
```

```
angela@ubuntu-linux:/home/elliott$ whoami
```

```
angela
```

Go back home!

- Now, as user angela, I will navigate to the /var/log directory:

```
angela@ubuntu-linux:/home/elliot$ cd /var/log
```

```
angela@ubuntu-linux:/var/log$ pwd
```

```
/var/log
```

- Then I run the cd ~ command:

```
angela@ubuntu-linux:/var/log$ cd ~
```

```
angela@ubuntu-linux:~$ pwd
```

```
/home/angela
```

Take me back!

- Now, what if angela wants to go back as quickly as possible to her previous working directory?
- Running the cd - command is the fastest method that will land angela back to her previous working directory:

```
angela@ubuntu-linux:~$ pwd
```

```
/home/angela
```

```
angela@ubuntu-linux:~$ cd -
```

```
/var/log
```

Hidden Files

- The current directory . and the parent directory .. exist under each directory in the Linux filesystem.
- But how come we can't see them when we run the ls command?

```
elliot@ubuntu-linux:~/Desktop$ pwd  
/home/elliot/Desktop  
elliot@ubuntu-linux:~/Desktop$ ls  
hello.txt  
elliot@ubuntu-linux:~/Desktop$ ls -l  
total 4  
-rw-r--r-- 1 elliot elliot 37 Jan 19 14:20 hello.txt
```

Hidden Files

- You need to use the -a option with the ls command as follows:

```
elliott@ubuntu-linux:~/Desktop$ ls -a  
... hello.txt
```

- Hooray! Now you can see all the files.
- The -a option shows you all the files, including hidden files and of course you can use the full option name --all, which will do the same thing:

```
elliott@ubuntu-linux:~/Desktop$ ls --all  
... hello.txt
```

Hidden Files

- To demonstrate further, go to your user home directory and run the ls command:

```
angela@ubuntu-linux:~$ ls  
Music
```

- Now run the ls -a command:

```
angela@ubuntu-linux:~$ ls -a  
... .bash_logout .bashrc Music .profile
```

Passing command arguments

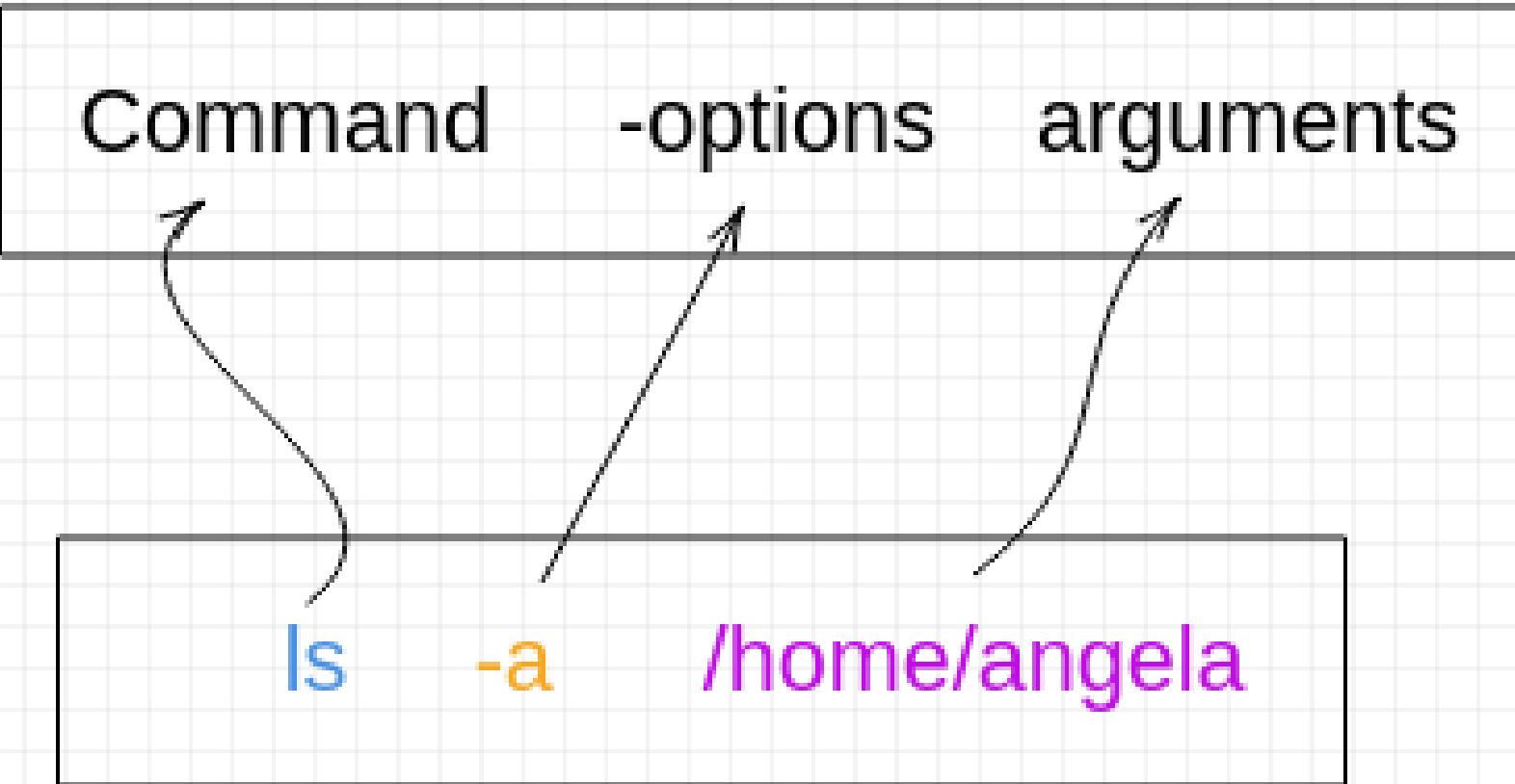
- For example, if your current working directory is /home/elliot:

```
elliot@ubuntu-linux:~$ pwd  
/home/elliot
```

- You can list all the files in /home/angela by running the ls -a /home/angela command:

```
elliot@ubuntu-linux:~$ ls -a /home/angela  
... .bash_history .bash_logout .bashrc Music .profile  
elliot@ubuntu-linux:~$ pwd  
/home/elliot  
elliot@ubuntu
```

Passing command arguments



Passing command arguments

- For example, we can do a long listing for all the files in /home/angela by running the ls -a -l /home/angela command:

```
elliott@ubuntu-linux:~$ ls -a -l /home/angela
total 28
drwxr-xr-x 3 angela angela 4096 Jan 20 13:43 .
drwxr-xr-x 9 root   root   4096 Jan 17 04:37 ..
-rw----- 1 angela angela  90 Jan 20 13:43 .bash_history
-rw-r--r-- 1 angela angela 220 Apr  4 2018 .bash_logout
-rw-r--r-- 1 angela angela 3771 Apr  4 2018 .bashrc
drwxrwxr-x 2 angela angela 4096 Jan 19 19:42 Music
-rw-r--r-- 1 angela angela  807 Apr  4 2018 .profile
```

Passing command arguments

- Notice that the ordering of the options doesn't matter here, so if you run the ls -l -a /home/angela command:

```
elliott@ubuntu-linux:~$ ls -l -a /home/angela
total 28
drwxr-xr-x 3 angela angela 4096 Jan 20 13:43 .
drwxr-xr-x 9 root root 4096 Jan 17 04:37 ..
-rw----- 1 angela angela 90 Jan 20 13:43 .bash_history
-rw-r--r-- 1 angela angela 220 Apr  4 2018 .bash_logout
-rw-r--r-- 1 angela angela 3771 Apr  4 2018 .bashrc
drwxrwxr-x 2 angela angela 4096 Jan 19 19:42 Music
-rw-r--r-- 1 angela angela 807 Apr  4 2018 .profile
```

Passing command arguments

```
elliott@ubuntu-linux:~$ ls -l -a /home/angela /home/elliott  
/home/angela:  
  
total 28  
drwxr-xr-x 3 angela angela 4096 Jan 20 13:43 .  
drwxr-xr-x 9 root root 4096 Jan 17 04:37 ..  
-rw------- 1 angela angela 90 Jan 20 13:43 .bash_history  
-rw-r--r-- 1 angela angela 220 Apr 4 2018 .bash_logout  
-rw-r--r-- 1 angela angela 3771 Apr 4 2018 .bashrc  
drwxrwxr-x 2 angela angela 4096 Jan 19 19:42 Music  
-rw-r--r-- 1 angela angela 807 Apr 4 2018 .profile  
  
/home/elliott:  
total 28  
drwxr-xr-x 3 elliot elliot 4096 Jan 20 16:26 .  
drwxr-xr-x 9 root root 4096 Jan 17 04:37 ..  
-rw------- 1 elliot elliot 90 Jan 20 13:43 .bash_history  
-rw-r--r-- 1 elliot elliot 220 Dec 26 23:47 .bash_logout  
-rw-r--r-- 1 elliot elliot 3771 Dec 26 23:47 .bashrc  
drwxr-xr-x 2 elliot elliot 4096 Jan 19 14:20 Desktop  
-rw-r--r-- 1 elliot elliot 807 Apr 4 2018 .profile
```

The touch command

- Let's do a long listing for all the files in /home/elliot one more time to discuss something very important:

```
elliot@ubuntu-linux:~$ ls -a -l /home/elliot
```

```
total 28
```

```
drwxr-xr-x 3 elliot elliot 4096 Jan 20 16:26 .
```

```
drwxr-xr-x 9 root root 4096 Jan 17 04:37 ..
```

```
-rw----- 1 elliot elliot 90 Jan 20 13:43 .bash_history
```

```
-rw-r--r-- 1 elliot elliot 220 Dec 26 23:47 .bash_logout
```

```
-rw-r--r-- 1 elliot elliot 3771 Dec 26 23:47 .bashrc
```

```
drwxr-xr-x 2 elliot elliot 4096 Jan 19 14:20 Desktop
```

```
-rw-r--r-- 1 elliot elliot 807 Apr 4 2018 .profile
```

The touch command

- To demonstrate, let's first get the modification time on elliot's Desktop directory, you can do that by running the ls -l -d /home/elliot/Desktop command:

```
elliot@ubuntu-linux:~$ ls -l -d /home/elliot/Desktop  
drwxr-xr-x 2 elliot elliot 4096 Jan 19 14:20 /home/elliot/Desktop
```

The touch command

- Now if you run the touch /home/elliot/Desktop command:

```
elliot@ubuntu-linux:~$ touch /home/elliot/Desktop  
elliot@ubuntu-linux:~$ ls -l -d /home/elliot/Desktop  
drwxr-xr-x 2 elliot elliot 4096 Jan 20 19:42 /home/elliot/Desktop  
elliot@ubuntu-linux:~$ date  
Sun Jan 20 19:42:08 CST 2020
```

The touch command

- What if we try to update the modification time of a file that doesn't exist? What will happen?
- The only way to know is to try it.
- Notice that user elliot has only one visible (not hidden) file in his home directory, which happens to be the Desktop directory:

```
elliot@ubuntu-linux:~$ pwd  
/home/elliot  
elliot@ubuntu-linux:~$ ls -l  
total 4  
drwxr-xr-x 2 elliot elliot 4096 Jan 20 19:42 Desktop
```

The touch command

- Now watch what will happen when user elliot runs the touch blabla command:

```
elliot@ubuntu-linux:~$ touch blabla
elliot@ubuntu-linux:~$ ls -l
total 4
-rw-r--r-- 1 elliot elliot 0 Jan 20 20:00 blabla
drwxr-xr-x 2 elliot elliot 4096 Jan 20 19:42 Desktop
```

The touch command

- By default, the touch command changes both the modification and access times of a file. I have created three files in elliot's home directory: file1, file2, and file3:

```
elliot@ubuntu-linux:~$ ls -l
total 8
drwxr-xr-x 6 elliot elliot 4096 Jan 25 22:13 Desktop
drwxr-xr-x 3 elliot elliot 4096 Jan 25 22:18 dir1
-rw-r--r-- 1 elliot elliot 0 Feb 29 2004 file1
-rw-r--r-- 1 elliot elliot 0 Apr 11 2010 file2
-rw-r--r-- 1 elliot elliot 0 Oct  3 1998 file3
```

The touch command

- Now to change only the modification time of file1.
- We pass the -m option to the touch command:

```
elliot@ubuntu-linux:~$ touch -m file1
elliot@ubuntu-linux:~$ ls -l
total 8
drwxr-xr-x 6 elliot elliot 4096 Jan 25 22:13 Desktop
drwxr-xr-x 3 elliot elliot 4096 Jan 25 22:18 dir1
-rw-r--r-- 1 elliot elliot 0 Jan 25 23:08 file1
-rw-r--r-- 1 elliot elliot 0 Apr 11 2010 file2
-rw-r--r-- 1 elliot elliot 0 Oct  3 1998 file3
elliot@ubuntu-linux:~$
```

The touch command

```
elliott@ubuntu-linux:~$ ls -l
total 8
drwxr-xr-x 6 elliot elliot 4096 Jan 25 22:13 Desktop
drwxr-xr-x 3 elliot elliot 4096 Jan 25 22:18 dir1
-rw-r--r-- 1 elliot elliot 0    Jan 25 23:08 file1
-rw-r--r-- 1 elliot elliot 0    Apr 11  2010 file2
-rw-r--r-- 1 elliot elliot 0    Oct 3   1998 file3
elliott@ubuntu-linux:~$ ls -l -u
total 8
drwxr-xr-x 6 elliot elliot 4096 Jan 25 22:13 Desktop
drwxr-xr-x 3 elliot elliot 4096 Jan 25 22:18 dir1
-rw-r--r-- 1 elliot elliot 0    Feb 29 2004  file1
-rw-r--r-- 1 elliot elliot 0    Apr 11 2010  file2
-rw-r--r-- 1 elliot elliot 0    Oct 3  1998  file3
```

The touch command

- To do this, we pass the -a option to the touch command:

```
elliott@ubuntu-linux:~$ touch -a file2
elliott@ubuntu-linux:~$ ls -l
total 8
drwxr-xr-x 6 elliot elliot 4096 Jan 25 22:13 Desktop
drwxr-xr-x 3 elliot elliot 4096 Jan 25 22:18 dir1
-rw-r--r-- 1 elliot elliot 0 Jan 25 23:08 file1
-rw-r--r-- 1 elliot elliot 0 Apr 11 2010 file2
-rw-r--r-- 1 elliot elliot 0 Oct 3 1998 file3
elliott@ubuntu-linux:~$ ls -l -u
total 8
drwxr-xr-x 6 elliot elliot 4096 Jan 25 22:13 Desktop
drwxr-xr-x 3 elliot elliot 4096 Jan 25 22:18 dir1
-rw-r--r-- 1 elliot elliot 0 Feb 29 2004 file1
-rw-r--r-- 1 elliot elliot 0 Jan 25 23:20 file2
-rw-r--r-- 1 elliot elliot 0 Oct 3 1998 file3
elliott@ubuntu-linux:~$
```

The touch command

```
elliott@ubuntu-linux:~$ touch -a file2
elliott@ubuntu-linux:~$ ls -l
total 8
drwxr-xr-x 6 elliot elliot 4096 Jan 25 22:13 Desktop
drwxr-xr-x 3 elliot elliot 4096 Jan 25 22:18 dir1
-rw-r--r-- 1 elliot elliot     0 Jan 25 23:08 file1
-rw-r--r-- 1 elliot elliot     0 Apr 11  2010 file2
-rw-r--r-- 1 elliot elliot     0 Oct  3  1998 file3
elliott@ubuntu-linux:~$ ls -l -u
total 8
drwxr-xr-x 6 elliot elliot 4096 Jan 25 22:13 Desktop
drwxr-xr-x 3 elliot elliot 4096 Jan 25 22:18 dir1
-rw-r--r-- 1 elliot elliot     0 Feb 29  2004 file1
-rw-r--r-- 1 elliot elliot     0 Jan 25 23:20 file2
-rw-r--r-- 1 elliot elliot     0 Oct  3  1998 file3
elliott@ubuntu-linux:~$
```

The touch command

- Now to change both the modification and access times of file3, you can run the touch command with no options:

```
elliott@ubuntu-linux:~$ ls -l file3  
-rw-r--r-- 1 elliot elliot 0 Oct 3 1998 file3  
elliott@ubuntu-linux:~$ touch file3  
elliott@ubuntu-linux:~$ ls -l file3  
-rw-r--r-- 1 elliot elliot 0 Jan 25 23:27 file3  
elliott@ubuntu-linux:~$ ls -l -u file3  
-rw-r--r-- 1 elliot elliot 0 Jan 25 23:27 file3
```

The touch command

- Awesome! You can also pass the -t option to the ls command to list the files sorted by modification times, newest first:

```
elliott@ubuntu-linux:~$ ls -l -t
total 8
-rw-r--r-- 1 elliot elliot 0 Jan 25 23:27 file3
-rw-r--r-- 1 elliot elliot 0 Jan 25 23:08 file1
drwxr-xr-x 3 elliot elliot 4096 Jan 25 22:18 dir1
drwxr-xr-x 6 elliot elliot 4096 Jan 25 22:13 Desktop
-rw-r--r-- 1 elliot elliot 0 Apr 11 2010 file2
```

The touch command

- You can add the -u option to sort by access times instead:

```
elliott@ubuntu-linux:~$ ls -l -t -u
total 8
-rw-r--r-- 1 elliot elliot 0 Jan 25 23:27 file3
-rw-r--r-- 1 elliot elliot 0 Jan 25 23:20 file2
-rw-r--r-- 1 elliot elliot 0 Jan 25 23:20 file1
drwxr-xr-x 3 elliot elliot 4096 Jan 25 22:18 dir1
drwxr-xr-x 6 elliot elliot 4096 Jan 25 22:13 Desktop
```

The touch command

- You can also pass the -r option to reverse the sorting:

```
elliott@ubuntu-linux:~$ ls -l -t -r
total 8
-rw-r--r-- 1 elliot elliot 0 Apr 11 2010 file2
drwxr-xr-x 6 elliot elliot 4096 Jan 25 22:13 Desktop
drwxr-xr-x 3 elliot elliot 4096 Jan 25 22:18 dir1
-rw-r--r-- 1 elliot elliot 0 Jan 25 23:08 file1
-rw-r--r-- 1 elliot elliot 0 Jan 25 23:27 file3
```

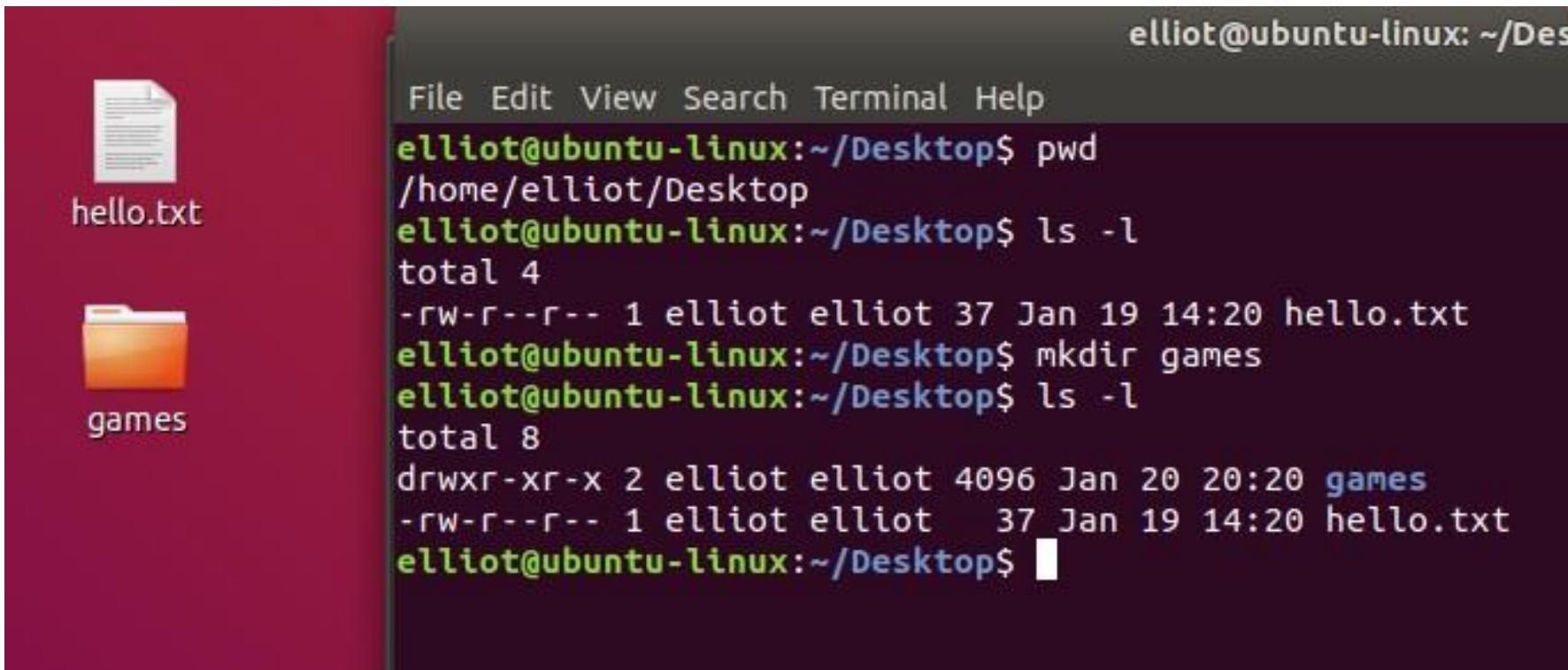
Making directories

- To create directories in Linux, we use the `mkdir` command, which is short for make directory.
- In elliot's desktop, let's create a directory named `games` by running the `mkdir games` command:

```
elliot@ubuntu-linux:~/Desktop$ mkdir games
elliot@ubuntu-linux:~/Desktop$ ls -l
total 8
drwxr-xr-x 2 elliot elliot 4096 Jan 20 20:20 games
-rw-r--r-- 1 elliot elliot 37 Jan 19 14:20 hello.txt
elliot@ubuntu-linux:~/Desktop$
```

Making directories

- Notice that my current working directory is /home/elliot/Desktop; that's why I was able to use a relative path.



The image shows a dual-pane interface. On the left is a file manager window with a dark red background, displaying two items: a white document icon labeled "hello.txt" and an orange folder icon labeled "games". On the right is a terminal window with a dark background and light-colored text. The terminal session starts with the user's name and location: "elliot@ubuntu-linux: ~/Desktop". The user runs the "pwd" command to show the current directory: "/home/elliot/Desktop". Then, they run "ls -l" to list the contents of the directory, which shows "hello.txt" and the newly created "games" folder. Finally, the user runs "mkdir games" to create the "games" directory, and then "ls -l" again to verify its creation, showing it as a folder named "games". The terminal ends with the user's name and location again: "elliot@ubuntu-linux: ~/Desktop\$".

```
elliot@ubuntu-linux: ~/Desktop$ pwd  
/home/elliot/Desktop  
elliot@ubuntu-linux: ~/Desktop$ ls -l  
total 4  
-rw-r--r-- 1 elliot elliot 37 Jan 19 14:20 hello.txt  
elliot@ubuntu-linux: ~/Desktop$ mkdir games  
elliot@ubuntu-linux: ~/Desktop$ ls -l  
total 8  
drwxr-xr-x 2 elliot elliot 4096 Jan 20 20:20 games  
-rw-r--r-- 1 elliot elliot 37 Jan 19 14:20 hello.txt  
elliot@ubuntu-linux: ~/Desktop$
```

Making directories

- You can also create multiple directories at the same time.
- For example, you can create three directories – Music, Movies, and Books – on your desktop by running the `mkdir Music Movies Books` command:

```
elliot@ubuntu-linux:~/Desktop$ mkdir Music Movies Books
```

```
elliot@ubuntu-linux:~/Desktop$ ls -l
```

```
total 20
```

```
drwxr-xr-x 2 elliot elliot 4096 Jan 21 01:54 Books
```

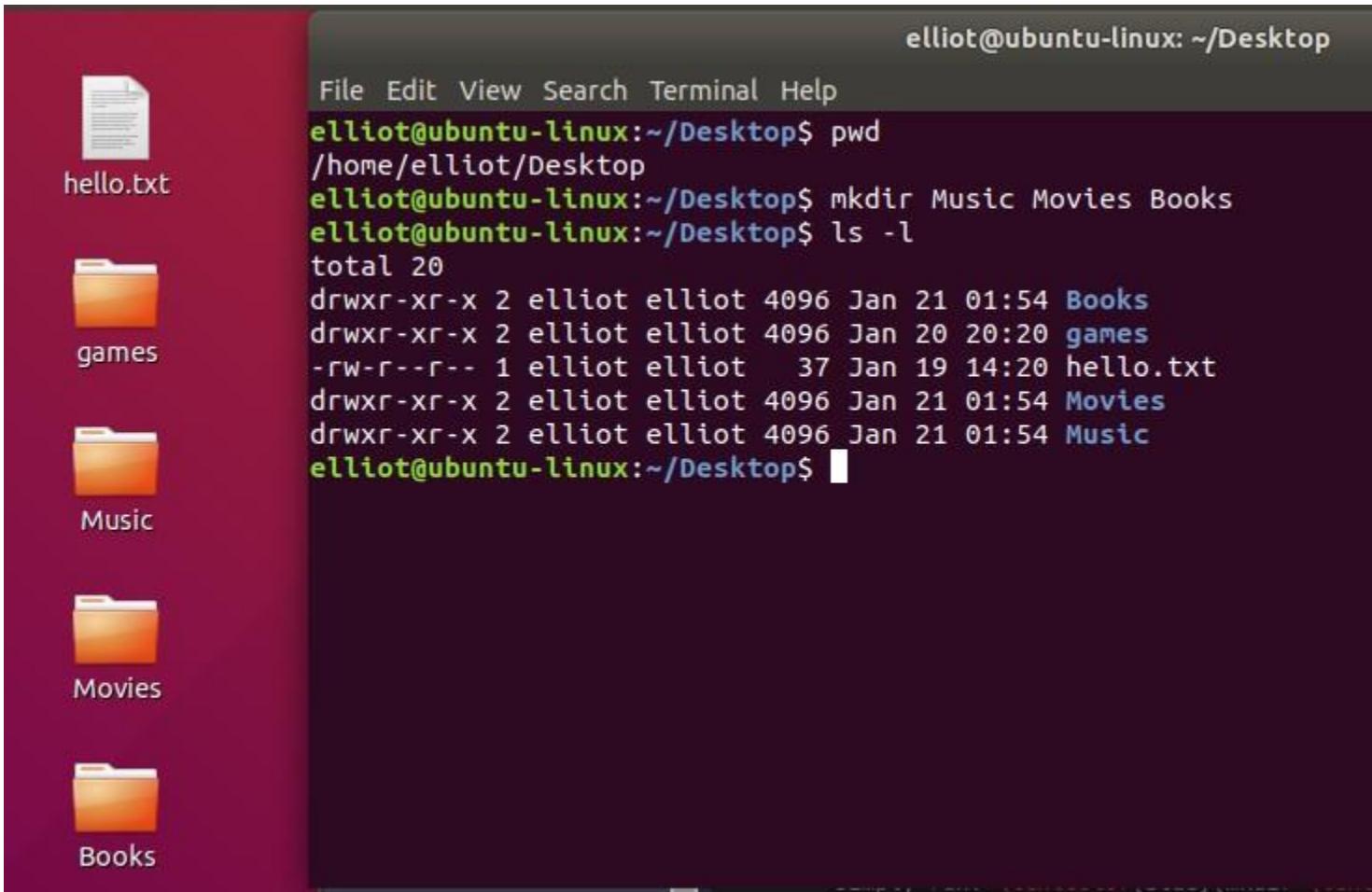
```
drwxr-xr-x 2 elliot elliot 4096 Jan 20 20:20 games
```

```
-rw-r--r-- 1 elliot elliot 37 Jan 19 14:20 hello.txt
```

```
drwxr-xr-x 2 elliot elliot 4096 Jan 21 01:54 Movies
```

```
drwxr-xr-x 2 elliot elliot 4096 Jan 21 01:54 Music
```

Making directories



The image shows a dual-pane interface. On the left is a file manager window with a dark red background, displaying a file named "hello.txt" and five empty folder icons labeled "games", "Music", "Movies", and "Books". On the right is a terminal window with a dark background, showing a command-line session:

```
elliott@ubuntu-linux: ~/Desktop
File Edit View Search Terminal Help
elliott@ubuntu-linux:~/Desktop$ pwd
/home/elliott/Desktop
elliott@ubuntu-linux:~/Desktop$ mkdir Music Movies Books
elliott@ubuntu-linux:~/Desktop$ ls -l
total 20
drwxr-xr-x 2 elliot elliot 4096 Jan 21 01:54 Books
drwxr-xr-x 2 elliot elliot 4096 Jan 20 20:20 games
-rw-r--r-- 1 elliot elliot 37 Jan 19 14:20 hello.txt
drwxr-xr-x 2 elliot elliot 4096 Jan 21 01:54 Movies
drwxr-xr-x 2 elliot elliot 4096 Jan 21 01:54 Music
elliott@ubuntu-linux:~/Desktop$ █
```

Making directories

```
elliott@ubuntu-linux:~$ pwd  
/home/elliott  
elliott@ubuntu-linux:~$ mkdir -p dir1/dir2/dir3  
elliott@ubuntu-linux:~$ ls  
blabla Desktop dir1  
elliott@ubuntu-linux:~$ cd dir1  
elliott@ubuntu-linux:~/dir1$ ls  
dir2  
elliott@ubuntu-linux:~/dir1$ cd dir2  
elliott@ubuntu-linux:~/dir1/dir2$ ls  
dir3  
elliott@ubuntu-linux:~/dir1/dir2$ cd dir3  
elliott@ubuntu-linux:~/dir1/dir2/dir3$ pwd  
/home/elliott/dir1/dir2/dir3  
elliott@ubuntu-linux:~/dir1/dir2/dir3$
```

Making directories

- You can use the recursive -R option to do a recursive listing on /home/elliot/dir1 and see all the files underneath /home/elliot/dir1 without the hassle of changing to each directory:

```
elliot@ubuntu-linux:~$ ls -R dir1
```

dir1:

dir2

dir1/dir2:

dir3

dir1/dir2/dir3:

```
elliot@ubuntu-linux:~$
```

Making directories

- You can also create a new directory with multiple subdirectories by including them inside a pair of curly brackets and each subdirectory separated by a comma like in the following:

```
elliot@ubuntu-linux:~/dir1/dir2/dir3$ mkdir -p dir4/{dir5,dir6,dir7}
```

```
elliot@ubuntu-linux:~/dir1/dir2/dir3$ ls -R dir4
```

dir4:

dir5 dir6 dir7

dir4/dir5:

dir4/dir6:

dir4/dir7:

Combining command options

ls option	What it does
-l	Long and detailed listing of files.
-a	List the hidden files.
-d	List directories themselves, not their contents.
-t	Sort files by modification times.
-u	When used with -l, it shows access times instead of modification times. When used with -lt, it will sort by, and show, access times.
-r	Will reverse listing order.
-R	List subdirectories recursively.

Combining command options

```
elliott@ubuntu-linux:~$ ls -latr
```

```
total 120
```

```
-rw-r--r-- 1 elliot elliot 0 Apr 11 2010 file2
-rw-r--r-- 1 elliot elliot 807 Dec 26 23:47 .profile
-rw-r--r-- 1 elliot elliot 3771 Dec 26 23:47 .bashrc
drwxr-xr-x 9 root root 4096 Jan 17 04:37 ..
-rw-r--r-- 1 elliot elliot 220 Jan 20 17:23 .bash_logout
drwxr-xr-x 6 elliot elliot 4096 Jan 25 22:13 Desktop
-rw-r--r-- 1 elliot elliot 0 Jan 25 23:08 file1
-rw-r--r-- 1 elliot elliot 0 Jan 25 23:27 file3
drwxr-xr-x 3 elliot elliot 4096 Jan 25 23:52 dir1
-rw----- 1 elliot elliot 3152 Jan 26 00:01 .bash_history
drwxr-xr-x 17 elliot elliot 4096 Jan 30 23:32 .
```

Combining command options

- Will yield the same result as running the ls -l -a -t -r command:

```
elliott@ubuntu-linux:~$ ls -l -a -t -r
total 120
-rw-r--r-- 1 elliot elliot 0 Apr 11 2010 file2
-rw-r--r-- 1 elliot elliot 807 Dec 26 23:47 .profile
-rw-r--r-- 1 elliot elliot 3771 Dec 26 23:47 .bashrc
drwxr-xr-x 9 root root 4096 Jan 17 04:37 ..
-rw-r--r-- 1 elliot elliot 220 Jan 20 17:23 .bash_logout
drwxr-xr-x 6 elliot elliot 4096 Jan 25 22:13 Desktop
-rw-r--r-- 1 elliot elliot 0 Jan 25 23:08 file1
-rw-r--r-- 1 elliot elliot 0 Jan 25 23:27 file3
drwxr-xr-x 3 elliot elliot 4096 Jan 25 23:52 dir1
-rw------- 1 elliot elliot 3152 Jan 26 00:01 .bash_history
drwxr-xr-x 17 elliot elliot 4096 Jan 30 23:32 .
```

Combining command options

- Before this lesson comes to an end, I want to show you a pretty cool tip.
- First, let's create a directory named `averylongdirectoryname`:

```
elliot@ubuntu-linux:~$ mkdir averylongdirectoryname
```

```
elliot@ubuntu-linux:~$ ls -ld averylongdirectoryname
```

```
drwxr-xr-x 2 elliot elliot 4096 Mar 2 12:57 averylongdirectoryname
```

- Tab Completion is one of the most useful features in the Linux command line. You can use this feature to let the shell automatically complete (suggest) command names and file paths.
- To demonstrate, type (don't run) the following text on your terminal:

```
elliot@ubuntu-linux:~$ cd ave
```

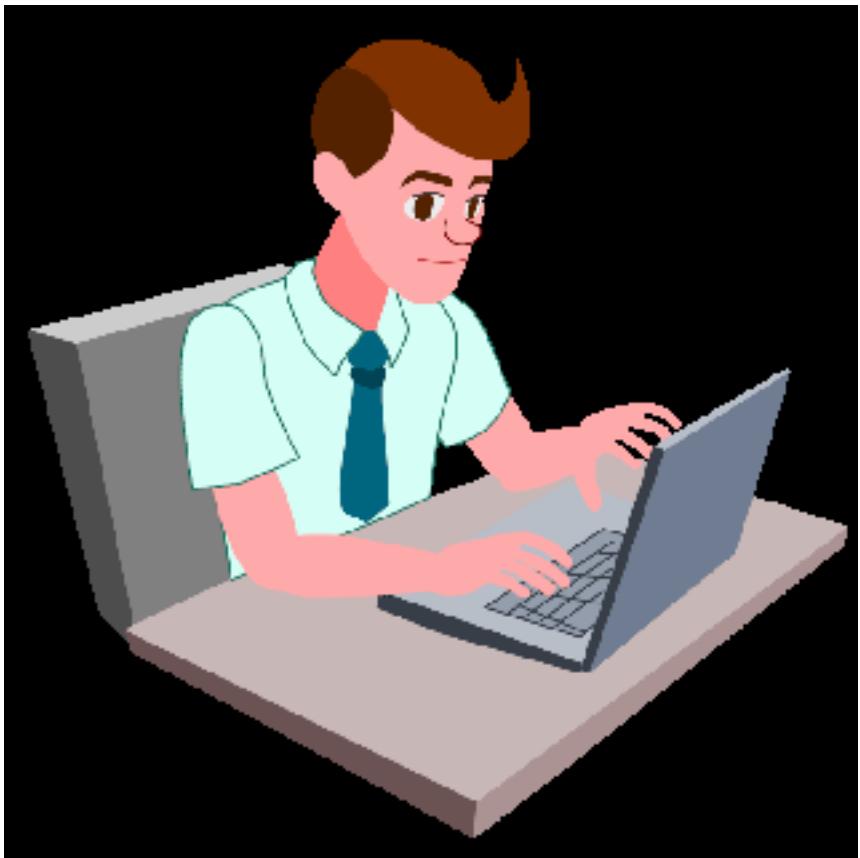
Combining command options

- Now press the Tab key on your keyboard, and the shell will automatically complete the directory name for you:

```
elliot@ubuntu-linux:~$ cd averylongdirectoryname/
```

Knowledge check

- For the following exercises, open up your terminal and try to solve the following tasks:
 1. Do a long listing for all the files in /var/log.
 2. Display the contents of the file /etc/hostname.
 3. Create three files – file1, file2, and file3 – in /home/elliot.
 4. List all the files (including hidden files) of elliot's home directory.
 5. Create a directory named fsociety in /home/elliot.



"Complete Lab"

3. Linux File Editors

Linux File Editors

- First of all, let me tell you something that may surprise you, Linux implements what is called "Everything is a file" philosophy.
- This means that on your Linux system, everything is represented by a file. For example, your hard disk is represented by a file.
- A running program (process) is represented by a file. Even your peripheral devices, such as your keyboard, mouse, and printer, are all represented by files.

Graphical editors – gedit and kate

- We start with the most basic and simple editors out there, These are the graphical editors!
- If you are using a GNOME version of any Linux distribution, then you will have the text editor gedit installed by default.
- On the other hand, if you are using a KDE version of Linux, then you will have the text editor kate installed by default.

Graphical editors – gedit and kate

- Anyways, there is really not a lot to discuss on graphical editors.
- They are pretty intuitive and easy to use.
- For example, if you want to view a text file with gedit, then you run the gedit command followed by any filename:
`elliott@ubuntu-linux:~$ gedit /proc/cpuinfo`



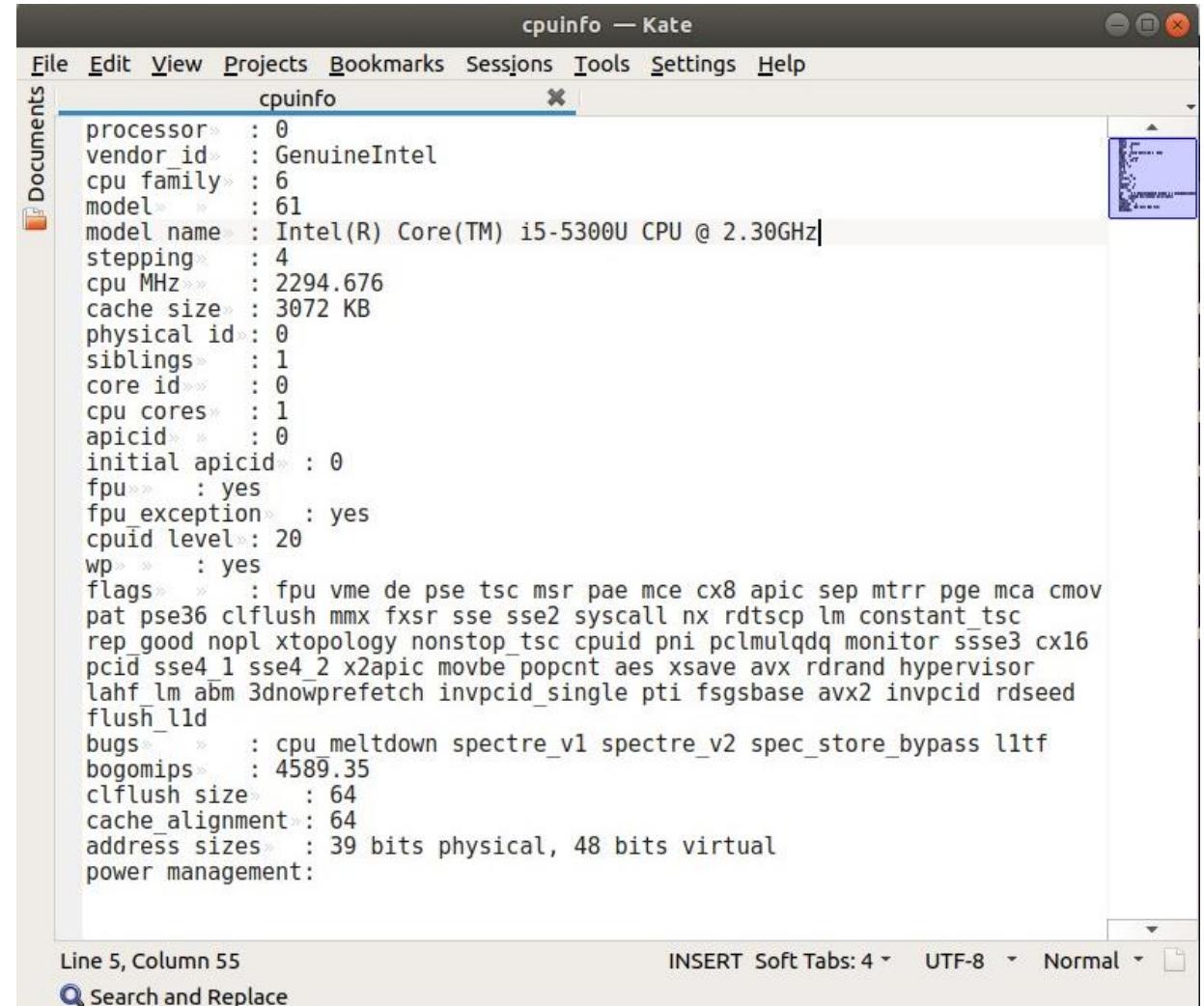
The screenshot shows the gedit graphical editor displaying the contents of the /proc/cpuinfo file. The window title is "cpuinfo [Read-Only] /proc". The text area contains detailed CPU specifications, including processor type (Intel(R) Core(TM) i5-5300U), frequency (2.30GHz), and various flags and features. The bottom status bar shows "Plain Text" and "Tab Width: 8".

```
processor : 0
vendor_id : GenuineIntel
cpu family : 6
model : 61
model name : Intel(R) Core(TM) i5-5300U CPU @ 2.30GHz
stepping : 4
cpu MHz : 2294.676
cache size : 3072 KB
physical id : 0
siblings : 1
core id : 0
cpu cores : 1
apicid : 0
initial apicid : 0
fpu : yes
fpu_exception : yes
cpuid level : 20
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush
mmx fxsr sse sse2 syscall nx rdtscp lm constant_tsc rep_good nopl xtopology nonstop_tsc cpuid pn
pclmulqdq monitor ssse3 cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt aes xsave avx rdrand
hypervisor lahf_lm abm 3dnowprefetch invpcid_single pti fsgsbase avx2 invpcid rdseed flush_l1d
bugs : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf
bogomips : 4589.35
clflush size : 64
cache_alignment : 64
address sizes : 39 bits physical, 48 bits virtual
power management:
```

Graphical editors – gedit and kate

- If you don't have gedit and have kate instead, then you can run:

```
elliott@ubuntu-linux:~$ kate  
/proc/cpuinfo
```



The screenshot shows the Kate text editor interface with the title bar "cpuinfo — Kate". The menu bar includes File, Edit, View, Projects, Bookmarks, Sessions, Tools, Settings, and Help. The document tab is titled "cpuinfo". The left sidebar shows a "Documents" section with a folder icon. The main text area displays the /proc/cpuinfo file content, which contains detailed information about the system's CPU. The status bar at the bottom shows "Line 5, Column 55" and "Search and Replace".

```
processor : 0
vendor_id : GenuineIntel
cpu family : 6
model : 61
model name : Intel(R) Core(TM) i5-5300U CPU @ 2.30GHz
stepping : 4
cpu MHz : 2294.676
cache size : 3072 KB
physical id : 0
siblings : 1
core id : 0
cpu cores : 1
apicid : 0
initial apicid : 0
fpu : yes
fpu_exception : yes
cpuID level : 20
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
pat pse36 clflush mmx fxsr sse sse2 syscall nx rdtscp lm constant_tsc
rep_good nopl xtopology nonstop_tsc cpuid dni pclmulqdq monitor ssse3 cx16
pcid sse4_1 sse4_2 x2apic movbe popcnt aes xsave avx rdrand hypervisor
lahf_lm abm 3dnowprefetch invpcid_single pti fsgsbase avx2 invpcid rdseed
flush_lld
bugs : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf
bogomips : 4589.35
clflush size : 64
cache_alignment : 64
address sizes : 39 bits physical, 48 bits virtual
power management:
```

Graphical editors – gedit and kate

- You can also use the graphical editors to create new files on your system.
- For example, if you want to create a file named cats.txt in /home/elliot, then you can run the gedit /home/elliot/cats.txt command:

```
elliot@ubuntu-linux:~$ gedit /home/elliot/cats.txt
```



Graphical editors – gedit and kate

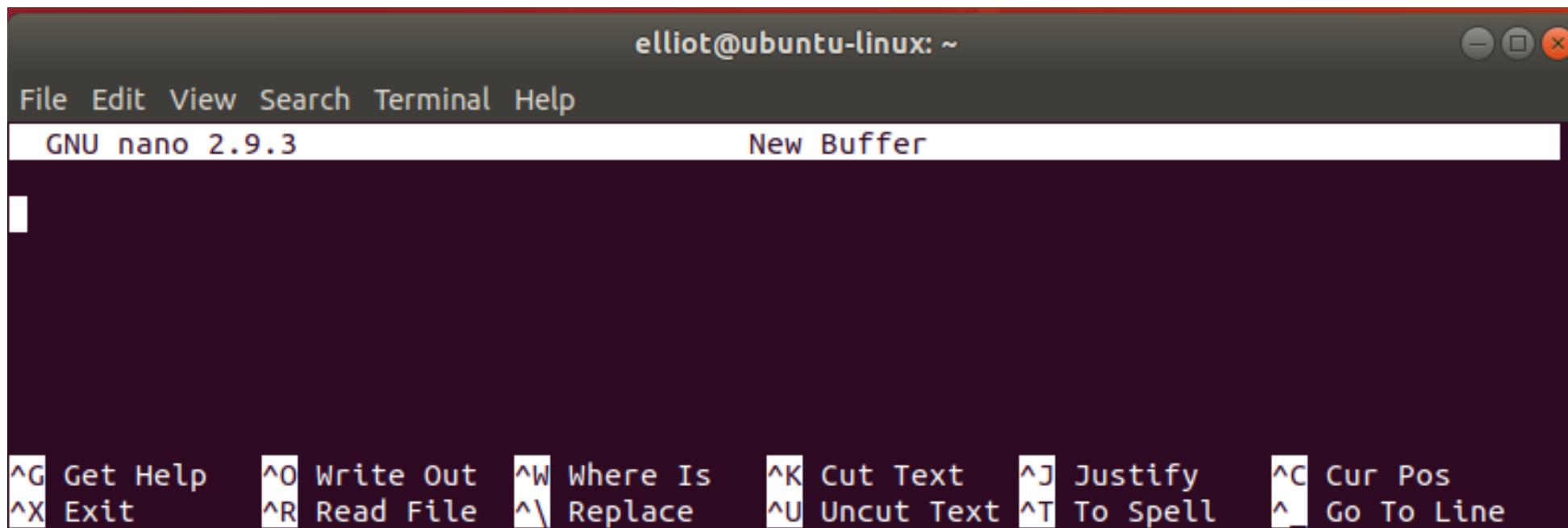
- Now insert the line "I love cats!" then save and close the file.
- The file cats.txt now exists in my home directory, and I can view it with the cat command:

```
elliott@ubuntu-linux:~$ pwd  
/home/elliott  
elliott@ubuntu-linux:~$ ls -l cats.txt  
-rw-r--r-- 1 elliot elliot 13 Feb 2 14:54 cats.txt  
elliott@ubuntu-linux:~$ cat cats.txt  
I love cats!
```

The nano editor

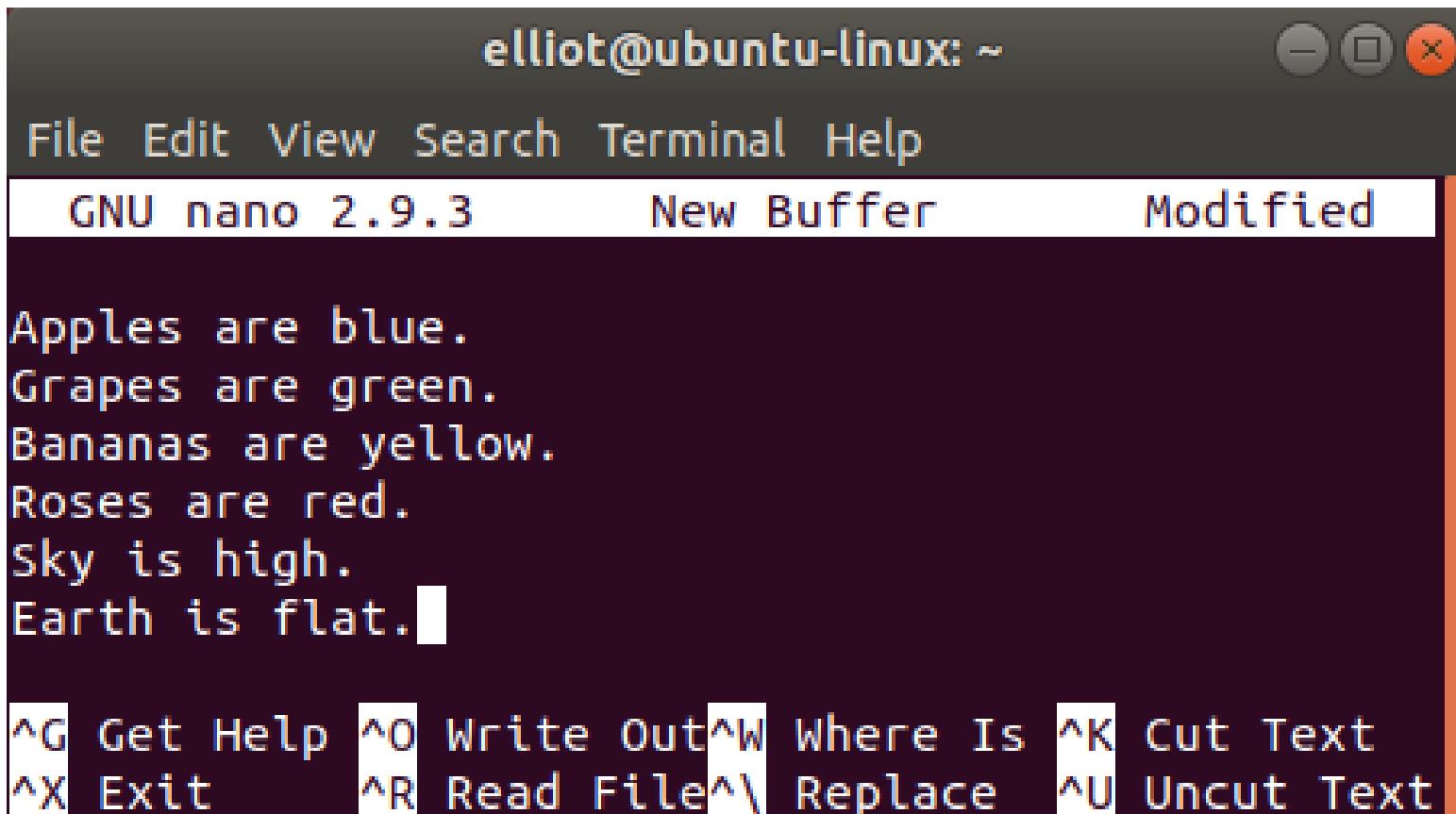
- The nano editor is a very popular and easy-to-use command-line editor.
- You can open the nano editor by running the nano command:

elliot@ubuntu-linux:~\$ nano



The nano editor

- Now add the six lines that are shown in the following screenshot:



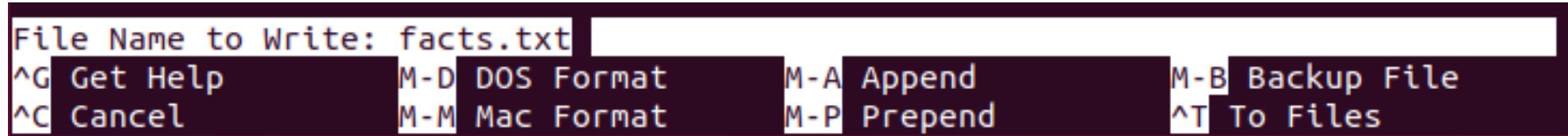
The nano editor

- Look at the bottom of the nano editor screen; you will see a lot of shortcuts:

```
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos  
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell  ^L Go To Line
```

The nano editor

- Notice that the Ctrl+O shortcut is triggered by pressing Ctrl and then the letter O.
- You don't have to press the + key or the upper case letter O.
- Now let's use the shortcut Ctrl+O to save the file; it will ask you for a filename, you can insert facts.txt:



The nano editor

- Press Enter to confirm.
- Now let's exit the nano editor (use the Ctrl+X shortcut) to verify that the file facts.txt is created:

```
elliott@ubuntu-linux:~$ ls -l facts.txt  
-rw-r--r-- 1 elliot elliot 98 Apr 30 15:17 facts.txt
```

- Now let's open facts.txt again to fix the false facts we have added there! To open the file facts.txt with the nano editor, you can run the nano facts.txt command:

```
elliott@ubuntu-linux:~$ nano facts.txt
```

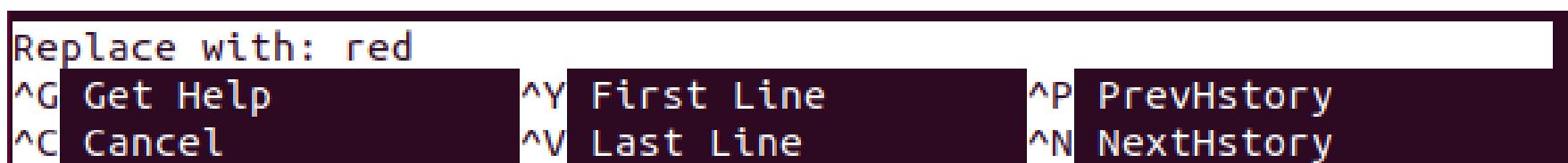
The nano editor

- When you press **Ctrl+**, it will ask you to enter the word that you want to replace; you can enter blue, as shown in the following screenshot:



A screenshot of the nano editor's search interface. The top bar has a white input field containing "Search (to replace): blue". Below the input field is a menu bar with several options: "Get Help", "Case Sens", "Backwards", "First Line", "PrevHstory", "Cancel", "Regexp", "No Replace", "Last Line", and "NextHstory".

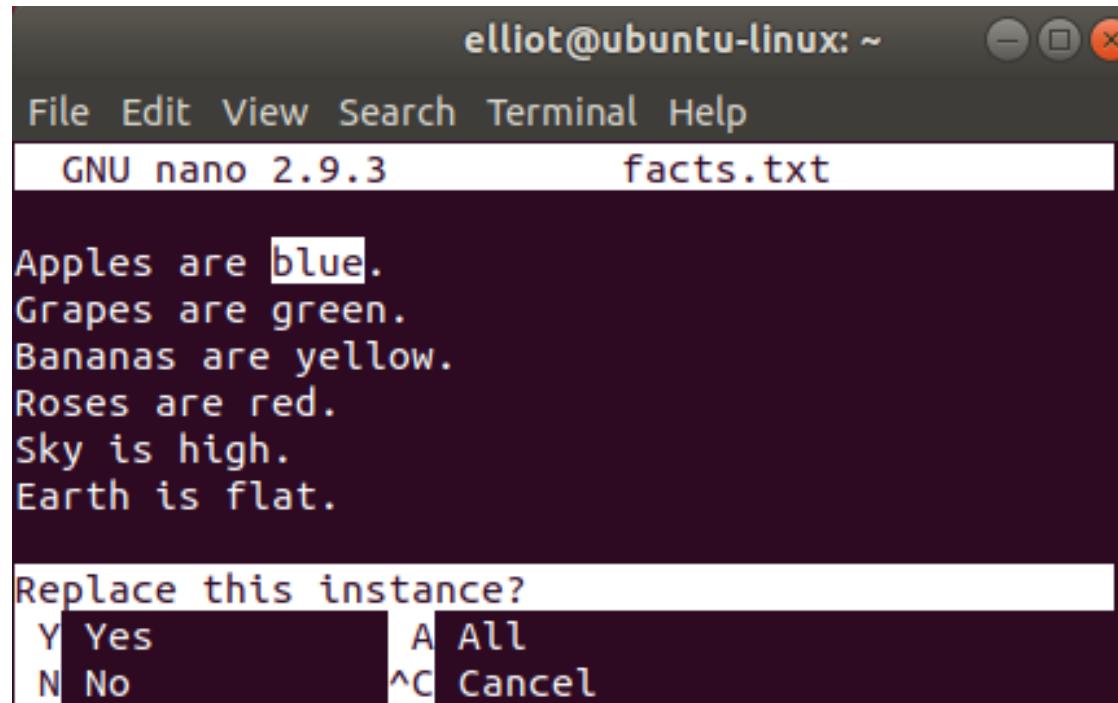
- Hit Enter, and then it will ask you to enter the substitute word. You can enter red, as shown in the following screenshot:



A screenshot of the nano editor's replace interface. The top bar has a white input field containing "Replace with: red". Below the input field is a menu bar with several options: "Get Help", "First Line", "PrevHstory", "Cancel", "Last Line", and "NextHstory".

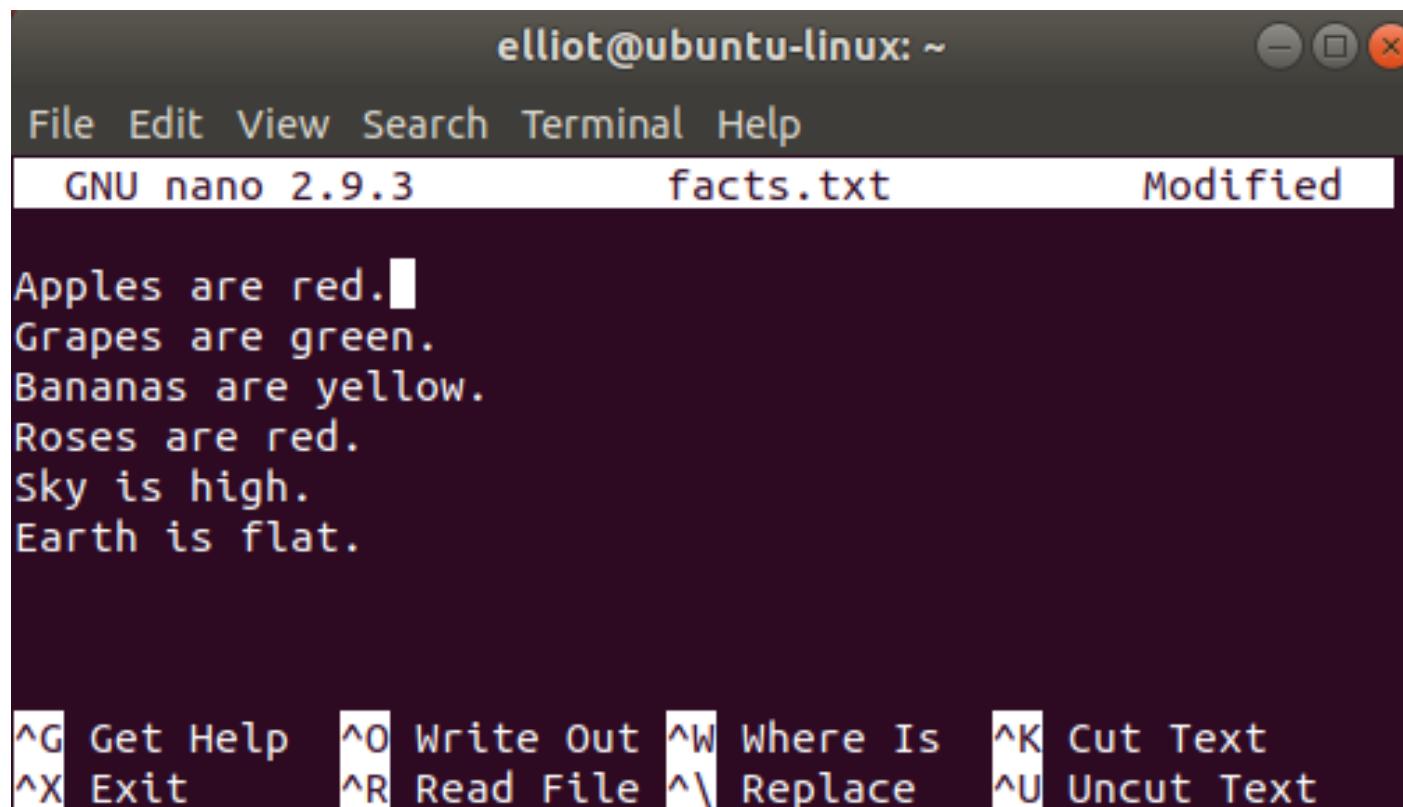
The nano editor

- You can then hit Enter, and it will go through each instance of the word blue and ask you if you want to replace it.
- Luckily, we only have one occurrence of blue.



The nano editor

- Press Y and BOOM! The word red replaced blue.



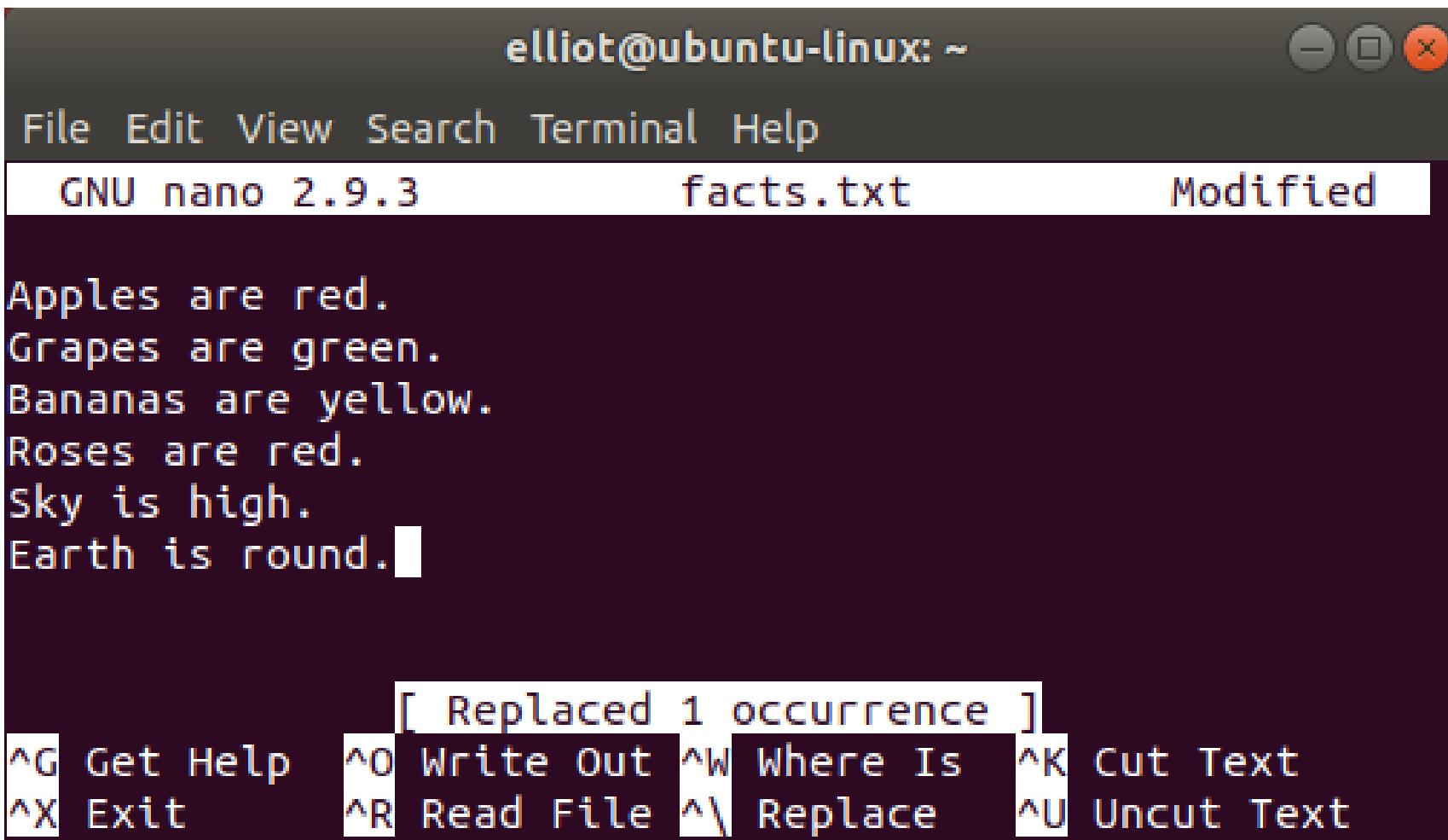
A screenshot of the nano text editor window on a Linux desktop. The title bar shows "elliot@ubuntu-linux: ~". The menu bar includes File, Edit, View, Search, Terminal, and Help. The status bar shows "GNU nano 2.9.3", "facts.txt", and "Modified". The main text area contains the following text:

```
Apples are red.  
Grapes are green.  
Bananas are yellow.  
Roses are red.  
Sky is high.  
Earth is flat.
```

The bottom of the window displays a series of keyboard shortcuts:

$\wedge G$	Get Help	$\wedge O$	Write Out	$\wedge W$	Where Is	$\wedge K$	Cut Text
$\wedge X$	Exit	$\wedge R$	Read File	$\wedge \backslash$	Replace	$\wedge U$	Uncut Text

The nano editor



A screenshot of the nano text editor window on a Linux desktop. The title bar shows the user 'elliott@ubuntu-linux: ~'. The menu bar includes File, Edit, View, Search, Terminal, and Help. The status bar displays 'GNU nano 2.9.3', the file name 'facts.txt', and 'Modified'. The main text area contains the following text:

```
Apples are red.  
Grapes are green.  
Bananas are yellow.  
Roses are red.  
Sky is high.  
Earth is round.
```

In the bottom right corner of the text area, there is a message: '[Replaced 1 occurrence]'. The bottom of the window shows a series of keyboard shortcuts:

^G Get Help	^O Write Out	^W Where Is	^K Cut Text
^X Exit	^R Read File	^\\ Replace	^U Uncut Text

The vi editor

- Let's open the facts.txt file with the vi editor; to do that, you run the vi facts.txt command:

elliot@ubuntu-linux:~\$ vi facts.txt

- This will open the vi editor, as shown in the following screenshot:

Apples are red.
Grapes are green.
Bananas are yellow.
Roses are red.
Sky is high.
Earth is round.

"facts.txt" 6 lines, 98 characters

Insert mode

Key	What it does
i	Inserts text before the current cursor position.
I	Inserts text at the beginning of the current line.
a	Appends text after the current cursor position.
A	Appends text after the end of the current line.
o	Creates a new line below the current line.
O	Creates a new line above the current line.

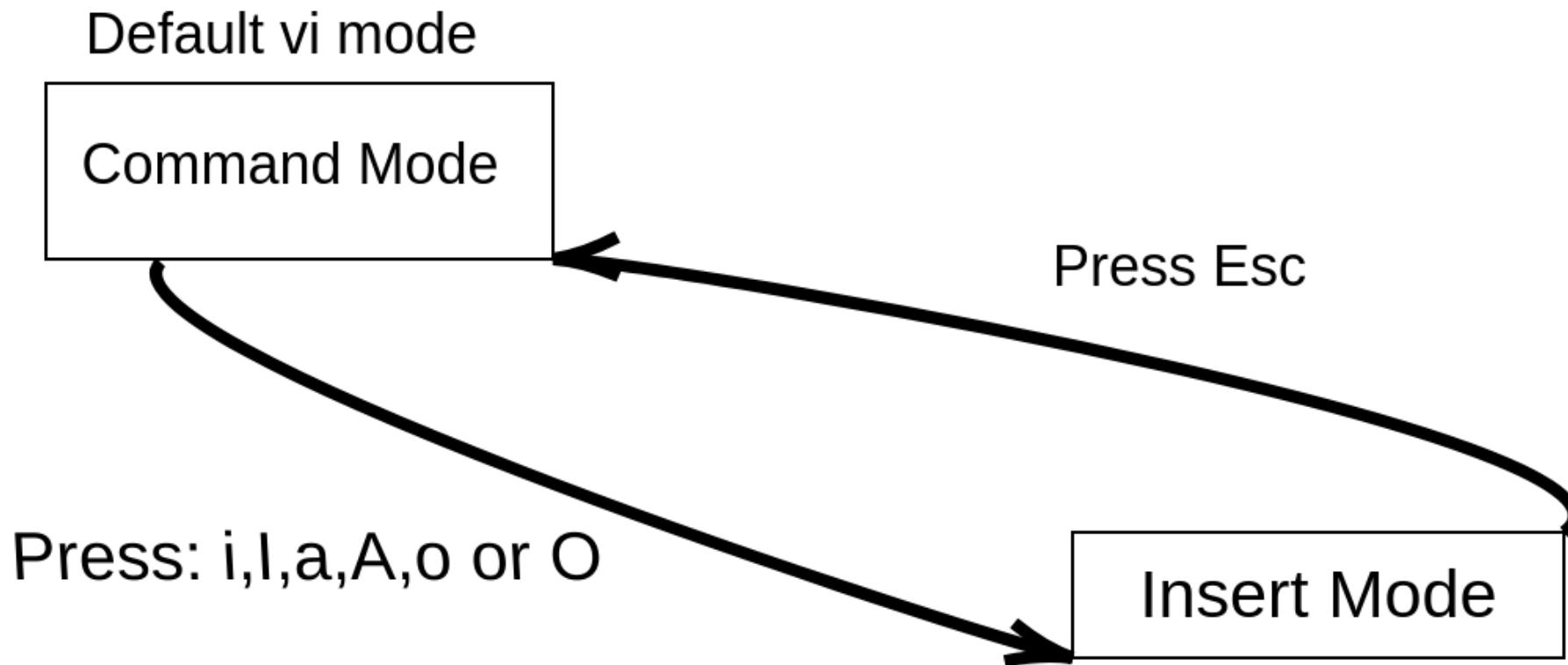
Insert mode

- You can navigate in the vi editor with your arrow keys, just like you would do in the nano editor.
- Now navigate to the last line in the file facts.txt and then press the letter o to switch into insert mode.
- You can now add the line "Linux is cool!"

```
Apples are red.  
Grapes are green.  
Bananas are yellow.  
Roses are red.  
Sky is high.  
Earth is round.  
Linux is cool! █
```

Insert mode

- With insert mode, you can add as much text as you want.
- To switch back to command mode, you need to press the Esc key.



vi command	What it does
yy	Copy (yank) the current line.
3yy	Copy (yank) three lines (starting with the current line).
yw	Copy (yank) one word starting at the cursor position.
2yw	Copy (yank) two words starting at the cursor position.
p	Paste after the current cursor position.
P	Paste before the current cursor position.
dd	Cut (delete) the current line.
4dd	Cut (delete) four lines (starting with the current line).
dw	Cut (delete) one word starting at the cursor position.
x	Delete the character at the cursor position.
u	Undo the last change.
U	Undo all changes to the line.
/red	Search for the word red in the file.
:%s/bad/good	Replace the word bad with good.
:set number	Show line numbers.
:set nonumber	Hide line numbers.
:7	Go to line number 7.
G	Jump to the end of the file.
gg	Jump to the beginning of the file.

Command mode

Command mode

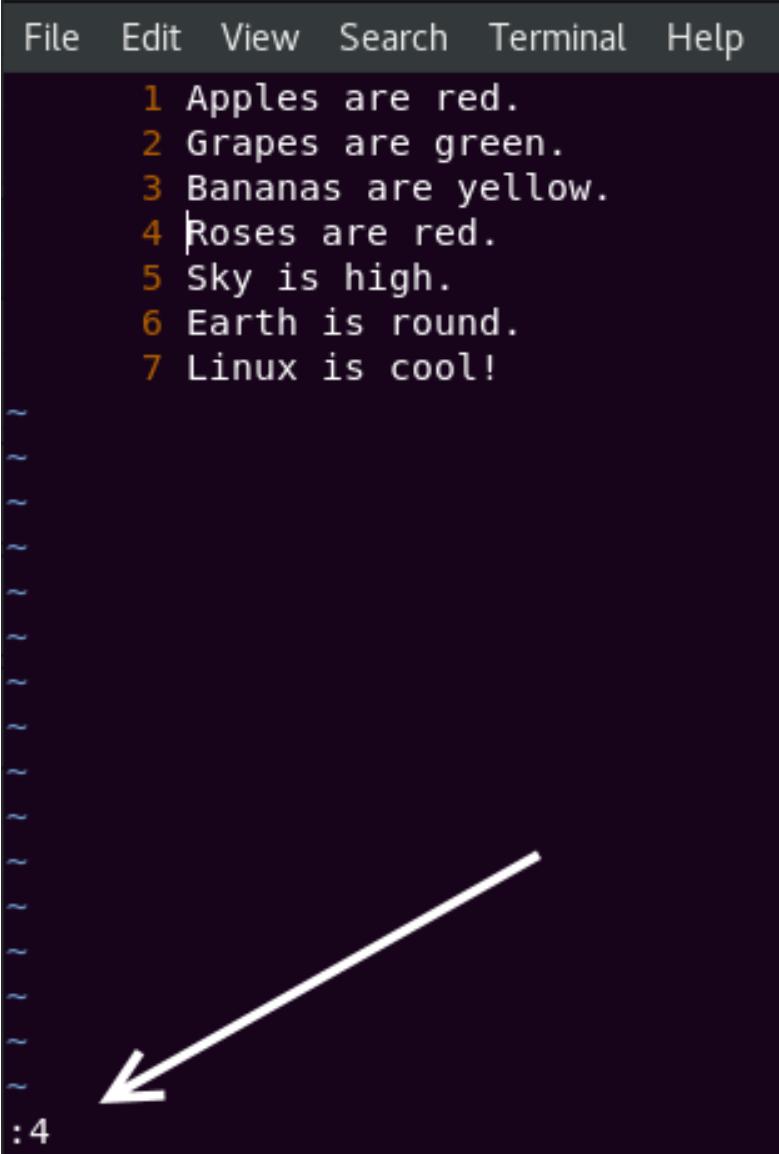
- Let's start by showing line numbers as it will make our life much easier!
- To do that, you run the :set number command, as shown in the following screenshot:



```
File Edit View Search Terminal Help  
1 Apples are red.  
2 Grapes are green.  
3 Bananas are yellow.  
4 Roses are red.  
5 Sky is high.  
6 Earth is round.  
7 Linux is cool!  
:set number
```

Command mode

- Now let's copy line 4. You want to make sure the cursor is on line 4; you can do that by running the :4 command, as shown in the following screenshot:



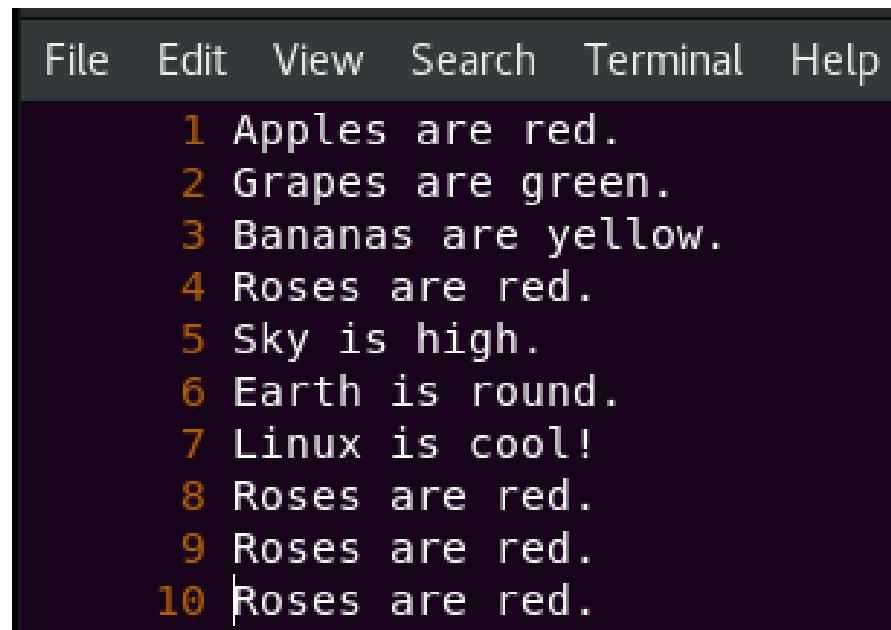
A screenshot of a terminal window with a dark background and light-colored text. At the top, there is a menu bar with options: File, Edit, View, Search, Terminal, and Help. Below the menu, seven lines of text are displayed, each preceded by a small orange number (1 through 7). Lines 1 through 6 are standard text, while line 7 is a statement about Linux. A large white arrow points from the bottom left towards the number 4, which is located at the bottom of the screen. The number 4 is also highlighted with a small orange box.

```
File Edit View Search Terminal Help
1 Apples are red.
2 Grapes are green.
3 Bananas are yellow.
4 |Roses are red.
5 Sky is high.
6 Earth is round.
7 Linux is cool!
```

:4

Command mode

- Now press the sequence yy, and it will copy the entire line.
- Let's paste it three times at the end of the file.
- So navigate to the last line and then press p three times, it will paste the copied line three times, as shown in the following screenshot:



A screenshot of a terminal window with a dark background and light-colored text. The window has a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". Below the menu, there is a numbered list of ten items, each starting with a number from 1 to 10 followed by a sentence. The sentences are: 1 Apples are red., 2 Grapes are green., 3 Bananas are yellow., 4 Roses are red., 5 Sky is high., 6 Earth is round., 7 Linux is cool!, 8 Roses are red., 9 Roses are red., and 10 Roses are red. The numbers 1 through 10 are colored orange, while the rest of the text is white.

```
File Edit View Search Terminal Help
1 Apples are red.
2 Grapes are green.
3 Bananas are yellow.
4 Roses are red.
5 Sky is high.
6 Earth is round.
7 Linux is cool!
8 Roses are red.
9 Roses are red.
10 Roses are red.
```

Command mode

- Alright! Let's replace the word cool with awesome because we all know Linux is not just cool; it's awesome!
 - To do that, you run the `:%s/cool/awesome` command, as shown in the following screenshot:

Command mode

- Let's also replace the word Roses with Cherries because we all know that not all roses are red.
- To do that, run the `:%s/Roses/Cherries` command, as shown in the following screenshot:

```
1 Apples are red.  
2 Grapes are green.  
3 Bananas are yellow.  
4 Cherries are red.  
5 Sky is high.  
6 Earth is round.  
7 Linux is awesome!  
8 Cherries are red.  
9 Cherries are red.  
10 Cherries are red.
```

4 substitutions on 4 lines

Saving and exiting vi

- Eventually, when you are done viewing or editing a file in vi, you would want to exit the vi editor.
- There are multiple ways you can use to exit the vi editor, Table lists all of them.

vi command	What it does
:w	Save the file but do not quit vi.
:wq	Save the file and quit vi.
ZZ	Save the file and quit vi (same as :wq, just faster!).
:x	Save the file and quit vi (same as :wq or ZZ).
:q	Quit vi without saving.
:q!	Forcefully quit vi without saving.

Saving and exiting vi

- All achieve the same result, that is, saving and exiting vi

```
1 Apples are red.  
2 Grapes are green.  
3 Bananas are yellow.  
4 Cherries are red.  
5 Sky is high.  
6 Earth is round.  
7 Linux is awesome!  
8 Cherries are red.  
9 Cherries are red.  
10 Cherries are red.
```

```
:wq|
```



- To view the facts.txt file that we created, you can run the cat facts.txt command:

```
elliott@ubuntu-linux:~$ cat facts.txt
```

Apples are red.

Grapes are green.

Bananas are yellow.

Cherries are red.

Sky is high.

Earth is round.

Linux is awesome!

Cherries are red.

Cherries are red.

Cherries are red.

The cat command

The cat command

- Now let's view each of the three files using the cat command:

```
elliot@ubuntu-linux:~$ cat file1.txt
```

First File

```
elliot@ubuntu-linux:~$ cat file2.txt
```

Second File

```
elliot@ubuntu-linux:~$ cat file3.txt
```

Third File

The cat command

- Now let's concatenate both file1.txt and file2.txt together by running the cat file1.txt file2.txt command:

```
elliott@ubuntu-linux:~$ cat file1.txt file2.txt
```

First File

Second File

- We can also concatenate all three files:

```
elliott@ubuntu-linux:~$ cat file1.txt file2.txt file3.txt
```

First File

Second File

Third File

The cat command

- Keep in mind that order matters; for example, running the cat file2.txt file1.txt command:

```
elliott@ubuntu-linux:~$ cat file2.txt file1.txt  
Second File  
First File
```

- For example, if you want to view the facts.txt file in reverse order, you can run the tac facts.txt command:

```
elliott@ubuntu-linux:~$ tac facts.txt
```

Cherries are red.

Cherries are red.

Cherries are red.

Linux is awesome!

Earth is round.

Sky is high.

Cherries are red.

Bananas are yellow.

Grapes are green.

Apples are red.

The tac command

The more command

```
elliott@ubuntu-linux:~$ more /etc/services
# Network services, Internet style
# Note that it is presently the policy of IANA to assign a single well-known
# port number for both TCP and UDP; hence, officially ports have two entries
# even if the protocol doesn't support UDP operations.

tcpmux 1/tcp # TCP port service multiplexer
systat 11/tcp users
netstat 15/tcp ftp 21/tcp
fsp 21/udp fspd
ssh 22/tcp # SSH Remote Login Protocol
telnet 23/tcp
smtp 25/tcp mail
whois 43/tcp nickname
tacacs 49/tcp # Login Host Protocol (TACACS)
tacacs 49/udp
--More--(7%)
```

The less command

- The less command is another pager program, just like more; it allows you to view text files one page at a time.
- The advantage of less is that you can use the UP/DOWN arrow keys to navigate through the file. Also, less is faster than more.
- You can view the /etc/services file with less by running the command:

```
elliot@ubuntu-linux:~$ less /etc/services
```

- You can also use more navigation keys with less.

- For example, we know that facts.txt has ten lines in it, and so running the head facts.txt command will display all the file contents:

```
elliott@ubuntu-linux:~$ head facts.txt
```

Apples are red.

Grapes are green.

Bananas are yellow.

Cherries are red.

Sky is high.

Earth is round.

Linux is awesome!

Cherries are red.

Cherries are red.

Cherries are red.

Heads or tails?

Heads or tails?

- You can also pass the `-n` option to specify the number of lines you wish to view.
- For example, to display the first three lines of `facts.txt`, you can run the `head -n 3 facts.txt` command:

```
elliot@ubuntu-linux:~$ head -n 3 facts.txt
```

Apples are red.

Grapes are green.

Bananas are yellow.

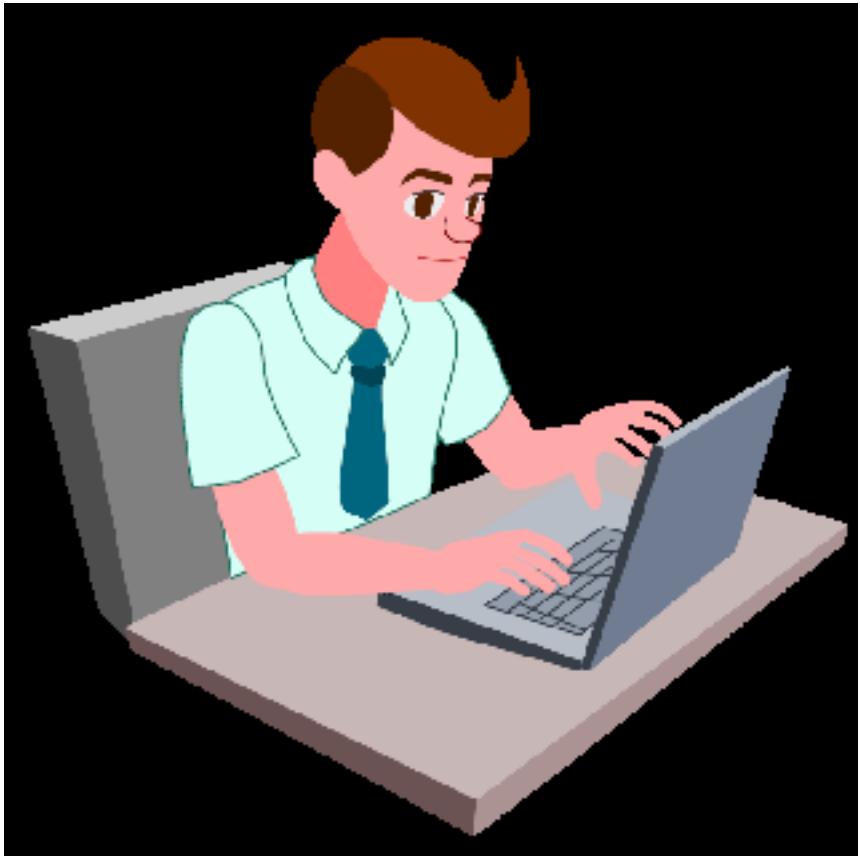
Heads or tails?

- On the other hand, the tail command displays the last few lines of a file.
- By default, it shows the last ten lines.
- You can also use the -n option to specify the number of lines you wish to view, For example, to display the last two lines in facts.txt, you can run the tail -n 2 facts.txt command:

```
elliot@ubuntu-linux:~$ tail -n 2 facts.txt
Cherries are red.
Cherries are red.
```

Knowledge check

- For the following exercises, open up your Terminal and try to solve the following tasks:
 1. Only view the first two lines of the file facts.txt.
 2. Only view the last line of the file facts.txt.
 3. Display the contents of the file facts.txt in a reversed order.
 4. Open the file facts.txt using the vi editor.
 5. Exit the vi editor and consider yourself one of the elites.



"Complete Lab"

4. Copying, Moving, and Deleting Files

Copying one file

- Sometimes you need to copy a single file, Luckily this is a simple operation on the command line.
- I have a file named cats.txt in my home directory:

```
elliot@ubuntu-linux:~$ cat cats.txt
I love cars!
I love cats!
I love penguins!
elliot@ubuntu-linux:~$
```

Copying one file

- I can use the cp command to make a copy of cats.txt named copycats.txt as follows:

```
elliott@ubuntu-linux:~$ cp cats.txt copycats.txt
```

```
elliott@ubuntu-linux:~$ cat copycats.txt
```

I love cars!

I love cats!

I love penguins!

```
elliott@ubuntu-linux:~$
```

Copying one file

- I can also copy the file cats.txt to another directory.
- For example, I can copy the file cats.txt to /tmp by running the cp cats.txt /tmp command:

```
elliot@ubuntu-linux:~$ cp cats.txt /tmp  
elliot@ubuntu-linux:~$ cd /tmp  
elliot@ubuntu-linux:/tmp$ ls  
cats.txt  
elliot@ubuntu-linux:/tmp$
```

Copying one file

- Notice that the copied file has the same name as the original file.
- I can also make another copy in /tmp with a different name:

```
elliott@ubuntu-linux:~$ cp cats.txt /tmp/cats2.txt
```

```
elliott@ubuntu-linux:~$ cd /tmp
```

```
elliott@ubuntu-linux:/tmp$ ls
```

```
cats2.txt  cats.txt
```

```
elliott@ubuntu-linux:/tmp$
```

Copying multiple files

- You may also want to copy multiple files at once.
- To demonstrate, let's begin by creating three files apple.txt, banana.txt, and carrot.txt in Elliot's home directory:

```
elliot@ubuntu-linux:~$ touch apple.txt banana.txt carrot.txt  
elliot@ubuntu-linux:~$ ls  
apple.txt carrot.txt copycats.txt dir1  
banana.txt cats.txt Desktop  
elliot@ubuntu-linux:~$
```

Copying multiple files

- To copy the three newly created files to /tmp, you can run the cp apple.txt ba- nana.txt carrot.txt /tmp command:

```
elliott@ubuntu-linux:~$ cp apple.txt banana.txt carrot.txt /tmp  
elliott@ubuntu-linux:~$ cd /tmp
```

```
elliott@ubuntu-linux:/tmp$ ls  
apple.txt banana.txt carrot.txt cats2.txt cats.txt  
elliott@ubuntu-linux:/tmp$
```

- Child's play! In general, the cp command follows the syntax:
`cp source_file(s) destination`

Copying one directory

- You may also want to copy an entire directory; that's also easily accomplished.
- To demonstrate, create a directory named cities in your home directory, and inside cities, create three files paris, tokyo, and london as follows:

```
elliot@ubuntu-linux:~$ mkdir cities
```

```
elliot@ubuntu-linux:~$ cd cities/
```

```
elliot@ubuntu-linux:~/cities$ touch paris tokyo london
```

```
elliot@ubuntu-linux:~/cities$ ls
```

```
london paris tokyo
```

Copying one directory

- Now if you want to copy the cities directory to /tmp, you have to pass the recursive -r option to the cp command as follows:

```
elliot@ubuntu-linux:~/cities$ cd ..  
elliot@ubuntu-linux:~$ cp -r cities /tmp
```

- You will get an error message if you omitted the -r option:

```
elliot@ubuntu-linux:~$ cp cities /tmp  
cp: -r not specified; omitting directory 'cities'
```

Copying one directory

- You can verify that the cities directory is copied to /tmp by listing the files in /tmp:

```
elliot@ubuntu-linux:~$ cd /tmp  
elliot@ubuntu-linux:/tmp$ ls  
apple.txt banana.txt carrot.txt cats2.txt cats.txt cities  
elliot@ubuntu-linux:/tmp$ ls cities  
london paris tokyo
```

Copying multiple directories

- You can also copy multiple directories the same way you copy multiple files; the only difference is that you have to pass the recursive -r option to the cp command.
- To demonstrate, create the three directories d1, d2, and d3 in Elliot's home directory:

```
elliot@ubuntu-linux:~$ mkdir d1 d2 d3
```

Copying multiple directories

- Now you can copy all three directories to /tmp by running the cp -r d1 d2 d3 /tmp command:

```
elliott@ubuntu-linux:~$ cp -r d1 d2 d3 /tmp
```

```
elliott@ubuntu-linux:~$ cd /tmp
```

```
elliott@ubuntu-linux:/tmp$ ls
```

```
apple.txt banana.txt carrot.txt cats2.txt cats.txt cities d1 d2 d3
```

Moving one file

- To do this, you can use the mv command.
- For example, you can move the file copycats.txt from Elliot's home directory to /tmp by running the mv copycats.txt /tmp command:

```
elliot@ubuntu-linux:~$ mv copycats.txt /tmp
elliot@ubuntu-linux:~$ ls
apple.txt  carrot.txt  cities  d2  Desktop  Downloads
banana.txt  cats.txt  d1  d3  dir1  Pictures
elliot@ubuntu-linux:~$ cd /tmp
elliot@ubuntu-linux:/tmp$ ls
apple.txt  carrot.txt  cats.txt  copycats.txt  d2
banana.txt  cats2.txt  cities  d1          d3
```

Moving multiple files

- You can also move multiple files the same way you can copy multiple files.
- For example, you can move the three files apple.txt, banana.txt, and carrot.txt from /tmp to /home/elliot/d1 as follows:

```
elliot@ubuntu-linux:/tmp$ mv apple.txt banana.txt carrot.txt
```

```
/home/elliot/d1
```

```
elliot@ubuntu-linux:/tmp$ ls
```

```
cats2.txt cats.txt cities copycats.txt d1 d2 d3
```

```
elliot@ubuntu-linux:/tmp$ cd /home/elliot/d1
```

```
elliot@ubuntu-linux:~/d1$ ls
```

```
apple.txt banana.txt carrot.txt
```

```
elliot@ubuntu-linux:~/d1$
```

Moving multiple files

- As you can see, the three files apple.txt, banana.txt, and carrot.txt are no longer located in /tmp as they all moved to /home/elliot/d1.
- In general, the mv command follows the syntax:

`mv source_file(s) destination`

Moving one directory

- You can also use the mv command to move directories.
- For example, if you want to move the directory d3 and put it inside d2, then you can run the mv d3 d2 command:

```
elliot@ubuntu-linux:~$ mv d3 d2
```

```
elliot@ubuntu-linux:~$ cd d2
```

```
elliot@ubuntu-linux:~/d2$ ls
```

```
d3
```

```
elliot@ubuntu-linux:~/d2$
```

Moving multiple directories

- You can also move multiple directories at once.
- To demonstrate, create a directory named big in Elliot's home directory:

```
elliot@ubuntu-linux:~$ mkdir big
```

- Now you can move the three directories d1, d2, and cities to the big directory as follows:

```
elliot@ubuntu-linux:~$ mv d1 d2 cities big
```

```
elliot@ubuntu-linux:~$ ls big
```

```
cities d1 d2
```

```
elliot@ubuntu-linux:~$
```

Renaming files

- You can also use the mv command to rename files.
- For example, if you want to rename the file cats.txt to dogs.txt, you can run the mv cats.txt dogs.txt command:

```
elliot@ubuntu-linux:~$ mv cats.txt dogs.txt
```

```
elliot@ubuntu-linux:~$ cat dogs.txt
```

I love cars!

I love cats!

I love penguins!

```
elliot@ubuntu-linux:~$
```

Renaming files

- If you want to rename the directory big to small, you can run the mv big small command:

```
elliot@ubuntu-linux:~$ mv big small
```

```
elliot@ubuntu-linux:~$ ls small
```

```
cities d1 d2
```

```
elliot@ubuntu-linux:~$
```

Hiding files

- You can hide any file by renaming it to a name that starts with a dot.
- Let's try it; you can hide the file dogs.txt by renaming it to .dogs.txt as follows:

```
elliott@ubuntu-linux:~$ ls  
apple.txt banana.txt carrot.txt dogs.txt Desktop dir1 small  
elliott@ubuntu-linux:~$ mv dogs.txt .dogs.txt  
elliott@ubuntu-linux:~$ ls  
apple.txt banana.txt carrot.txt Desktop dir1 small  
elliott@ubuntu-linux:~$
```

Hiding files

- you can see, the file dogs.txt is now hidden as it got renamed to .dogs.txt.
- You can unhide .dogs.txt by renaming it and removing the leading dot from the filename:

```
elliott@ubuntu-linux:~$ mv .dogs.txt dogs.txt
```

```
elliott@ubuntu-linux:~$ ls
```

```
apple.txt banana.txt carrot.txt dogs.txt Desktop dir1 small
```

```
elliott@ubuntu-linux:~$
```

Removing files

- You can use the rm command to remove (delete) files.
- For example, if you want to remove the file dogs.txt, you can run the rm dogs.txt command:

```
elliott@ubuntu-linux:~$ ls  
apple.txt banana.txt carrot.txt dogs.txt Desktop dir1 small  
elliott@ubuntu-linux:~$ rm dogs.txt  
elliott@ubuntu-linux:~$ ls  
apple.txt banana.txt carrot.txt Desktop dir1 small
```

Removing files

- You can also remove multiple files at once.
- For example, you can remove the three files apple.txt, banana.txt, and carrot.txt by running the rm apple.txt banana.txt carrot.txt command:

```
elliot@ubuntu-linux:~$ rm apple.txt banana.txt carrot.txt
```

```
elliot@ubuntu-linux:~$ ls
```

```
Desktop dir1 small
```

```
elliot@ubuntu-linux:~$
```

Removing directories

- You can pass the recursive -r option to the rm command to remove directories.
- To demonstrate, let's first create a directory named garbage in Elliot's home directory:

```
elliot@ubuntu-linux:~$ mkdir garbage
```

```
elliot@ubuntu-linux:~$ ls
```

```
Desktop dir1 garbage small
```

- Now let's try to remove the garbage directory:

```
elliot@ubuntu-linux:~$ rm garbage
```

```
rm: cannot remove 'garbage': Is a directory
```

```
elliot@ubuntu-linux:~$
```

Removing directories

- Shoot! I got an error because I didn't pass the recursive -r option.
- I will pass the recursive option this time:

```
elliot@ubuntu-linux:~$ rm -r garbage
```

```
elliot@ubuntu-linux:~$ ls
```

```
Desktop dir1 small
```

- Cool! We got rid of the garbage directory.
- You can also use the rmdir command to remove only empty directories.
- To demonstrate, let's create a new directory named garbage2 and inside it, create a file named old:

```
elliot@ubuntu-linux:~$ mkdir garbage2
```

```
elliot@ubuntu-linux:~$ cd garbage2
```

```
elliot@ubuntu-linux:~/garbage2$ touch old
```

Removing directories

- Now let's go back to Elliot's home directory and attempt to remove garbage2 with the rmdir command:

```
elliot@ubuntu-linux:~/garbage2$ cd ..
```

```
elliot@ubuntu-linux:~$ rmdir garbage2
```

```
rmdir: failed to remove 'garbage2': Directory not empty
```

- As you can see, it wouldn't allow you to remove a nonempty directory. Therefore, let's delete the file old that's inside garbage2 and then reattempt to remove garbage2:

```
elliot@ubuntu-linux:~$ rm garbage2.old
```

```
elliot@ubuntu-linux:~$ rmdir garbage2
```

```
elliot@ubuntu-linux:~$ ls
```

```
Desktop dir1 small
```

```
elliot@ubuntu-linux:~$
```

Removing directories

- For the final example in this lesson, let's create a directory named garbage3, then create two files a1.txt and a2.txt inside it:

```
elliott@ubuntu-linux:~$ mkdir garbage3
```

```
elliott@ubuntu-linux:~$ cd garbage3/
```

```
elliott@ubuntu-linux:~/garbage3$ touch a1.txt a2.txt
```

```
elliott@ubuntu-linux:~/garbage3$ ls
```

```
a1.txt a2.txt
```

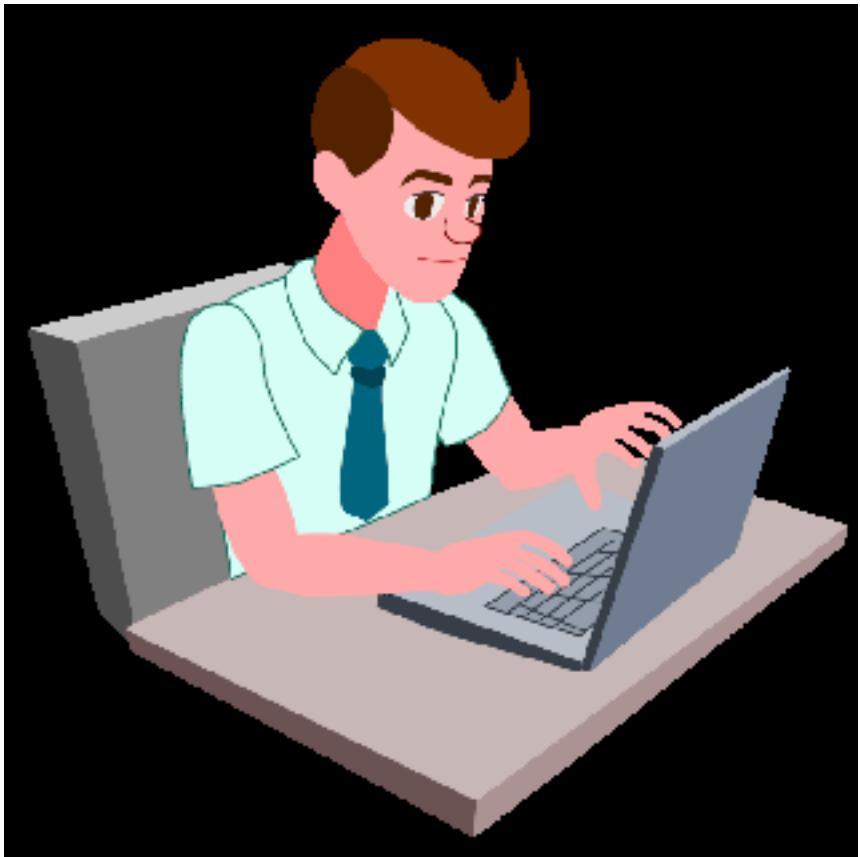
Removing directories

- Now let's get back to Elliot's home directory and attempt to remove garbage3:

```
elliot@ubuntu-linux:~/garbage3$ cd ..  
elliot@ubuntu-linux:~$ rmdir garbage3  
rmdir: failed to remove 'garbage3': Directory not empty  
elliot@ubuntu-linux:~$ rm -r garbage3  
elliot@ubuntu-linux:~$ ls  
Desktop dir1 Downloads Pictures small  
elliot@ubuntu-linux:~$
```

Knowledge check

- For the following exercises, open up your Terminal and try to solve the following tasks:
 1. Create three files hacker1, hacker2, and hacker3 in your home directory.
 2. Create three directories Linux, Windows, and Mac in your home directory.
 3. Create a file named cool inside the Linux directory you created in task 2.
 4. Create a file named boring inside the Windows directory you created in task 2.
 5. Create a file named expensive in the Mac directory you created in task 2.
 6. Copy the two files hacker1 and hacker2 to the /tmp directory.
 7. Copy the two directories Windows and Mac to the /tmp directory.
 8. Move the file hacker3 to the /tmp directory.
 9. Move the directory Linux to the /tmp directory.
 10. Remove the file expensive from the Mac directory (in your home directory).
 11. Remove the directory Mac from your home directory.
 12. Remove the directory Windows from your home directory.
 13. Remove the file hacker2 from your home directory.
 14. Rename the file hacker1 to hacker01.



"Complete Lab"

5. Linux Command Types

The four categories of linux commands

All Linux commands must fall into one of these following four categories:

1. An executable program: Which is usually written in the C programming language. The cp command is an example of an executable command.
2. An alias: Which is basically another name for a command (or a group of commands).
3. A shell builtin: The shell supports internal commands as well. The exit and cd commands are two examples of a shell builtin command.
4. A shell function: These are functions that help us achieve a specific task and are essential in writing shell scripts. We will cover this in more detail later, for now, just know they exist.

Determining a command's type

- You can use the type command to determine the type (category) of a command.

For example, if you want to know the type of the pwd command you can simply run the type pwd command:

```
elliot@ubuntu-linux:~$ type pwd  
pwd is a shell builtin
```

- So now you know that the pwd command is a shell builtin command.
Now let's figure out the type of the ls command:

```
elliot@ubuntu-linux:~$ type ls  
ls is aliased to `ls --color=auto'
```

Determining a command's type

- Now you know why you see a colorful output every time you run the ls command, Let's see the type of the date command:

```
elliot@ubuntu-linux:~$ type date  
date is /bin/date
```

- Any command that lives in /bin or /sbin is an executable program.
- Therefore, we can conclude that the date command is an executable program as it resides in /bin.
- Finally, let's determine the type of the type command itself:

```
elliot@ubuntu-linux:~$ type type  
type is a shell builtin
```

Finding a command's location

- Every time you run an executable command, there a file somewhere on the system that gets executed.
- You can use the which command to determine the location of an executable command.
- For example, if you want to know the location of the rm command, you can run the which rm command:

```
elliot@ubuntu-linux:~$ which rm  
/bin/rm
```

- So now you know that rm lives in the /bin directory. Let's see the location of the reboot command:

```
elliot@ubuntu-linux:~$ which reboot  
/sbin/reboot
```

What does the command do?

- You can use the whatis command to get a brief description of what a command does.
- For example, if you want to know the purpose of the free command, you can run the whatis free command:

elliot@ubuntu-linux:~\$ whatis free

free (1) - Display amount of free and used memory in the system

- As you can see, the free command, as we already know, displays the amount of free and used memory in the system.
- Cool! Now let's see what the df command does:

elliot@ubuntu-linux:~\$ whatis df

df (1) - report file system disk space usage

What does the command do?

- Finally, let's see what the which command does:

```
elliot@ubuntu-linux:~$ whatis which  
which (1)      - locate a command
```

- In general, if you want to read the man page of a command, you can run:

`man command_name`

- For example, if you want to view the man page of the touch command, you can run the man touch command:

`elliott@ubuntu-linux:~$ man touch`

```
TOUCH(1)                                         User Commands
TOUCH(1)

NAME
    touch - change file timestamps

SYNOPSIS
    touch [OPTION]... FILE...

DESCRIPTION
    Update the access and modification times of each FILE to the current time.
    A FILE argument that does not exist is created empty, unless -c or -h is supplied.
    A FILE argument string of - is handled specially and causes touch to change the times of the file associated with standard output.
    Mandatory arguments to long options are mandatory for short options too.

    -a      change only the access time
    -c, --no-create
           do not create any files
    -d, --date=STRING
           parse STRING and use it instead of current time
    -f      (ignored)
    -h, --no-dereference
           affect each symbolic link instead of any referenced file (useful only on systems that can change the timestamps of a symlink)
    -m      change only the modification time
    -r, --reference=FILE
           use this file's times instead of current time
```

The man page

The man page

man keys	What it does
Space	Scrolls forward one page.
<i>Ctrl+F</i>	Scrolls forward one page (same as space).
<i>Ctrl+B</i>	Scrolls backward one page.
/word	Will search for a word (pattern) in the man page. For example, /access will search for the word access in the man page
<i>q</i>	Will quit the man page.
<i>n</i>	After you search for a word, you can use <i>n</i> to look for the next occurrence of the word in the man page.
<i>N</i>	After you search for a word, you can use <i>N</i> to look for the previous occurrence of the word in the man page.

The man page

- I can't stress enough the importance of man pages.
- They can be your best friend in the darkest moments, trust me!
- You should also know that there is a man page for man itself:

```
elliot@ubuntu-linux:~$ man man
```

Help for shell builtins

- If you play around enough with man pages, you may notice that a lot of shell builtin commands do not have a man page.
- For instance, there is no man page for the cd or the exit commands:

```
elliott@ubuntu-linux:~$ type cd
```

```
cd is a shell builtin
```

```
elliott@ubuntu-linux:~$ man cd
```

```
No manual entry for cd
```

```
elliott@ubuntu-linux:~$ type exit
```

```
exit is a shell builtin
```

```
elliott@ubuntu-linux:~$ man exit
```

```
No manual entry for exit
```

Help for shell builtins

- That's because shell builtin commands do not have man pages, but do not freak out just yet! You can still find help on how to use shell builtins by using the help command.
- For example, to get help on how to use the exit command, you can run:

```
elliott@ubuntu-linux:~$ help exit  
exit: exit [n]  
Exit the shell.
```

Exits the shell with a status of N. If N is omitted, the exit status is that of the last command executed.

Help for shell builtins

```
elliott@ubuntu-linux:~$ help cd
cd: cd [-L|-P] [dir]
      Change the shell working directory.

      Change the current directory to DIR. The default DIR is the value of
      the HOME shell variable.

      The variable CDPATH defines the search path for the directory containing DIR.
      Alternative directory names in CDPATH are separated by a colon (:).
      A null directory name is the same as the current directory.
      If DIR begins with a slash (/), then CDPATH is not used.

      If the directory is not found, and the shell option `cdable_vars' is set,
      the word is assumed to be a variable name. If that variable has a value,
      its value is used for DIR.

      Options:
          -L force symbolic links to be followed
          -P use the physical directory structure without following symbolic links
      The default is to follow symbolic links, as if '-L' were specified.

      Exit Status:
      Returns 0 if the directory is changed; non-zero otherwise.
```

The info page

- The GNU project launched the info pages as an alternative documentation to the man pages.
- The GNU project once claimed that man pages are outdated and needed replacement and so they came up with the info pages.
- You can view the info page of any command by running:

`info command_name`

The info page

- For example, to view the info page of the ls command, you can run the info ls command:

```
elliott@ubuntu-linux:~$ info ls

Next: dir invocation, Up: Directory listing

10.1 'ls': List directory contents
=====
The 'ls' program lists information about files (of any type, including directories). Options and file arguments can be intermixed arbitrarily, as usual.

For non-option command-line arguments that are directories, by default 'ls' lists the contents of directories, not recursively, and omitting files with names beginning with '.'. For other non-option arguments, by default 'ls' lists just the file name. If no non-option argument is specified, 'ls' operates on the current directory, acting as if it had been invoked with a single argument of '.'.

By default, the output is sorted alphabetically, according to the locale settings in effect.(1) If standard output is a terminal, the output is in columns (sorted vertically) and control characters are output as question marks; otherwise, the output is listed one per line and control characters are output as-is.

Because 'ls' is such a fundamental program, it has accumulated many options over the years. They are described in the subsections below; within each section, options are listed alphabetically (ignoring case). The division of options into the subsections is not absolute, since some options affect more than one aspect of 'ls''s operation.
```

The very helpful apropos command

- The apropos command is one of the most helpful and yet underrated Linux commands.
- Let's see a brief description of what the apropos command does:

```
elliott@ubuntu-linux:~$ whatis apropos  
apropos (1)      - search the manual page names and descriptions
```

The very helpful apropos command

- For example, let's say you want to rename a file, but you are unsure which Linux command to use; in this case, you can run the apropos rename command:

```
elliott@ubuntu-linux:~$ apropos rename
file-rename (1p)    - renames multiple files
File::Rename (3pm)  - Perl extension for renaming multiple files
gvfs-rename (1)     - (unknown subject)
mmove (1)          - move or rename an MSDOS file or subdirectory
mren (1)           - rename an existing MSDOS file
mv (1)             - move (rename) files
prename (1p)        - renames multiple files
rename (1)          - renames multiple files
rename.ul (1)       - rename files
```

The very helpful apropos command

- Let's say you want to view the calendar but you're unsure which command to use; in this case, you can run:

```
elliott@ubuntu-linux:~$ apropos calendar
```

```
cal (1)
```

- displays a calendar and the date of Easter

```
calendar (1)
```

- reminder service

```
ncal (1)
```

- displays a calendar and the date of Easter

The very helpful apropos command

- For the last example, let's say you want to display your CPU information, but you don't know which command to use; in this case, you can run:

```
elliott@ubuntu-linux:~$ apropos cpu
chcpu (8)          - configure CPUs
cpuid (4)          - x86 CPUID access device
cpuset (7)          - confine processes to processor and memory node
subsets
lscpu (1)          - display information about the CPU architecture
msr (4)            - x86 CPU MSR access device
sched (7)          - overview of CPU scheduling
taskset (1)         - set or retrieve a process's CPU affinity
```

The very helpful apropos command

- Here you go! You can see that it listed the lscpu command that we have used earlier.
- The apropos command is here to rescue you whenever you forget a command or you're unsure which command to use.
- You just have to supply a keyword (preferably a verb) that highlights what you want to accomplish to the apropos command:

apropos keyword

The very helpful apropos command

- The man -k command will display the same result as the apropos command.

```
elliott@ubuntu-linux:~$ man -k cpu
chcpu (8)          - configure CPUs
cpuid (4)          - x86 CPUID access device
cpuset (7)          - confine processes to processor and memory node
subsets
lscpu (1)          - display information about the CPU architecture
msr (4)            - x86 CPU MSR access device
sched (7)          - overview of CPU scheduling
taskset (1)         - set or retrieve a process's CPU affinity
```

The /usr/share/doc directory

- It may also include a TODO file that contains a list of unfinished tasks/features; contributors usually check the TODO files to help fix bugs and develop new features.
- To demonstrate, let's go to the nano documentation directory:

```
elliot@ubuntu-linux:~$ cd /usr/share/doc/nano  
elliot@ubuntu-linux:/usr/share/doc/nano$ pwd  
/usr/share/doc/nano
```

The /usr/share/doc directory

- Now list the contents of the directory to see what's inside:

```
elliott@ubuntu-linux:/usr/share/doc/nano$ ls  
AUTHORS          copyright faq.html      nano.html  README  TODO  
changelog.Debian.gz  examples IMPROVEMENTS.gz NEWS.gz  THANKS.gz
```

Knowledge check

For the following exercises, open up your Terminal and try to solve the following tasks:

1. You need to know if the echo command is a shell builtin or an executable program, which command would you run?
2. Display the location of the uptime command executable file.
3. Show a brief description of the mkdir command.
4. You forgot how to use the mv command, what are you going to do?
5. You forgot which command is used to display the calendar, what are you going to do?
6. The history command is a shell builtin and so it doesn't have a man page. You want to clear your history but don't know how. What are you going to do?



"Complete Lab"

6. Hard versus Soft Links

File inodes

When you go to a grocery store, you will find that each product has a set of attributes like:

- Product type: Chocolate
- Product price: \$2.50
- Product supplier: Kit Kat
- Amount left: 199

Displaying file inode number

- There are two commands you can use to view the inode number of a file:

`ls -i file`

`stat file`

- For example, to view the inode number of facts.txt, you can run the command `ls -i facts.txt`:

```
elliott@ubuntu-linux:~$ ls -i facts.txt
```

```
924555 facts.txt
```

Displaying file inode number

- It will spit out the inode number for you.
- You can also use the stat command:

```
elliot@ubuntu-linux:~$ stat facts.txt
```

File: facts.txt

Size: 173 Blocks: 8 IO Block: 4096 regular file

Device: 801h/2049d Inode: 924555 Links: 1

Access: (0644/-rw-r--r--)Uid: (1000/ tom) Gid: (1000/ tom)

Access: 2019-05-08 13:41:16.544000000 -0600

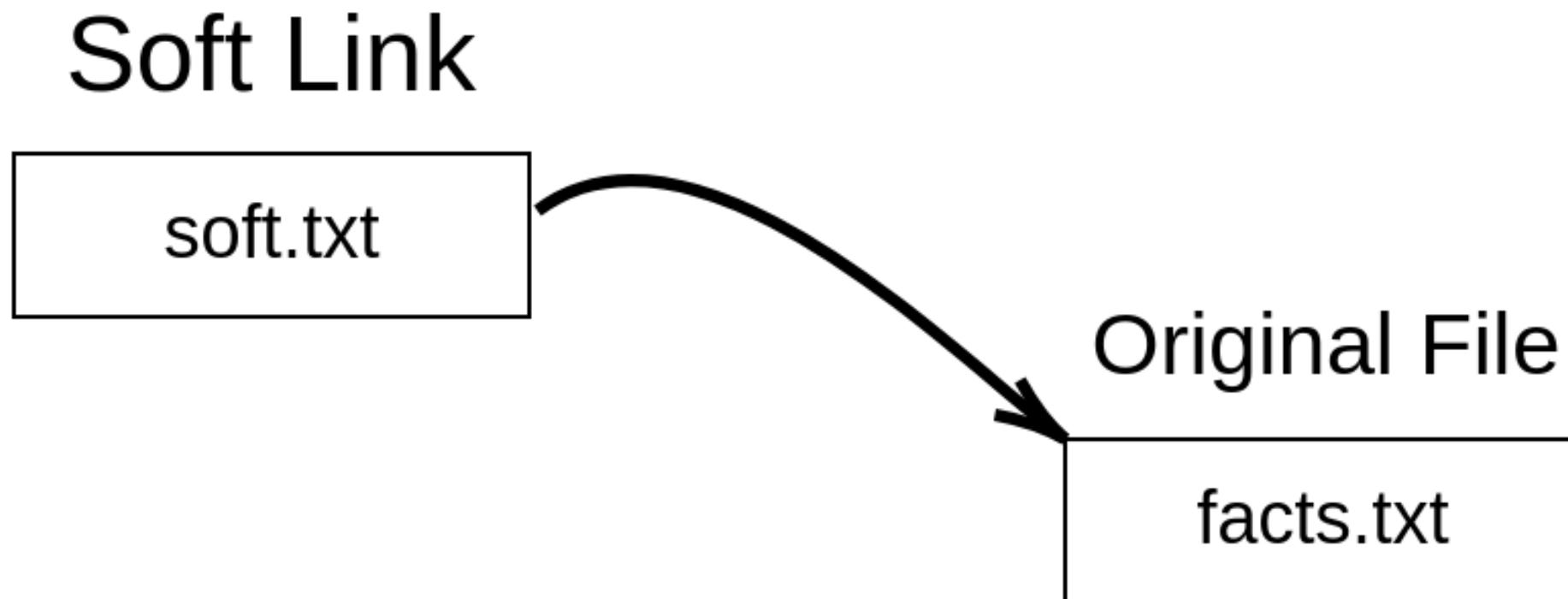
Modify: 2019-05-08 12:50:44.112000000 -0600

Change: 2019-05-08 12:50:44.112000000 -0600

Birth: -

Creating soft links

A picture is worth a thousand words, so the following diagram will help you visualize soft links.



Creating soft links

- To create a soft link, we use the ln command with the -s option as follows:

```
ln -s original_file soft_link
```

- So to create a soft link named soft.txt to the facts.txt file, you can run the command ln -s facts.txt soft.txt:

```
elliot@ubuntu-linux:~$ ln -s facts.txt soft.txt
```

- Now let's do a long listing on the soft link file soft.txt that we just created:

```
elliot@ubuntu-linux:~$ ls -l soft.txt  
lrwxrwxrwx 1 tom tom 9 May 8 21:48 soft.txt -> facts.txt
```

Creating soft links

- Now let's check the contents of the file soft.txt:

```
elliott@ubuntu-linux:~$ cat soft.txt
```

Apples are red.

Grapes are green.

Bananas are yellow.

Cherries are red.

Sky is high.

Earth is round.

Linux is awesome!

Cherries are red.

Cherries are red.

Cherries are red.

- To demonstrate, open the file soft.txt with any text editor and add the line "Grass is green." at the very end of the file, and then save and exit so the contents of soft.txt will be as follows:

```
elliot@ubuntu-linux:~$ cat soft.txt
```

Apples are red.

Grapes are green.

Bananas are yellow.

Cherries are red.

Sky is high.

Earth is round.

Linux is awesome!

Cherries are red.

Cherries are red.

Cherries are red.

Grass is green.

Creating soft links

Creating soft links

- Now let's check the contents of the original file facts.txt:

```
elliot@ubuntu-linux:~$ cat facts.txt  
Apples are red.  
Grapes are green.  
Bananas are yellow.  
Cherries are red.  
Sky is high.  
Earth is round.  
Linux is awesome!  
Cherries are red.  
Cherries are red.  
Cherries are red.  
Grass is green.
```

- Now if you delete the soft link, nothing will happen to the original file, it remains intact:

```
elliott@ubuntu-linux:~$ rm soft.txt
```

```
elliott@ubuntu-linux:~$ cat facts.txt
```

Apples are red.

Grapes are green.

Bananas are yellow.

Cherries are red.

Sky is high.

Earth is round.

Linux is awesome!

Cherries are red.

Cherries are red.

Cherries are red.

Grass is green.

Creating soft links

Creating soft links

- Now let's create the soft link soft.txt again:

```
elliot@ubuntu-linux:~$ ln -s facts.txt soft.txt
```

- If you delete the original file facts.txt, the soft link soft.txt will become useless! But before we delete the facts.txt file

- let's make a copy of it in /tmp because we will need it later on:

```
elliot@ubuntu-linux:~$ cp facts.txt /tmp
```

Creating soft links

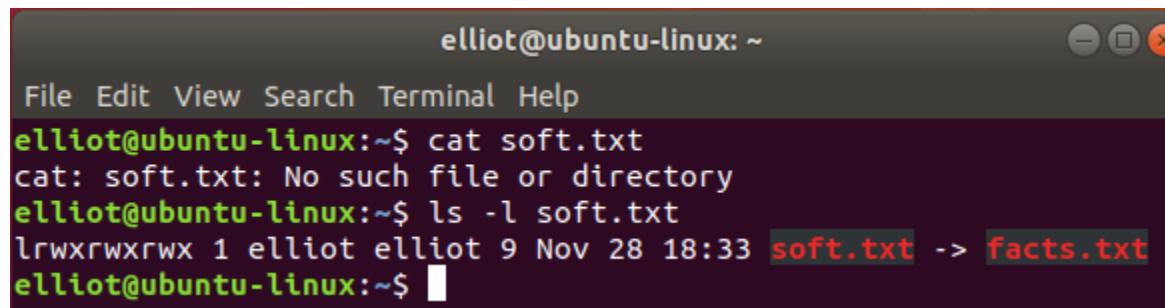
- Now let's delete the file facts.txt from elliot's home directory and see what happens to the soft link:

```
elliot@ubuntu-linux:~$ rm facts.txt
```

```
elliot@ubuntu-linux:~$ cat soft.txt
```

```
cat: soft.txt: No such file or directory
```

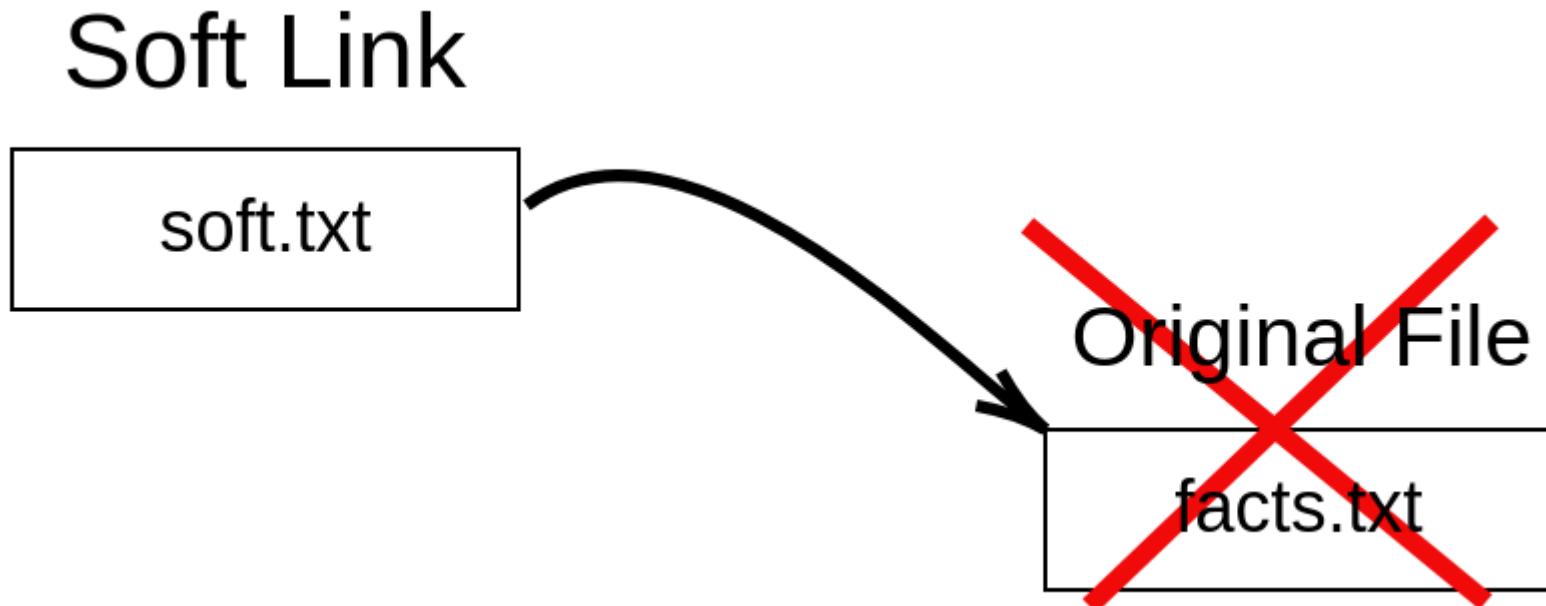
- As you can see, the soft link soft.txt becomes useless as it's now pointing to nowhere.
- Keep in mind that the file soft.txt still exists, as shown in the following screenshot.



```
elliot@ubuntu-linux: ~
File Edit View Search Terminal Help
elliot@ubuntu-linux:~$ cat soft.txt
cat: soft.txt: No such file or directory
elliot@ubuntu-linux:~$ ls -l soft.txt
lrwxrwxrwx 1 elliot elliot 9 Nov 28 18:33 soft.txt -> facts.txt
elliot@ubuntu-linux:~$
```

Creating soft links

The following diagram shows you that the soft link soft.txt points to nowhere after the original file facts.txt has been deleted.



- Now if we moved facts.txt back to elliot's home directory:
- elliot@ubuntu-linux:~\$ mv /tmp/facts.txt /home/elliot
- The soft link soft.txt will be useful again! You can say that we resurrected the soft link!

elliot@ubuntu-linux:~\$ cat soft.txt

Apples are red.

Grapes are green.

Bananas are yellow.

Cherries are red.

Sky is high.

Earth is round.

Linux is awesome!

Cherries are red.

Cherries are red.

Cherries are red.

Grass is green.

Creating soft links

- Let's compare the inode numbers of the soft link `soft.txt` and the original file `facts.txt`:

```
elliott@ubuntu-linux:~$ ls -i soft.txt facts.txt  
925155 facts.txt 924556 soft.txt
```

- As you can see, the inode numbers of the two files are different. Finally, let's run the `stat` command on the soft link `soft.txt`:

```
elliott@ubuntu-linux:~$ stat soft.txt  
File: soft.txt -> facts.txt  
Size: 9 Blocks: 0 IO Block: 4096 symbolic link  
Device: 801h/2049d Inode: 924556 Links: 1  
Access: (0777/lwxrwxrwx)Uid: ( 1000/ tom) Gid: ( 1000/ tom)  
Access: 2019-05-08 22:04:58.636000000 -0600  
Modify: 2019-05-08 22:02:18.356000000 -0600  
Change: 2019-05-08 22:02:18.356000000 -0600  
Birth: -
```

Creating soft links

Creating soft links

- You can create soft links to directories the same way you can create soft links to files.
- To demonstrate, let's first create a directory named sports in elliot's home directory. And inside sports, create three files – swimming, soccer, and hockey – as follows:

```
elliot@ubuntu-linux:~$ mkdir sports
```

```
elliot@ubuntu-linux:~$ touch sports/swimming sports/soccer sports/hockey
```

```
elliot@ubuntu-linux:~$ ls sports
```

```
hockey soccer swimming
```

- Now let's create a soft link named softdir1 to the sports directory:

```
elliot@ubuntu-linux:~$ ln -s sports softdir1
```

Creating soft links

- Now if you change to softdir1, you are actually changing to sports, and so you will see the same directory contents:

```
elliot@ubuntu-linux:~$ cd softdir1
```

```
elliot@ubuntu-linux:~/softdir1$ ls
```

```
hockey soccer swimming
```

Creating hard links

The following diagram helps you visualize hard links:

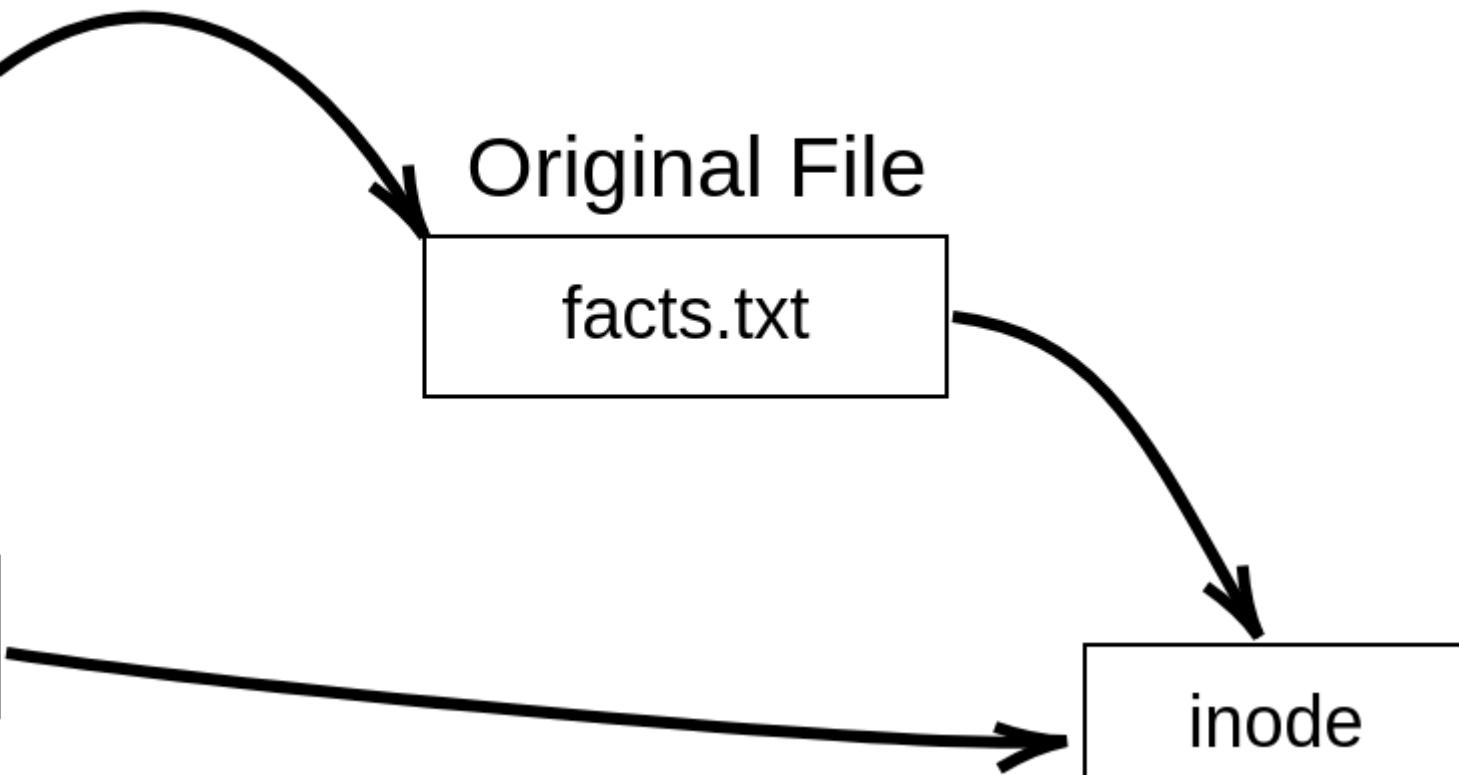
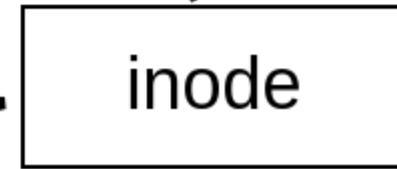
Soft Link



Original File



Hard Link



Creating hard links

- We use the same ln command to create hard links, but this time we omit the -s option:

ln original_file hard_link

- So to create a hard link named hard.txt to the file facts.txt, you can simply run the command ln facts.txt hard.txt:

```
elliott@ubuntu-linux:~$ ln facts.txt hard.txt
```

Creating hard links

- Now let's do a long listing on the hard link hard.txt and the original file facts.txt:

```
elliott@ubuntu-linux:~$ ls -l hard.txt  
-rw-rw-r-- 2 tom tom 210 May 9 00:07 hard.txt  
elliott@ubuntu-linux:~$ ls -l facts.txt  
-rw-rw-r-- 2 tom tom 210 May 9 00:07 facts.txt
```

Creating hard links

- They are identical!
- The hard link also has the same contents just like the original file:

```
elliott@ubuntu-linux:~$ cat hard.txt
Apples are red.
Grapes are green.
Bananas are yellow.
Cherries are red.
Sky is high.
Earth is round.
Linux is awesome!
Cherries are red.
Cherries are red.
Cherries are red.
Grass is green.
```

Creating hard links

- Now add the line "Swimming is a sport." to the very end of the hard link hard.txt with the text editor of your choice:

```
elliot@ubuntu-linux:~$ cat hard.txt
Apples are red.
Grapes are green.
Bananas are yellow.
Cherries are red.
Sky is high.
Earth is round.
Linux is awesome!
Cherries are red.
Cherries are red.
Cherries are red.
Grass is green.
Swimming is a sport.
```

Creating hard links

- Now just like in the case with soft links, the content of the original file has also changed:

```
elliot@ubuntu-linux:~$ cat facts.txt
Apples are red.
Grapes are green.
Bananas are yellow.
Cherries are red.
Sky is high.
Earth is round.
Linux is awesome!
Cherries are red.
Cherries are red.
Cherries are red.
Grass is green.
Swimming is a sport.
```

Creating hard links

- Now let's check the inode numbers of both files:

```
elliot@ubuntu-linux:~ ls -i hard.txt facts.txt  
925155 facts.txt 925155 hard.txt
```

Creating hard links

- Notice that both files have the same inode number.
- Now let's run the stat command on both files:

```
elliot@ubuntu-linux:~$ stat hard.txt facts.txt
```

File: hard.txt

Size: 210 Blocks: 8 IO Block: 4096 regular file

Device: 801h/2049d Inode: 925155 Links: 2

Access: (0664/-rw-rw-r--)Uid: (1000/ elliot) Gid: (1000/ elliot)

Access: 2019-05-09 00:07:36.884000000 -0600

Modify: 2019-05-09 00:07:25.708000000 -0600

Change: 2019-05-09 00:07:25.720000000 -0600

Birth: -

File: facts.txt

Size: 210 Blocks: 8 IO Block: 4096 regular file

Device: 801h/2049d Inode: 925155 Links: 2

Access: (0664/-rw-rw-r--)Uid: (1000/ elliot) Gid: (1000/ elliot)

Access: 2019-05-09 00:07:36.884000000 -0600

Modify: 2019-05-09 00:07:25.708000000 -0600

Change: 2019-05-09 00:07:25.720000000 -0600

Birth: -

- Now unlike the case with soft links, if you delete the original file facts.txt:

```
elliott@ubuntu-linux:~$ rm facts.txt
```

- The hard link remains intact:

```
elliott@ubuntu-linux:~$ cat hard.txt
```

Apples are red.

Grapes are green.

Bananas are yellow.

Cherries are red.

Sky is high.

Earth is round.

Linux is awesome!

Cherries are red.

Cherries are red.

Cherries are red.

Grass is green.

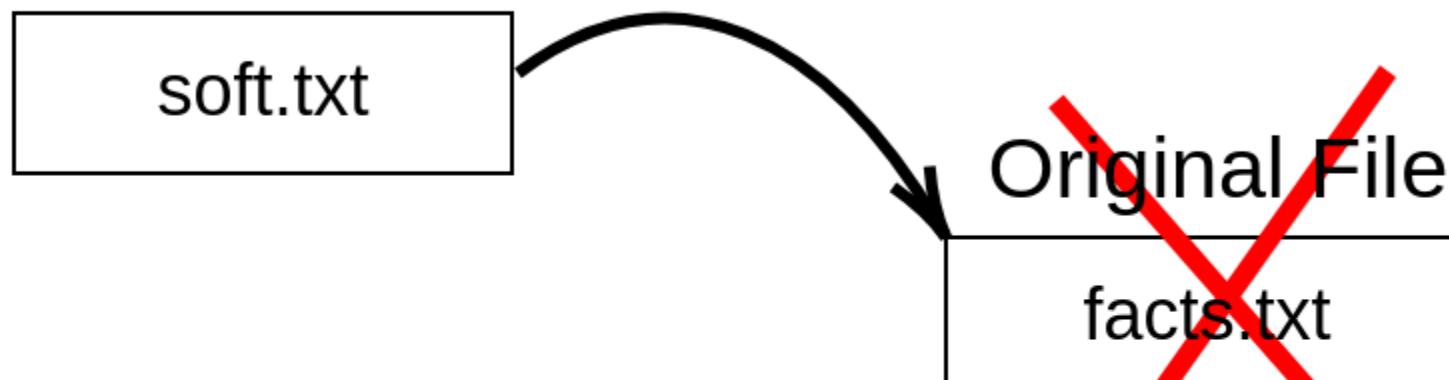
Swimming is a sport.

Creating hard links

Creating hard links

- The following diagram shows you why the hard link remains intact.

Soft Link



Hard Link



Creating hard links

- Now notice that after the removal of the file facts.txt, the number of hard links count of the file hard.txt will decrease to one:

```
elliot@ubuntu-linux:~$ stat hard.txt
```

```
File: hard.txt
```

```
Size: 210 Blocks: 8 IO Block: 4096 regular file
```

```
Device: 801h/2049d Inode: 925155 Links: 1
```

```
Access: (0664/-rw-rw-r--)Uid: ( 1000/ elliot) Gid: ( 1000/ elliot)
```

```
Access: 2019-05-09 00:17:21.176000000 -0600
```

```
Modify: 2019-05-09 00:07:25.708000000 -0600
```

```
Change: 2019-05-09 00:17:18.696000000 -0600
```

```
Birth: -
```

Creating hard links

- You can't create a hard link to a directory.
- If you don't believe me, then try creating a hard link named variables to the /var directory:

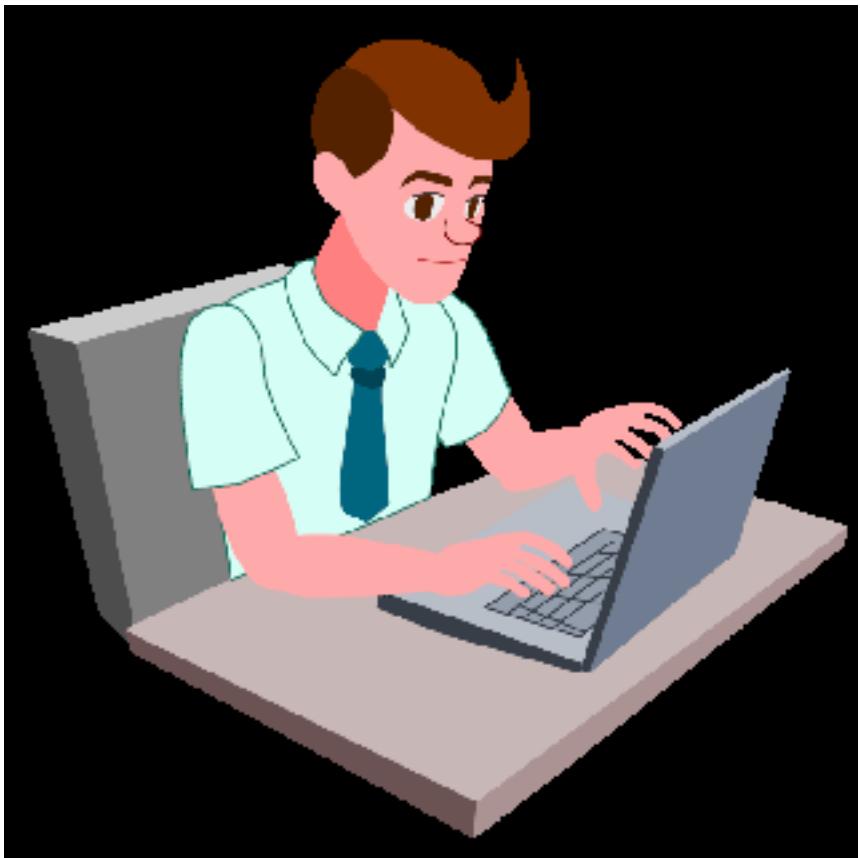
```
elliott@ubuntu-linux:~$ ln /var variables  
ln: /var: hard link not allowed for directory
```

I told you hard links are not allowed for directories! Why do you doubt me?

For the following exercises, open up your Terminal and try to solve the following tasks:

1. Display the inode number of the /var/log directory.
2. Display the number of hard links for the /boot directory.
3. Create a new directory named coins in your home directory.
4. Create a soft link to coins named currency.
5. Inside the coins directory, create two files – silver and gold.
6. Create a new file bronze inside currency.
7. List the contents of both directories – coins and currency.
8. Create a new file beverages with the line "coffee is awesome" in your home directory and create a hard link named drinks to beverages.
9. Add the line "lemon is refreshing" to the drinks file and then remove the beverages file.
10. Display the contents of your drinks file.

Knowledge check



"Complete Lab"

7. Superusers and the Root Login

A blurred background image of a person's hands typing on a laptop keyboard, positioned on the left side of the slide.

Superusers and the Root Login

- User elliot has been able to do quite a few things on the system.
- However, there are a whole lot of things that user elliot can't do!
- To demonstrate, let's try to create a file named happy in the /var directory:

```
elliot@ubuntu-linux:~$ touch /var/happy  
touch: cannot touch '/var/happy': Permission denied
```

- Oops! We got a Permission denied error.
- Now let's try to create a new directory named games in /etc:

```
elliot@ubuntu-linux:/$ mkdir /etc/games  
mkdir: cannot create directory '/etc/games': Permission denied
```

Accessing the root user

- You can run the sudo -i command to access the root user for the first time on your system:

```
elliott@ubuntu-linux:~$ sudo -i  
[sudo] password for elliot:  
root@ubuntu-linux:~#
```

Accessing the root user

- Let's run the whoami command to make sure that we are now logged in as the root user:

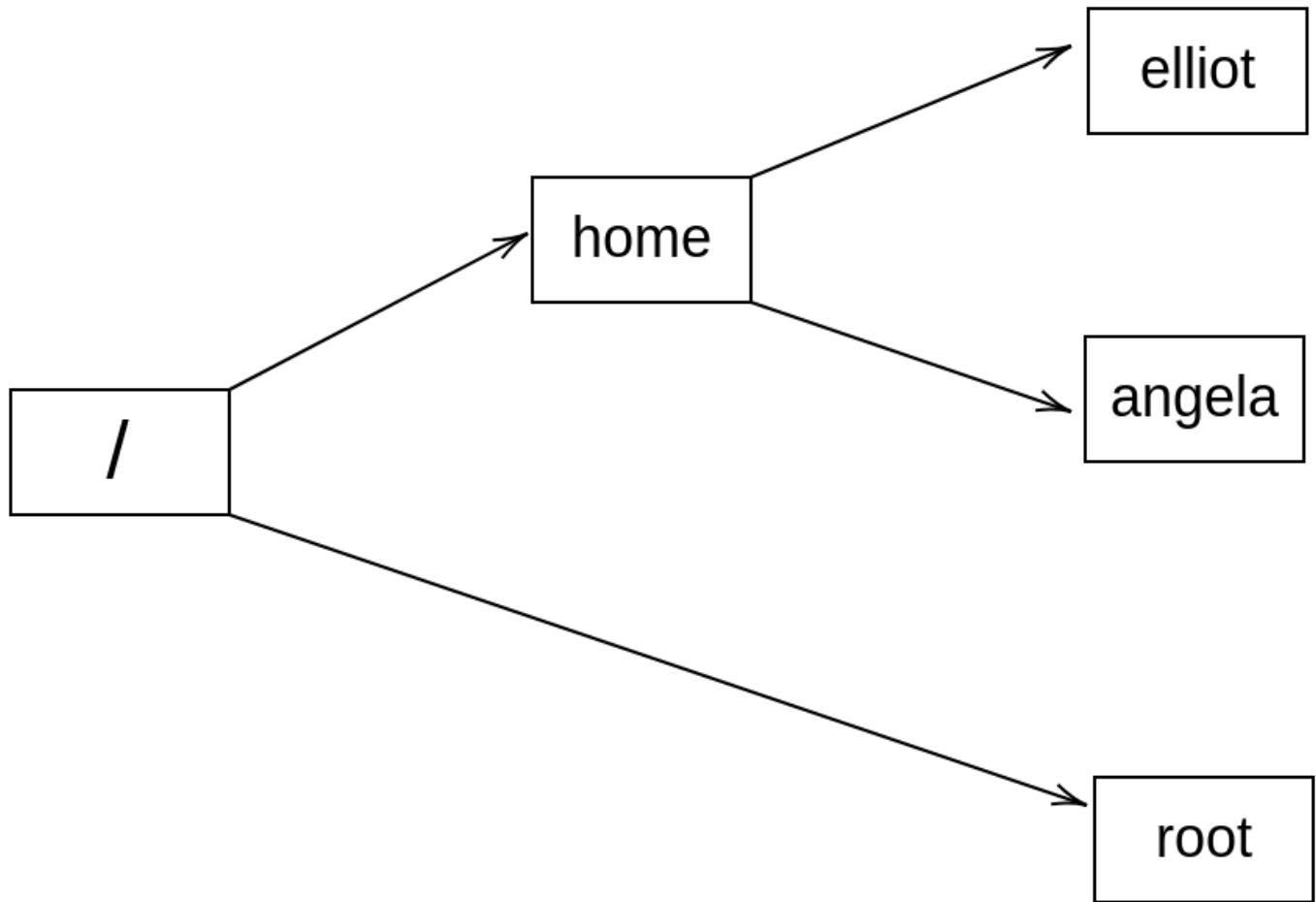
```
root@ubuntu-linux:~# whoami  
root
```

- Awesome! Now let's display the current working directory:

```
root@ubuntu-linux:~# pwd  
/root
```

Accessing the root user

- Remember earlier that I told you that the home directory for the root user is /root and not under /home.



Accessing the root user

- Now let's rerun both commands that we got permission denied for, but this time, we run both commands as the root user.

```
root@ubuntu-linux:~# touch /var/happy
```

```
root@ubuntu-linux:~# ls -l /var/happy
```

```
-rw-r--r-- 1 root root 0 Apr 15 10:53 /var/happy
```

- As you can see, nothing can stop the root user from doing anything!

Now let's create the directory games in /etc:

```
root@ubuntu-linux:~# mkdir /etc/games
```

```
root@ubuntu-linux:~# ls -ld /etc/games
```

```
drwxr-xr-x 2 root root 4096 Apr 15 10:55 /etc/games
```

Setting the root password

- You can also use the su command to switch to the root user but first, you need to set the root's password:

```
root@ubuntu-linux:~# passwd
```

```
Enter new UNIX password:
```

```
Retype new UNIX password:
```

```
passwd: password updated successfully
```

- Amazing, now exit the root user:

```
root@ubuntu-linux:~# exit
```

```
logout
```

```
elliot@ubuntu-linux:~$ whoami
```

```
elliot
```

Setting the root password

- Now you can use the su root command to switch to the root user:

```
elliot@ubuntu-linux:~$ su root
```

```
Password:
```

```
root@ubuntu-linux:/home/elliot# whoami  
root
```

The dash difference

- Notice that my current working directory is now /home/elliot and not /root.
- If I want to change that, I can exit back to user elliot and rerun the su command but this time, I will add a dash (hyphen) before root as follows:

```
root@ubuntu-linux:/home/elliot# exit  
exit  
elliott@ubuntu-linux:~$ su - root  
Password:  
root@ubuntu-linux:~# pwd  
/root
```

The dash difference

- If you want to switch to user elliot but preserve root's shell environment settings, then you don't need the dash:

```
root@ubuntu-linux:~# pwd  
/root  
root@ubuntu-linux:~# su elliot  
elliot@ubuntu-linux:/root$ pwd  
/root  
elliot@ubuntu-linux:/root$
```

The dash difference

- Notice how the current working directory didn't change when I switched to user elliot.
- Now, let's exit and switch back again to user elliot, but this time, we will put a dash before the username:

```
elliot@ubuntu-linux:/root$ exit
```

```
exit
```

```
root@ubuntu-linux:~# pwd
```

```
/root
```

```
root@ubuntu-linux:~# su - elliot
```

```
elliot@ubuntu-linux:~$ pwd
```

```
/home/elliot
```

The dash difference

- Let's try out our cool tip! As user elliot, run the su command without specifying a username:

```
elliot@ubuntu-linux:~$ su
```

Password:

```
root@ubuntu-linux:/home/elliot#
```

- You can then enter the root password to log in as root.
- You can also use the dash to acquire root's shell environment settings:

```
elliot@ubuntu-linux:~$ su -
```

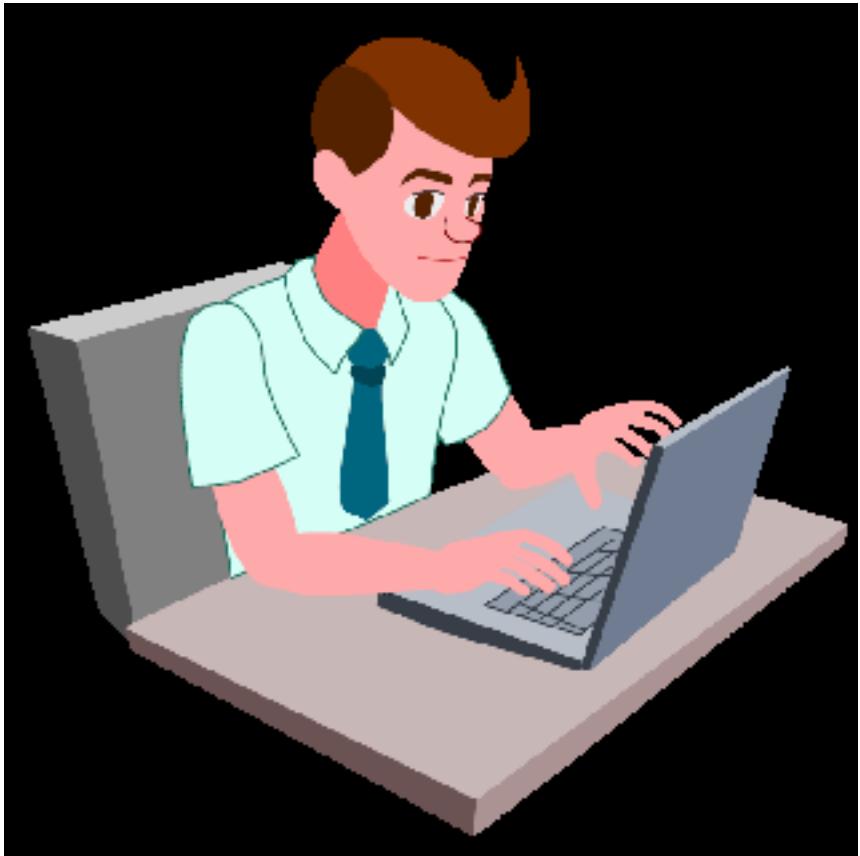
Password:

```
root@ubuntu-linux:~# pwd  
/root
```

Setting the root password

For the following exercises, open up your Terminal and try to solve the following tasks:

1. Switch to the root user.
2. Change the password for the root user.
3. Switch to user elliot and land in /home/elliot.
4. Now switch to user root but preserve the current working directory /home- /elliot.



"Complete Lab"

8. Controlling the Population

The /etc/passwd file

- In Linux, user information is stored in the /etc/passwd file. Every line in /etc/passwd corresponds to exactly one user.
 - Every line in /etc/passwd consists of 7 fields, each separated by a colon, and each field represents a user attribute.
 - For example, the entry for user elliot will look something like this:

```
1 2 3 4 5 6 7  
elliot:x:1000:1000:Elliot Alderson:/home/elliot:/bin/bash
```

Adding users

- Before you can add a user on your system, you have to become root:

```
elliott@ubuntu-linux:~$ su -
```

Password:

```
root@ubuntu-linux:~#
```

- Now, we are ready to add users. We all love Tom & Jerry, so let's begin by adding user tom.
- To do that, you need to run the command useradd -m tom:

```
root@ubuntu-linux:~# useradd -m tom
```

Adding users

- Just like that, the user tom is now added to our system.
- You will also see a new line added to the end of the /etc/passwd file for the new user tom; let's view it with the lovely tail command:

```
root@ubuntu-linux:~# tail -n 1 /etc/passwd  
tom:x:1007:1007::/home/tom:/bin/sh
```

- We used the -m option with the useradd command to ensure that a new home directory will be created for user tom.
- So let's try to change to the /home/tom directory to make sure it's indeed created:

```
root@ubuntu-linux:~# cd /home/tom  
root@ubuntu-linux:/home/tom# pwd  
/home/tom
```

Adding users

- The first thing you may want to do after creating a new user is to set the user's password.
- You can set tom's password by running the command passwd tom:

```
root@ubuntu-linux:~# passwd tom  
Enter new UNIX password:  
Retype new UNIX password:  
passwd: password updated successfully
```

Adding users

- Now, let's create user jerry, But this time, we will choose the following attributes for user jerry:

UID	777
Comment	Jerry the Mouse
Shell	/bin/bash

- This is easy to do with the useradd command:

```
root@ubuntu-linux:~# useradd -m -u 777 -c "Jerry the Mouse" -s  
/bin/bash jerry
```

Adding users

- Now, let's view the last two lines of the /etc/passwd file to make some comparisons:

```
root@ubuntu-linux:~# tail -n 2 /etc/passwd
tom:x:1007:1007::/home/tom:/bin/sh
jerry:x:777:1008:Jerry the Mouse:/home/jerry:/bin/bash
```

- Now, let's set the password for user jerry:

```
root@ubuntu-linux:~# passwd jerry
```

Enter new UNIX password:

Retype new UNIX password:

```
passwd: password updated successfully
```

- Amazing! We have now created two users: tom and jerry. Now, let's switch to user tom:

```
root@ubuntu-linux:~# su - tom
```

```
$ whoami tom
```

```
$ pwd
```

```
/home/tom
```

```
$
```

Adding users

- You can use the echo \$SHELL command to display the user's default shell:

```
$ echo $SHELL  
/bin/sh
```

- As you can see, it displayed /bin/sh. Now, let's exit and switch to user jerry:

```
$ exit  
root@ubuntu-linux:~# su - jerry  
jerry@ubuntu-linux:~$ whoami  
jerry  
jerry@ubuntu-linux:~$ echo $SHELL  
/bin/bash
```

Adding users

Adding users

- Everything looks better with user jerry as we did set his default shell to be /bin/bash.
- Alright, now let's switch back to the root user:

```
jerry@ubuntu-linux:~$ exit  
logout  
root@ubuntu-linux:~#
```

Modifying user attributes

- For example, to change the default shell for user tom to be /bin/bash, you can run the command usermod -s /bin/bash tom:

```
root@ubuntu-linux:~# usermod -s /bin/bash tom
```

- Notice that you can also use the full name for the command option; so you can use --shell instead of -s. Anyways, let's see if we successfully changed the default shell for user tom:

```
root@ubuntu-linux:~# su - tom
```

```
tom@ubuntu-linux:~$ whoami
```

```
tom
```

```
tom@ubuntu-linux:~$ echo $SHELL
```

```
/bin/bash
```

Modifying user attributes

- Great! We successfully did it. You can also change the UID of tom to 444 by running the command usermod -u 444 tom:

```
root@ubuntu-linux:~# usermod -u 444 tom
```

- And we can indeed check that the UID of tom has changed by taking a peek at the /etc/passwd file:

```
root@ubuntu-linux:~# tail -n 2 /etc/passwd
tom:x:444:1007::/home/tom:/bin/bash
jerry:x:777:1008:Jerry the Mouse:/home/jerry:/bin/bash
```

Modifying user attributes

- We can even modify the comment field of user tom.
- Right now, it's empty, but you can set the comment field of user tom to "Tom the Cat" by running the command:

```
root@ubuntu-linux:~# usermod --comment "Tom the Cat" tom
```

- And again, we can verify that the comment is changed by looking at the /etc/passwd file:

```
root@ubuntu-linux:~# tail -n 2 /etc/passwd
tom:x:444:1007:Tom the Cat:/home/tom:/bin/bash
jerry:x:777:1008:Jerry the Mouse:/home/jerry:/bin/bash
```

Defining the skeleton

- If you list the contents of /home/jerry and /home/tom, you will see that they are empty:

```
root@ubuntu-linux:~# ls -l /home/tom
```

```
total 0
```

```
root@ubuntu-linux:~# ls -l /home/jerry
```

```
total 0
```

- The reason that both /home/jerry and /home/tom are empty is that the skeleton file /etc/skel is also empty:

```
root@ubuntu-linux:~# ls -l /etc/skel
```

```
total 0
```

Defining the skeleton

- Now, with your favorite text editor, create the file welcome.txt in /etc/skel and insert the line "Hello Friend!" in it:

```
root@ubuntu-linux:/etc/skel# ls  
welcome.txt
```

```
root@ubuntu-linux:/etc/skel# cat welcome.txt  
Hello Friend!
```

- Alright, so now you have created the file welcome.txt in /etc/skel, which means that any new user created will now have the file welcome.txt in their home directory.
- To demonstrate, let's create a new user named edward and then we will take a peek at his home directory:

```
root@ubuntu-linux:~# useradd -m -c "Edward Snowden" -s /bin/bash  
edward
```

Defining the skeleton

- Now, let's set the password for user edward:

```
root@ubuntu-linux:~# passwd edward
```

Enter new UNIX password:

Retype new UNIX password:

```
passwd: password updated successfully
```

- Now, the moment of truth comes! Let's switch to user edward and list the contents of his home directory:

```
root@ubuntu-linux:~# su - edward
```

```
edward@ubuntu-linux:~$ ls
```

welcome.txt

```
edward@ubuntu-linux:~$ cat welcome.txt
```

Hello Friend!

Changing the defaults

- Open up the file /etc/default/useradd and look for the following line:
`SHELL=/bin/sh`
- Change it to:
`SHELL=/bin/bash`
- Awesome! Now, any new user created will have /bin/bash as the default shell.

Let's test it by creating a new user named spy:

```
root@ubuntu-linux:~# useradd -m spy
```

Changing the defaults

Now, set the password for user spy:

```
root@ubuntu-linux:~# passwd spy
```

```
Enter new UNIX password:
```

```
Retype new UNIX password:
```

```
passwd: password updated successfully
```

Finally, let's switch to user spy and check the default shell:

```
root@ubuntu-linux:~# su - spy
```

```
spy@ubuntu-linux:~$ echo $SHELL
```

```
/bin/bash
```

```
spy@ubuntu-linux:~$ exit
```

```
logout
```

```
root@ubuntu-linux:~#
```

Changing the defaults

- Keep in mind that /bin/sh and /bin/bash are not the only two valid shells on your system; there are more!
- Check out the file /etc/shells to see a complete list of all the valid shells on your system:

```
root@ubuntu-linux:~# cat /etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/bash
/bin/rbash
/bin/dash
```

Removing users

- The last user we created was spy, right? Well, we don't need any spies on our system, so let's delete the user spy; you can delete user spy by running the command userdel spy:

```
root@ubuntu-linux:~# userdel spy
```

- And just like that, user spy is deleted. However, the home directory of spy still exists:

```
root@ubuntu-linux:~# ls -ld /home/spy  
drwxr-xr-x 2 1008 1010 4096 Apr 17 10:24 /home/spy
```

- We would have to manually delete it:

```
root@ubuntu-linux:~# rm -r /home/spy
```

Removing users

- Let's give it a try with user edward:

```
root@ubuntu-linux:~# userdel -r edward
```

- Now, let's check to see if the home directory for user edward still exists:

```
root@ubuntu-linux:~# ls -ld /home/edward  
ls: cannot access '/home/edward': No such file or directory
```

And as you can see, edward's home directory is removed.

The /etc/group file

- All groups have their information stored in the file /etc/group. And just like with the /etc/passwd file, every line in /etc/group corresponds to exactly one group, and each line consists of 4 fields.
- For example, one of the most famous groups in Linux is the sudo group:

1	2	3	4
sudo	:	x	:27:elliot

Adding groups

- Let's create a group named cartoon.
- To do that, you need to run the command groupadd cartoon:

```
root@ubuntu-linux:~# groupadd cartoon
```

- Notice that a new line with the group information will be added to the end of the file /etc/group:

```
root@ubuntu-linux:~# tail -n 1 /etc/group  
cartoon:x:1009:
```

Adding groups

- Let's create another group named developers, but this time, we will specify a GID of 888:

```
root@ubuntu-linux:~# groupadd --gid 888 developers
```

- Let's check the developers group entry in /etc/group:

```
root@ubuntu-linux:~# tail -n 1 /etc/group
developers:x:888:
```

Adding group members

- To add tom to the cartoon group, you simply run the command usermod -aG cartoon tom:

```
root@ubuntu-linux:~# usermod -aG cartoon tom
```

- Likewise, you can add jerry to the cartoon group:

```
root@ubuntu-linux:~# usermod -aG cartoon jerry
```

- Now, let's have a peek at the /etc/group file:

```
root@ubuntu-linux:~# tail -n 2 /etc/group
```

```
cartoon:x:1009:tom,jerry
```

```
developers:x:888:
```

Adding group members

- You can use the id command to view the group memberships of any user on the system.
- For example, if you want to check which groups tom belongs to, you can run the command id tom:

```
root@ubuntu-linux:~# id tom
```

```
uid=444(tom) gid=1007(tom) groups=1007(tom),1009(cartoon)
```

- Let's do some more practice by creating three new users – sara, peter, and rachel:

```
root@ubuntu-linux:~# useradd -m sara
```

```
root@ubuntu-linux:~# useradd -m peter
```

```
root@ubuntu-linux:~# useradd -m rachel
```

Adding group members

- And remember to set the password for each user:

```
root@ubuntu-linux:~# passwd sara  
Enter new UNIX password:  
Retype new UNIX password:  
passwd: password updated successfully  
root@ubuntu-linux:~# passwd peter  
Enter new UNIX password:  
Retype new UNIX password:  
passwd: password updated successfully  
root@ubuntu-linux:~# passwd rachel  
Enter new UNIX password:  
Retype new UNIX password:  
passwd: password updated successfully  
root@ubuntu-linux:~#
```

Adding group members

- let's add all three users to the developers group:

```
root@ubuntu-linux:~# usermod -aG developers sara  
root@ubuntu-linux:~# usermod -aG developers peter  
root@ubuntu-linux:~# usermod -aG developers rachel
```

- Now, let's have a peek at the /etc/group file:

```
root@ubuntu-linux:~# tail -n 5 /etc/group  
cartoon:x:1009:tom,jerry  
developers:x:888:sara,peter,rachel  
sara:x:1001:  
peter:x:1002:  
rachel:x:1003:
```

Primary versus secondary groups

- Let's create a new user named dummy:

```
root@ubuntu-linux:~# useradd -m dummy
```

- Now, if you look at the last line of the /etc/group file, you will see that a group named dummy is also created:

```
root@ubuntu-linux:~# tail -n 1 /etc/group  
dummy:x:1004:
```

- This dummy group is the primary group of user dummy; and if you run the id command on user dummy:

```
root@ubuntu-linux:~# id dummy  
uid=1004(dummy) gid=1004(dummy) groups=1004(dummy)
```

Primary versus secondary groups

- You will see that user dummy is indeed a member of the dummy group.
- Now, let's add user dummy to the cartoon group:

```
root@ubuntu-linux:~# usermod -aG cartoon dummy
```

- Let's run the id command on user dummy again:

```
root@ubuntu-linux:~# id dummy
uid=1004(dummy) gid=1004(dummy)
groups=1004(dummy),1009(cartoon)
```

Primary versus secondary groups

- You can see that user dummy is a member of two groups: dummy and cartoon.
- However, dummy is the primary group and cartoon is the secondary group.
- The primary group is always preceded by gid= in the output of the id command:

```
root@ubuntu-linux:~# id dummy  
uid=1004(dummy) gid=1004(dummy) groups=1004(dummy),1009(cartoon)  
          Primary           Secondary
```

Primary versus secondary groups

- Now let's add user dummy to the developers group:

```
root@ubuntu-linux:~# usermod -aG developers dummy
```

- Next, run the id command on user dummy again:

```
root@ubuntu-linux:~# id dummy
uid=1004(dummy) gid=1004(dummy)
groups=1004(dummy),1009(cartoon),888(developers)
```

- As you can see, user dummy is a member of two secondary groups: cartoon and developers.
- Alright! Enough with all this dummy stuff, Let's remove the user dummy:

```
root@ubuntu-linux:~# userdel -r dummy
```

Primary versus secondary groups

- To demonstrate, let's create a user named smurf with cartoon being the primary group of user smurf.
- This can easily be done by using the --gid option with the useradd command:

```
root@ubuntu-linux:~# useradd -m --gid cartoon smurf
```

- Now, take a peek at the /etc/group file:

```
root@ubuntu-linux:~# tail -n 1 /etc/group
rachel:x:1003:
```

Primary versus secondary groups

- Now let's check user smurf's group memberships:

```
root@ubuntu-linux:~# id smurf  
uid=1004(smurf) gid=1009(cartoon) groups=1009(cartoon)
```

- As you can see, smurf is only a member of the group cartoon, which is also his primary group, of course.
- You can also change the primary group of existing users.
- For example, you can set the developers group to be the primary group of user smurf as follows:

```
root@ubuntu-linux:~# usermod -g developers smurf  
root@ubuntu-linux:~# id smurf  
uid=1004(smurf) gid=888(developers) groups=888(developers)
```

Removing groups

- You can remove a group if it is no longer needed.
- To demonstrate, let's create a group named temp:

```
root@ubuntu-linux:~# groupadd temp
```

- Now, you can use the groupdel command to remove the temp group:

```
root@ubuntu-linux:~# groupdel temp
```

- Now, let's try removing the group sara:

```
root@ubuntu-linux:~# groupdel sara
```

```
groupdel: cannot remove the primary group of user 'sara'
```

File ownership and permissions

- Every file in Linux is owned by a specific user and a specific group.
- To demonstrate, let's switch to user smurf, and create a file named mysmurf in smurf's home directory:

```
root@ubuntu-linux:~# su - smurf  
smurf@ubuntu-linux:~$ touch mysmurf
```

- Now do a long listing on the file mysmurf:

```
smurf@ubuntu-linux:~$ ls -l mysmurf  
-rw-r--r-- 1 smurf developers 0 Oct 22 15:09 mysmurf
```

↑ ↑
User Group

File ownership and permissions

- If you do a long listing on the sports directory that's inside elliot's home directory:

```
smurf@ubuntu-linux:~$ ls -ld /home/elliot/sports
drwxr-xr-x 2 elliot elliot 4096 Oct 22 12:56 /home/elliot/sports
```

Changing file ownership

- You can use the chown command to change a file's ownership.
- In general, the syntax of the chown command is as follows:

`chown user:group file`

- For example, you can change the ownership of the file mysmurf, so that user elliot is the owner, and group cartoon is the group owner, as follows:

`smurf@ubuntu-linux:~$`

`smurf@ubuntu-linux:~$ chown elliot:cartoon mysmurf`

`chown: changing ownership of 'mysmurf': Operation not permitted`

Changing file ownership

- Only the root user can do it; let's switch to the root user and try again:

```
smurf@ubuntu-linux:~$ su -
```

Password:

```
root@ubuntu-linux:~# cd /home/smurf
```

```
root@ubuntu-linux:/home/smurf# chown elliot:cartoon mysmurf
```

- Success! Now let's view the ownership of the file mysmurf:

```
root@ubuntu-linux:/home/smurf# ls -l mysmurf  
-rw-r--r-- 1 elliot cartoon 0 Oct 22 15:09 mysmurf
```

Changing file ownership

- For example, if you want the user root to be the owner of mysmurf, you can run the following command:

```
root@ubuntu-linux:/home/smurf# chown root mysmurf
root@ubuntu-linux:/home/smurf# ls -l mysmurf
-rw-r--r-- 1 root cartoon 0 Oct 22 15:09 mysmurf
```

- You can also change the group owner without changing the user owner.
- For example, if you want the group developers to be the group owner of mysmurf, then you can run:

```
root@ubuntu-linux:/home/smurf# chown :developers mysmurf
root@ubuntu-linux:/home/smurf# ls -l mysmurf
-rw-r--r-- 1 root developers 0 Oct 22 15:09 mysmurf
```

Understanding file permissions

- The meaning of each of these access permissions is not the same for files and directories.
- The following diagram explains the differences between access permissions for files versus directories:

File

Read: Be able to view the file contents.

Write: Be able to edit the file contents.

Execute: Be able to run the file (if executable).

Directory

Read: Be able to list the directory contents.

Write: Be able to create and remove files in the directory.

Execute: Be able to change to the directory.

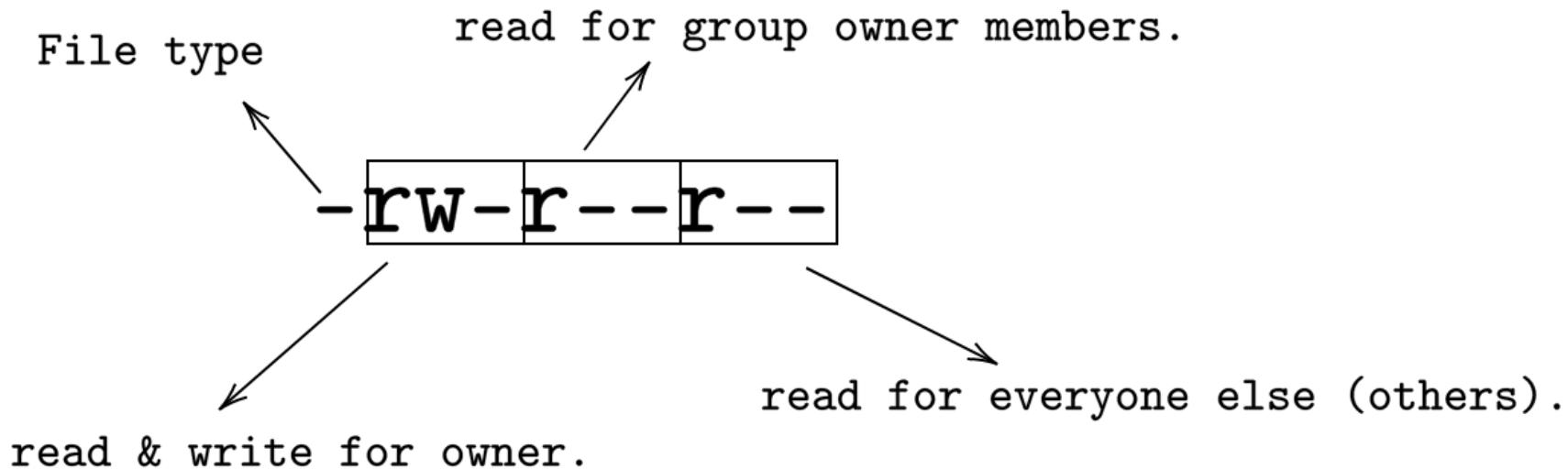
Understanding file permissions

- You can view the permissions of a file by doing a long listing.
- For example, to see the current permissions set on the mysmurf file, you can run:

```
root@ubuntu-linux:~# ls -l /home/smurf/mysmurf
```

```
-rw-r--r-- 1 root developers 0 Oct 22 15:09 /home/smurf/mysmurf
```

- The remaining nine slots are divided into three sets, each with three slots, just like in the following diagram:



Understanding file permissions

- Now let's get our hands dirty and do some examples to reinforce our understanding of file permissions.
- Let's first edit the mysmurf file and add the following line Smurfs are blue! so it looks like this:

```
root@ubuntu-linux:~# cat /home/smurf/mysmurf
Smurfs are blue!
```

- Now switch to user smurf and try reading the contents of the file mysmurf:

```
root@ubuntu-linux:~# su - smurf
smurf@ubuntu-linux:~$ cat mysmurf
Smurfs are blue!
```

Understanding file permissions

- Cool! User smurf can read the contents of the file mysmurf.
- Keep in mind that user smurf is not the owner of the file, but he is a member of the group developers:

```
smurf@ubuntu-linux:~$ id smurf  
uid=1004(smurf) gid=888(developers) groups=888(developers)
```

- So smurf can read the file because the group permission of mysmurf is r--. But can he edit the file? Let's see what will happen if user smurf tried to add the line I am smurf! to the file mysmurf:

```
smurf@ubuntu-linux:~$ echo "I am smurf!" >> mysmurf  
bash: mysmurf: Permission denied
```

Understanding file permissions

- Let's change the file ownership first:

```
smurf@ubuntu-linux:~$ su -
```

Password:

```
root@ubuntu-linux:~# chown smurf /home/smurf/mysmurf
```

```
root@ubuntu-linux:~# ls -l /home/smurf/mysmurf
```

```
-rw-r--r-- 1 smurf developers 17 Oct 23 11:06 /home/smurf/mysmurf
```

- Now let's switch back to user smurf and reattempt to edit the file mysmurf:

```
root@ubuntu-linux:~# su - smurf
```

```
smurf@ubuntu-linux:~$ echo "I am smurf!" >> mysmurf
```

```
smurf@ubuntu-linux:~$ cat mysmurf
```

Smurfs are blue!

I am smurf!

Understanding file permissions

- Cool! So user smurf has successfully edited the file.
- Now let's switch to user elliot and attempt to add the line I am not smurf! to the mysmurf file:

```
smurf@ubuntu-linux:~$ su - elliot
```

Password:

```
elliot@ubuntu-linux:~$ cd /home/smurf/
```

```
elliot@ubuntu-linux:/home/smurf$ echo "I am not smurf!" >> mysmurf
```

```
bash: mysmurf: Permission denied
```

- Permission denied! Notice that elliot is not the user owner and is not even a member of the developers group, so he is regarded as everyone else (others).
- However, he can read the file because others have read permission r--:

```
elliot@ubuntu-linux:/home/smurf$ cat mysmurf
```

Smurfs are blue!

I am smurf!

Changing file permissions

- Let's first switch to the root user:

```
elliott@ubuntu-linux:/home/smurf$ su -
```

Password:

```
root@ubuntu-linux:~# cd /home/smurf  
root@ubuntu-linux:/home/smurf#
```

- Now you can add the write permission for others (everyone else) by running the command:

```
root@ubuntu-linux:/home/smurf# chmod o+w mysmurf
```

- Here o+w means others+write, which means adding the write permission to others.
- Now do a long listing on mysmurf:

```
root@ubuntu-linux:/home/smurf# ls -l mysmurf  
-rw-r--rw- 1 smurf developers 29 Oct 23 11:34 mysmurf
```

- Now, switch back to user elliot and try to add the line I am not smurf! again:

```
root@ubuntu-linux:/home/smurf# su elliot  
elliot@ubuntu-linux:/home/smurf$ echo "I am not smurf!" >> mysmurf  
elliot@ubuntu-linux:/home/smurf$ cat mysmurf  
Smurfs are blue!  
I am smurf!  
I am not smurf!
```

Changing file permissions

Changing file permissions

- Success! User elliot can edit the file mysmurf. Now it's time to discuss the execute permission; let's go to elliot's home directory, and create a file named mydate.sh:

```
elliot@ubuntu-linux:/home/smurf$ cd /home/elliot  
elliot@ubuntu-linux:~$ touch mydate.sh
```

- Now add the following two lines to the file mydate.sh:

```
#!/bin/bash  
date
```

Changing file permissions

- You can add both lines by running the following two echo commands:

```
elliot@ubuntu-linux:~$ echo '#!/bin/bash' >> mydate.sh
```

```
elliot@ubuntu-linux:~$ echo date >> mydate.sh
```

- Do not worry about the meaning of the line '#!/bin/bash' now; I will explain it in a later lesson.
- Anyways, let's view the content of the file mydate.sh:

```
elliot@ubuntu-linux:~$ cat mydate.sh
```

```
#!/bin/bash
```

```
date
```

Changing file permissions

- Now do a long listing on the file mydate.sh:

```
elliott@ubuntu-linux:~$ ls -l mydate.sh  
-rw-rw-r-- 1 elliot elliot 17 Oct 23 12:28 mydate.sh
```

- Notice the absence of the execute permission here for everyone (the user owner, group owner, and others).
- Let's add the execute permission to everyone; you can do that by running the following command:

```
elliott@ubuntu-linux:~$ chmod a+x mydate.sh  
elliott@ubuntu-linux:~$ ls -l mydate.sh  
-rwxrwxr-x 1 elliot elliot 17 Oct 23 12:28 mydate.sh
```

Changing file permissions

- Finally, just enter the full path of mydate.sh and hit Enter:

```
elliott@ubuntu-linux:~$ /home/elliott/mydate.sh  
Wed Oct 23 12:38:51 CST 2019
```

Changing file permissions

- But now at least you know what it means for a file to be executable.
- Now remove the execute permission by running the command:

```
elliot@ubuntu-linux:~$ chmod a-x mydate.sh
```

```
elliot@ubuntu-linux:~$ ls -l mydate.sh
```

```
-rw-rw-r-- 1 elliot elliot 17 Oct 23 12:28 mydate.sh
```

- Here a-x means all-execute, which means remove the execute permission from everyone.
- Now try to run the script again:

```
elliot@ubuntu-linux:~$ /home/elliot/mydate.sh
```

```
bash: /home/elliot/mydate.sh: Permission denied
```

Directory permissions

- Now let's see how read, write, and execute permissions work on a directory.
- The easiest example will be the root's home directory /root, Let's do a long listing on /root:

```
root@ubuntu-linux:~# ls -ld /root  
drwx----- 5 root root 4096 Oct 22 14:28 /root
```

- As you can see, full permissions are given to the owner root and zero permissions for everyone else.
- Let's create a file inside /root named gold:

```
root@ubuntu-linux:~# touch /root/gold
```

Directory permissions

- Now let's switch to user smurf and try to list the contents of the /root directory:

```
root@ubuntu-linux:~# su - smurf
```

```
smurf@ubuntu-linux:~$ ls /root
```

```
ls: cannot open directory '/root': Permission denied
```

- User smurf gets a permission denied error as he's got no read permissions on the directory /root. Now, can smurf create a file inside /root?

```
smurf@ubuntu-linux:~$ touch /root/silver
```

```
touch: cannot touch '/root/silver': Permission denied
```

Directory permissions

- He cannot since he has no write permissions on /root. Can he delete a file inside /root?

```
smurf@ubuntu-linux:~$ rm /root/gold  
rm: cannot remove '/root/gold': Permission denied
```

- Again, no write permissions, so he can't delete a file in /root.
- Finally, can user smurf change to the /root directory?

```
smurf@ubuntu-linux:~$ cd /root  
-su: cd: /root: Permission denied
```

Directory permissions

- He cannot because smurf needs the execute permission to be able to change to the /root directory.
- Now, let's switch back to the root user and start adding some permissions:

```
smurf@ubuntu-linux:~$ exit
```

```
logout
```

```
root@ubuntu-linux:~# chmod o+rx /root
```

- Here, we added the read and execute permissions to others, so user smurf can now list the contents of the /root directory:

```
root@ubuntu-linux:~# su - smurf
```

```
smurf@ubuntu-linux:~$ ls /root  
gold
```

Directory permissions

- He can even change to the /root directory as we have added the execute permission as well:

```
smurf@ubuntu-linux:~$ cd /root
```

```
smurf@ubuntu-linux:/root$
```

- But he still has no write permissions, so he can't create or delete files in /root:

```
smurf@ubuntu-linux:/root$ rm gold
```

```
rm: remove write-protected regular empty file 'gold'? y
```

```
rm: cannot remove 'gold': Permission denied
```

```
smurf@ubuntu-linux:/root$ touch silver
```

```
touch: cannot touch 'silver': Permission denied
```

Directory permissions

- Let's add the write permission to others:

```
smurf@ubuntu-linux:/root$ su -
```

Password:

```
root@ubuntu-linux:~# chmod o+w /root
```

- Finally, switch to user smurf and try to create or remove a file in /root:

```
smurf@ubuntu-linux:~$ cd /root
```

```
smurf@ubuntu-linux:/root$ rm gold
```

```
rm: remove write-protected regular empty file 'gold'? y
```

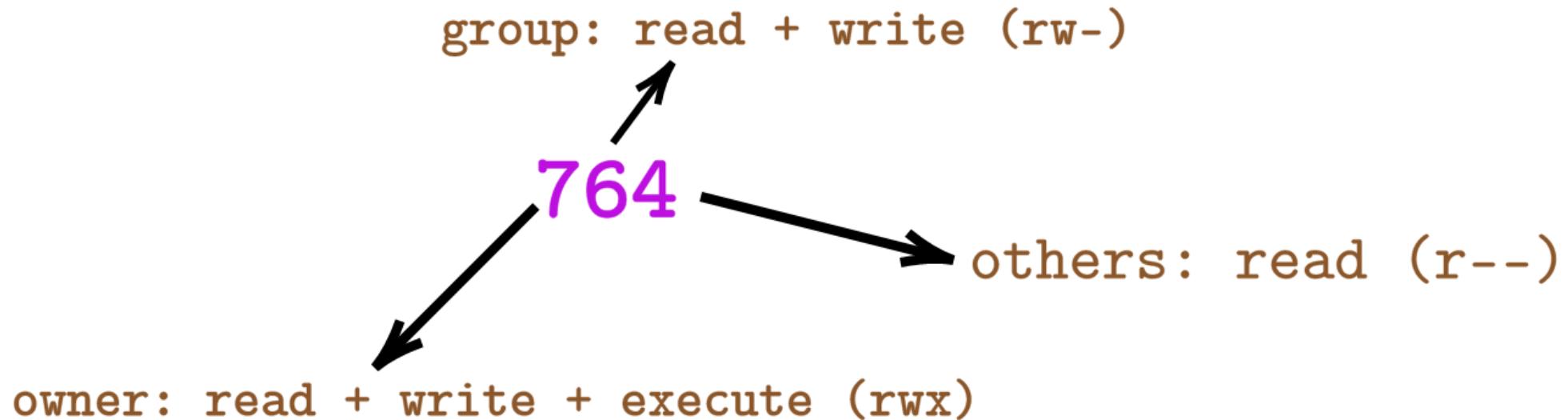
```
smurf@ubuntu-linux:/root$ touch silver
```

```
smurf@ubuntu-linux:/root$ ls
```

```
silver
```

Using octal notation

- Instead of the letters r, w, and x, you can use the numbers 4, 2, and 1 to set file permissions.
- Take a look at the following image:



Using octal notation

- Let's do some practice with the octal notation. There are currently zero permissions on the file mysmurf:

```
smurf@ubuntu-linux:~$ ls -l mysmurf  
----- 1 smurf developers 64 Oct 23 13:38 mysmurf
```

- We can use 777 to give full permissions to everyone:

```
smurf@ubuntu-linux:~$ chmod 777 mysmurf  
smurf@ubuntu-linux:~$ ls -l mysmurf  
-rwxrwxrwx 1 smurf developers 64 Oct 23 13:38 mysmurf
```

Using octal notation

- Cool! Now you can use the triplet 421 to give read permission for the owner, write permission for the group owner, and execute permission for others:

```
smurf@ubuntu-linux:~$ chmod 421 mysmurf
```

```
smurf@ubuntu-linux:~$ ls -l mysmurf
```

```
-r---w---x 1 smurf developers 64 Oct 23 13:38 mysmurf
```

- Let's do one more example. What if you want to give full permissions to the owner, read permission for the group owner, and zero permissions for others? That's easy; the correct triplet will be 740:

```
smurf@ubuntu-linux:~$ chmod 740 mysmurf
```

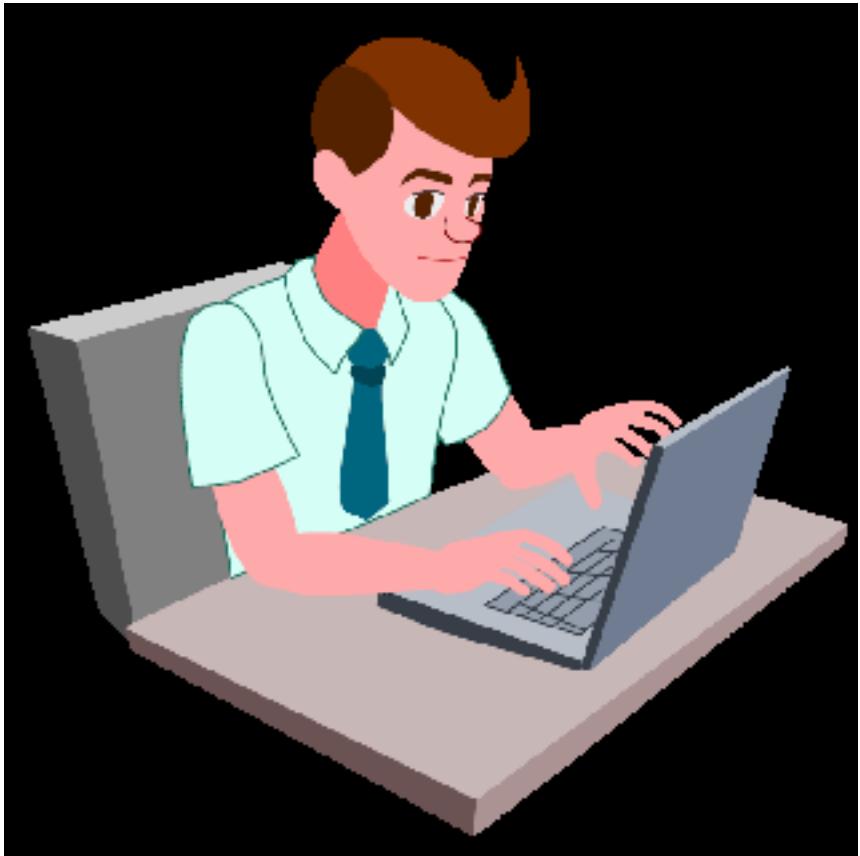
```
smurf@ubuntu-linux:~$ ls -l mysmurf
```

```
-rwxr----- 1 smurf developers 64 Oct 23 13:38 mysmurf
```

Knowledge check

For the following exercises, open up your Terminal and try to solve the following tasks:

1. Create a new user abraham with a user ID of 333.
2. Create a new group admins.
3. Add user abraham to the admins group.
4. Make admins the group owner of the directory /home/abraham.
5. Members of the admins group can only list the contents of the directory /home/abraham.



"Complete Lab"

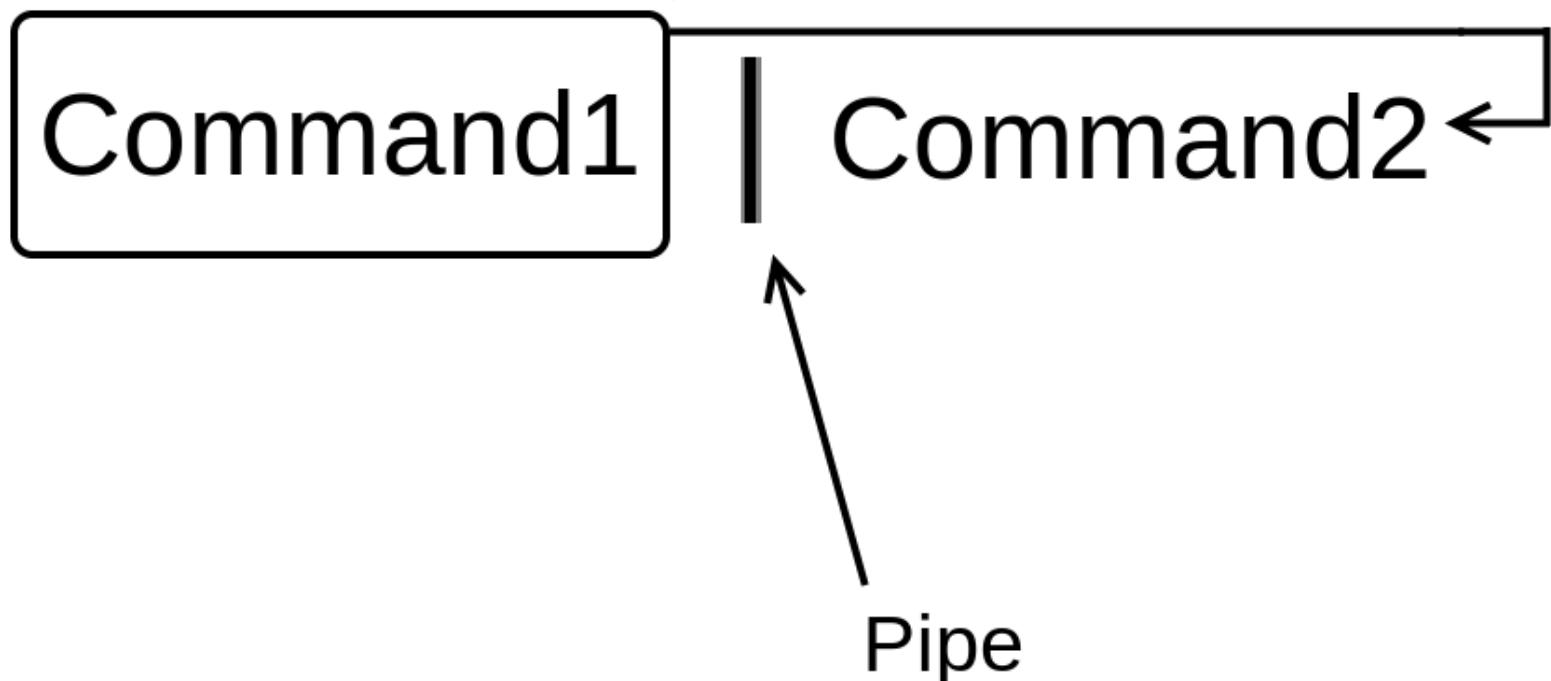
9. Piping and I/O Redirection



Linux pipes

- In Linux, you can use a pipe to send the output of one command to be the input (argument) of another command:

output of Command1



Linux pipes

- Before we do an example, let's first rename the hard.txt file to facts.txt, as we removed the facts.txt file back in lesson 6, Hard vs. Soft Links:

```
elliott@ubuntu-linux:~$ mv hard.txt facts.txt
```

- Now let's use the head command to view the first five lines of facts.txt:

```
elliott@ubuntu-linux:~$ head -n 5 facts.txt
```

Apples are red.

Grapes are green.

Bananas are yellow.

Cherries are red.

Sky is high.

Linux pipes

- That's where the power of Linux pipes comes into play.
- If you pipe the output of the previous command to the tail -n 1 command, you will get the fifth line:

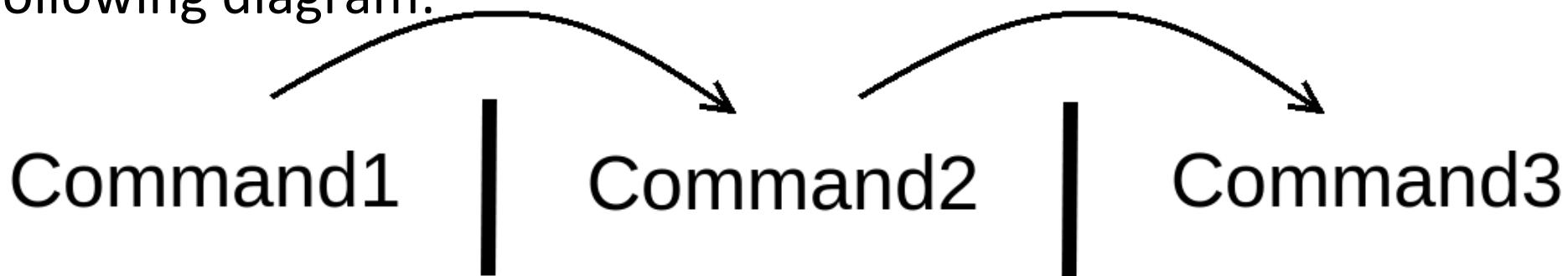
```
elliott@ubuntu-linux:~$ head -n 5 facts.txt | tail -n 1  
Sky is high.
```

- Let's do another example.
- If you want to display the seventh line of the file facts.txt, then you will show the first seven lines using the head command, then use a pipe to tail the last line:

```
elliott@ubuntu-linux:~$ head -n 7 facts.txt | tail -n 1  
Linux is awesome
```

Linux pipes

- You can also use more than one pipe at a time as demonstrated in the following diagram:



- For example, you already know that the lscpu command displays your processor information.
- The fourth line of the lscpu command output shows how many CPUs your machine has.
- You can display the fourth line of the lscpu command by using two pipes:

```
elliot@ubuntu-linux:~$ lscpu | head -n 4 | tail -n 1
```

CPU(s): 1

Linux pipes

- let's break down what happened here. The first pipe we used was to show the first four lines of the lscpu command:

```
elliot@ubuntu-linux:~$ lscpu | head -n 4
```

Architecture: x86_64

CPU op-mode(s): 32-bit, 64-bit

Byte Order: Little Endian

CPU(s): 1

- We then used the second pipe to tail the last line, which gets us the fourth line in this case:

```
elliot@ubuntu-linux:~$ lscpu | head -n 4 | tail -n 1
```

CPU(s): 1

Redirecting standard output

- You know that running the date command will display the current date on your terminal:

```
elliot@ubuntu-linux:~$ date  
Sat May 11 06:02:44 CST 2019
```

- Now by using the greater than sign >, you can redirect the output of the date command to a file instead of your terminal! Have a look:

```
elliot@ubuntu-linux:~$ date > mydate.txt
```

Redirecting standard output

- As you can see, there is no output displayed on your screen! That's because the output got redirected to the file mydate.txt:

```
elliot@ubuntu-linux:~$ cat mydate.txt
```

```
Sat May 11 06:04:49 CST 2019
```

- Cool! Let's try some more examples. You can print a line on your terminal with the echo command:

```
elliot@ubuntu-linux:~$ echo "Mars is a planet."
```

```
Mars is a planet.
```

- If you want to redirect the output to a file named planets.txt, you can run the command:

```
elliot@ubuntu-linux:~$ echo "Mars is a planet." > planets.txt
```

```
elliot@ubuntu-linux:~$ cat planets.txt
```

```
Mars is a planet
```

Redirecting standard output

- Awesome! Notice that the file planets.txt was also created in the process.
- Now let's add more planets to the file planets.txt:

```
elliott@ubuntu-linux:~$ echo "Saturn is a planet." > planets.txt
```

```
elliott@ubuntu-linux:~$ cat planets.txt
```

Saturn is a planet.

- So now let's append the line "Mars is a planet." back to the file planets.txt:

```
elliott@ubuntu-linux:~$ echo "Mars is a planet." >> planets.txt
```

```
elliott@ubuntu-linux:~$ cat planets.txt
```

Saturn is a planet.

Mars is a planet.

Redirecting standard output

- Great! As you can see, it added the line "Mars is a planet." to the end of the file.
- Let's append one more planet:

```
elliot@ubuntu-linux:~$ echo "Venus is a planet." >> planets.txt
```

```
elliot@ubuntu-linux:~$ cat planets.txt
```

Saturn is a planet.

Mars is a planet.

Venus is a planet.

Redirecting standard output

- So running the command:

```
elliot@ubuntu-linux:~$ date > mydate.txt
```

- Is the same as running the command:

```
elliot@ubuntu-linux:~$ date 1> mydate.txt
```

Redirecting standard error

- You will get an error message if you try to display the contents of a file that doesn't exist:

```
elliot@ubuntu-linux:~$ cat blabla  
cat: blabla: No such file or directory
```

- Now, this error message comes from standard error (stderr).
- If you try to redirect errors the same way we did with the standard output, it will not work:

```
elliot@ubuntu-linux:~$ cat blabla > error.txt  
cat: blabla: No such file or directory
```

Redirecting standard error

- It still displays the error message on your terminal.
- That's because stderr is linked to file descriptor 2. And thus, to redirect errors, you have to use 2>:

```
elliot@ubuntu-linux:~$ cat blabla 2> error.txt
```

- Now if you displayed the contents of the file error.txt, you would see the error message:

```
elliot@ubuntu-linux:~$ cat error.txt  
cat: blabla: No such file or directory
```

- Let's try to remove a file that doesn't exist:

```
elliot@ubuntu-linux:~$ rm brrrr  
rm: cannot remove 'brrrr': No such file or directory
```

Redirecting standard error

- Also produces an error message.
- We can append this error message to the file `error.txt` using `2>>`:

```
elliot@ubuntu-linux:~$ rm brrrr 2>> error.txt
```

- Now if you display the contents of the file `error.txt`:

```
elliot@ubuntu-linux:~$ cat error.txt
```

```
cat: blabla: No such file or directory
```

```
rm: cannot remove 'brrrr': No such file or directory
```

Redirecting all output to the same file

- There are some situations where you can get both standard output and an error message at the same time.
- For example, if you run the following command:

```
elliott@ubuntu-linux:~$ cat planets.txt blabla
```

```
Saturn is a planet.
```

```
Mars is a planet.
```

```
Venus is a planet.
```

```
cat: blabla: No such file or directory
```

Redirecting all output to the same file

- You can choose to redirect the error to another file:

```
elliott@ubuntu-linux:~$ cat planets.txt blabla 2> err.txt
```

Saturn is a planet.

Mars is a planet.

Venus is a planet.

- This way, you only see the standard output on the screen.

- Or you may choose to redirect the standard output:

```
elliott@ubuntu-linux:~$ cat planets.txt blabla 1> output.txt
```

cat: blabla: No such file or directory

Redirecting all output to the same file

- This way, you only see the error on the screen.
- Now, what if you want to redirect both the standard output and the error to the same file? In this case, you have to run:

```
elliot@ubuntu-linux:~$ cat planets.txt blabla > all.txt 2>&1
```

- what we are basically saying here is: "Redirect the stderr to the same place we are redirecting the stdout."
- Now if you displayed the contents of the file all.txt:

```
elliot@ubuntu-linux:~$ cat all.txt
```

Saturn is a planet.

Mars is a planet.

Venus is a planet.

```
cat: blabla: No such file or directory
```

Discarding output

- Sometimes you don't need to redirect output to anywhere; you just want to throw it away and get rid of it.
- In this case, you can redirect the output to /dev/null, This is often used with error messages
- For example:

```
elliott@ubuntu-linux:~$ cat planets.txt blabla 2> /dev/null
Saturn is a planet.
Mars is a planet.
Venus is a planet.
```

Redirecting standard input

- Some Linux commands interact with the user input through the standard input (which is your keyboard by default).
- For example, the read command reads input from the user and stores it in a variable.
- For example, you can run the command read weather:

```
elliot@ubuntu-linux:~$ read weather  
It is raining.
```

- It will then wait for you to enter a line of text. I entered the line It is raining and so it stored the line in the weather variable.

- You can use the echo command to display the contents of a variable:

```
elliot@ubuntu-linux:~$ echo $weather  
It is raining.
```

Redirecting standard input

- I can redirect standard input to come from a file instead using the less-than sign <, for example:

```
elliot@ubuntu-linux:~$ read message < mydate.txt
```

- This will read the contents of the file mydate.txt and store it in the message variable:

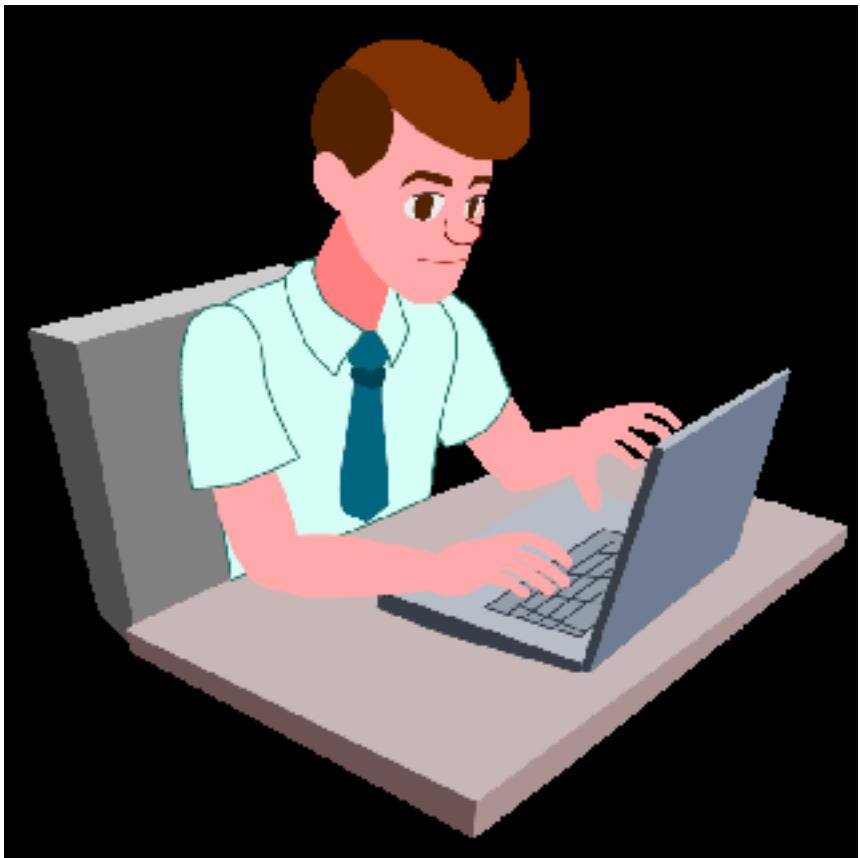
```
elliot@ubuntu-linux:~$ echo $message
```

```
Sat May 11 06:34:52 CST 2019
```

Knowledge check

For the following exercises, open up your terminal and try to solve the following tasks:

1. Display only the 5th line of the file facts.txt.
2. Save the output of the free command into a file named system.txt.
3. Append the output of the lscpu command to the file system.txt.
4. Run the command rmdir /var and redirect the error message to the file error.txt.



"Complete Lab"

10. Analyzing and Manipulating Files

Spot the difference

- You can use the diff command to compare the contents of two files and highlight the differences between them.
- To demonstrate, let's first make a copy of the file facts.txt named facts2.txt:

```
elliot@ubuntu-linux:~$ cp facts.txt facts2.txt
```

- Now let's append the line "Brazil is a country." to the file facts2.txt:

```
elliot@ubuntu-linux:~$ echo "Brazil is a country." >> facts2.txt
```

Spot the difference

- Now, run the diff command on both files:

```
elliott@ubuntu-linux:~$ diff facts.txt facts2.txt
12a13
> Brazil is a country.
```

Viewing file size

- You can use the du command to view file size. du stands for disk usage.
- If you want to see how many bytes are in a file, you can run the du command with the -b option:

```
elliot@ubuntu-linux:~$ du -b facts.txt  
210 facts.txt
```

- You can also use the -h option, which will print the file size in a human-readable format.
- For example, to view the size of the dir1 directory and its contents, you can run:

```
elliot@ubuntu-linux:~$ du -h dir1  
4.0K  dir1/cities  
16K   dir1/directory2  
24K   dir1
```

Counting characters, words, and lines

- The word count wc command is yet another very handy command. It counts the number of lines, words, and characters in a file.
- For example, to display the number of lines in the file facts.txt, you can use the -l option:

```
elliot@ubuntu-linux:~$ wc -l facts.txt  
12 facts.txt
```

- There are a total of 12 lines in the file facts.txt.
- To display the number of words, you can use the -w option:

```
elliot@ubuntu-linux:~$ wc -w facts.txt  
37 facts.txt
```

Counting characters, words, and lines

- There is a total of 37 words in the file facts.txt.
- To display the number of characters (bytes), you can use the -c option:

```
elliot@ubuntu-linux:~$ wc -c facts.txt  
210 facts.txt
```

- There is a total of 210 characters in the file facts.txt.
- Without any options, the wc command will display the number of lines, words, and characters side by side:

```
elliot@ubuntu-linux:~$ wc facts.txt  
12 37 210 facts.txt
```

Viewing the file type

- You can determine a file's type by using the file command.
- For example, if you want to determine the type of the file /var, you can run:

```
elliott@ubuntu-linux:~$ file /var  
/var: directory
```

- And as you would expect, the output shows that /var is a directory.
- If you want to show the type of the facts.txt file, you can run:

```
elliott@ubuntu-linux:~$ file facts.txt  
facts.txt: ASCII text
```

Viewing the file type

- Now let's create a soft link named soft.txt to the facts.txt file:

```
elliot@ubuntu-linux:~$ ln -s soft.txt facts.txt
```

- And run the file command on soft.txt:

```
elliot@ubuntu-linux:~$ file soft.txt  
soft.txt: symbolic link to facts.txt
```

Sorting files

- You can use the sort command to sort text files.
- For example, you can view the facts.txt file in sorted alphabetical order by running the command:

```
elliott@ubuntu-linux:~$ sort facts.txt
Apples are red.
Bananas are yellow.
Cherries are red.
Cherries are red.
Cherries are red.
Cherries are red.
Earth is round.
Grapes are green.
Grass is green.
Linux is awesome!
Sky is high.
Swimming is a sport.
```

Sorting files

- You can also use the `-r` option to sort in reverse order:

```
elliott@ubuntu-linux:~$ sort -r facts.txt
Swimming is a sport.
Sky is high.
Linux is awesome!
Grass is green.
Grapes are green.
Earth is round.
Cherries are red.
Cherries are red.
Cherries are red.
Cherries are red.
Bananas are yellow.
Apples are red.
```

Showing unique lines

- To view facts.txt without repeated lines, you can run:

```
elliott@ubuntu-linux:~$ uniq facts.txt
Apples are red.
Grapes are green.
Bananas are yellow.
Cherries are red.
Sky is high.
Earth is round.
Linux is awesome!
Cherries are red.
Grass is green.
Swimming is a sport.
```

Showing unique lines

- Notice that Cherries are red. is still shown twice in the output.
- That's because the uniq command only omits repeated lines but not duplicates! If you want to omit duplicates, you have to sort the file first and then use a pipe to apply the uniq command on the sorted output:

```
elliott@ubuntu-linux:~$ sort facts.txt | uniq  
Apples are red.  
Bananas are yellow.  
Cherries are red.  
Earth is round.  
Grapes are green.  
Grass is green.  
Linux is awesome!  
Sky is high.  
Swimming is a sport.
```

Searching for patterns

- The grep command is one of the most popular and useful commands in Linux.
- You can use grep to print the lines of text that match a specific pattern.
- For example, if you want to only display the lines that contain the word green in facts.txt, you can run:

```
elliot@ubuntu-linux:~$ grep green facts.txt  
Grapes are green.  
Grass is green.
```

Searching for patterns

- The grep command can also be very useful when used with pipes.
- For example, to only list the txt files in your home directory, you can run the command:

```
elliott@ubuntu-linux:~$ ls | grep txt
all.txt
error.txt
facts2.txt
facts.txt
Mars.txt
mydate.txt
output.txt
planets.txt
soft.txt
```

Searching for patterns

- You can use the `-i` option to make your search case-insensitive.
- For example, if you want to print the lines that contain the word Earth in `facts.txt`, then use the command:

```
elliot@ubuntu-linux:~$ grep earth facts.txt
```

```
elliot@ubuntu-linux:~$
```

- This will show no result because `grep` is case-sensitive by default. However, if you pass the `-i` option:

```
elliot@ubuntu-linux:~$ grep -i earth facts.txt
```

```
Earth is round.
```

The stream editor

- You can use the stream editor command sed to filter and transform text.
- For example, to substitute the word Sky with the word Cloud in facts.txt, you can run the command:

```
elliot@ubuntu-linux:~$ sed 's/Sky/Cloud/' facts.txt
Apples are red.
Grapes are green.
Bananas are yellow.
Cherries are red.
Cloud is high.
Earth is round.
Linux is awesome!
Cherries are red.
Cherries are red.
Cherries are red.
Grass is green.
Swimming is a sport.
```

The stream editor

- As you can see in the output, the word Sky is replaced with Cloud.
- However, the file facts.txt is not edited.
- To overwrite (edit) the file, you can use the -i option:

```
elliott@ubuntu-linux:~$ sed -i 's/Sky/Cloud/' facts.txt
elliott@ubuntu-linux:~$ cat facts.txt
Apples are red.
Grapes are green.
Bananas are yellow.
Cherries are red.
Cloud is high.
Earth is round.
Linux is awesome!
Cherries are red.
Cherries are red.
Cherries are red.
Grass is green.
Swimming is a sport.
```

Translating characters

```
elliot@ubuntu-linux:~$ cat facts.txt | tr [:lower:] [:upper:]
```

APPLES ARE RED.

GRAPES ARE GREEN.

BANANAS ARE YELLOW.

CHERRIES ARE RED.

CLOUD IS HIGH.

EARTH IS ROUND.

LINUX IS AWESOME!

CHERRIES ARE RED.

CHERRIES ARE RED.

CHERRIES ARE RED.

GRASS IS GREEN.

SWIMMING IS A SPORT.

- One popular use of the `tr` command is to change lower case letters to upper case (or vice versa).
- For example, if you want to display all the words in `facts.txt` in upper case, you can run:

Translating characters

- You can also display all the words in lower case:

```
elliott@ubuntu-linux:~$ cat facts.txt | tr [:upper:] [:lower:]  
apples are red.  
grapes are green.  
bananas are yellow.  
cherries are red.  
cloud is high.  
earth is round.  
linux is awesome!  
cherries are red.  
cherries are red.  
cherries are red.  
grass is green.  
swimming is a sport.
```

Translating characters

- You can also use the `-d` option to delete characters.
- For example, to remove all spaces in `facts.txt`, you can run:

```
elliott@ubuntu-linux:~$ cat facts.txt | tr -d ''  
Applesarerered.  
Grapesaregreen.  
Bananasareyellow.  
Cherriesarerered.  
Cloudishigh.  
Earthisround.  
Linuxisawesome!  
Cherriesarerered.  
Cherriesarerered.  
Cherriesarerered.  
Grassisgreen.  
Swimmingisasport.
```

Translating characters

- For example, running the command:

```
elliot@ubuntu-linux:~$ cat facts.txt | tr [:lower:] [:upper:] > upper.txt
```

- will store the output of the command:

```
cat facts.txt | tr [:lower:] [:upper:]
```

Cutting text

- If you only want to view the first word in each line (first column/field), then you can run the following command:

```
elliot@ubuntu-linux:~$ cut -d ' ' -f1 facts.txt
Apples
Grapes
Bananas
Cherries
Cloud
Earth
Linux
Cherries
Cherries
Cherries
Grass
Swimming
```

Cutting text

- If you want to view the third word of each line (third field), then you can use -f3 instead of -f1 as follows:

```
elliott@ubuntu-linux:~$ cut -d ' ' -f3 facts.txt
red.
green.
yellow.
red.
high.
round.
awesome!
red.
red.
red.
green.
a
```

Cutting text

- You can also select more than one field at a time.
- For example, to view the first and the third word of each line, you can use -f1,3:

```
elliott@ubuntu-linux:~$ cut -d ' ' -f1,3 facts.txt  
Apples red.  
Grapes green.  
Bananas yellow.  
Cherries red.  
Cloud high.  
Earth round.  
Linux awesome!  
Cherries red.  
Cherries red.  
Cherries red.  
Grass green.  
Swimming a
```

Text processing with awk

- You can use awk to achieve the same functionality as the cut command.
- For example, to view the first word of each line in the file facts.txt, you can run:

```
elliott@ubuntu-linux:~$ awk '{print $1}' facts.txt
Apples
Grapes
Bananas
Cherries
Cloud
Earth
Linux
Cherries
Cherries
Cherries
Grass
Swimming
```

Text processing with awk

- You can also view more than one field at a time; for example, to view the first and the second word of each line, you can run:

```
elliott@ubuntu-linux:~$ awk '{print $1,$2}' facts.txt
Apples are
Grapes are
Bananas are
Cherries are
Cloud is
Earth is
Linux is
Cherries are
Cherries are
Cherries are
Grass is
Swimming is
```

Text processing with awk

- To demonstrate, create a file named animals.txt and insert these four lines:

fox is smart

whale is big

cheetah is fast

penguin is cute

- Do not edit the format; keep the spaces messed up:

elliot@ubuntu-linux:~\$ cat animals.txt

fox is smart

whale is big

cheetah is fast

penguin is cute

- However, awk is smart enough to figure it out:

```
elliott@ubuntu-linux:~$ awk '{print $3}' animals.txt
```

smart

big

fast

cute

- For example, to print the lines that contain the word red in facts.txt, you can run the command:

```
elliott@ubuntu-linux:~$ awk '/red/{print}' facts.txt
```

Apples are red.

Cherries are red.

Cherries are red.

Cherries are red.

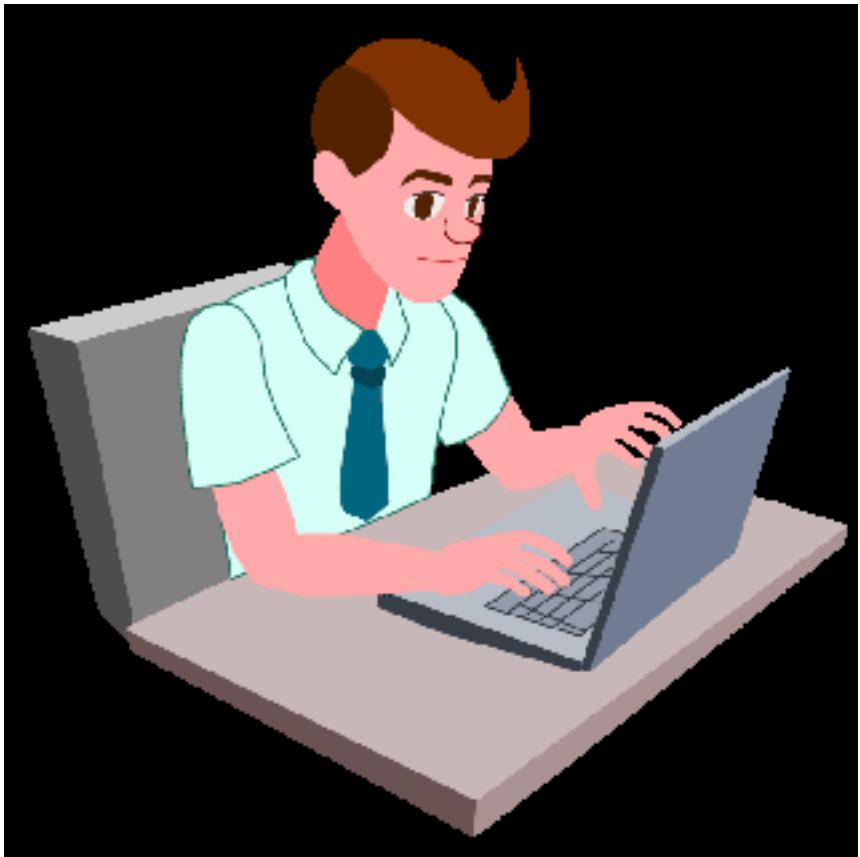
Cherries are red.

Text processing with awk

Knowledge check

For the following exercises, open up your Terminal and try to solve the following tasks:

1. Display the size (in bytes) of the file /etc/hostname.
2. Display only the group names in the file /etc/group.
3. Display the total number of lines in the file /etc/services.
4. Display only the lines that contain the word "bash" in the file /etc/passwd.
5. Display the output of the uptime command in all uppercase letters.



"Complete Lab"

11. Find/Locate Files

The locate command

- If you know the name of your file but you are unsure of the file's location, you can use the locate command to get the file's path.
- The locate command searches for a file location in a prebuilt file database, and thus it's crucial to update the file database before using the locate command.
- If you don't update the database, the locate command may fail to retrieve the location of newly created files.

Updating the file database

- To update the file database, you have to run the updatedb command as the root user:

```
root@ubuntu-linux:~# updatedb
```

- The updatedb command will not display any output.
- Now, let's say we forgot the location of the file facts.txt, and we don't remember where it is; in this case, you can run the locate command followed by the filename:

```
root@ubuntu-linux:~# locate facts.txt  
/home/elliot/facts.txt  
/var/facts.txt
```

Updating the file database

- Create an empty file named ghost.txt in the /home directory:

```
root@ubuntu-linux:/# touch /home/ghost.txt
```

- Now try searching for the file ghost.txt:

```
root@ubuntu-linux:/# locate ghost.txt
```

```
root@ubuntu-linux:/#
```

- The locate command couldn't find it! Why is that?..... That's because you created a new file, and the file database doesn't know about it yet.

- You have to run the updatedb command first to update the file database:

```
root@ubuntu-linux:/# updatedb
```

```
root@ubuntu-linux:/# locate ghost.txt
```

```
/home/ghost.txt
```

The find command

- The find command is a much more powerful command you can use to search for files in Linux.
- Unlike the locate command, the find command runs in real time, so you don't need to update any file database.
- The general syntax of the find command is as follows:

`find [starting-point(s)] [options] [expression]`

The find command

- For example, to search for all the .txt files under your /home directory, you can run:

```
root@ubuntu-linux:~# find /home -name "*.txt"
/home/elliot/facts2.txt
/home/elliot/dir1/directory2/file1.txt
/home/elliot/dir1/directory2/file3.txt
/home/elliot/dir1/directory2/file2.txt
/home/elliot/soft.txt
/home/elliot/facts.txt
/home/elliot/practise.txt
/home/elliot/upper.txt
/home/elliot/mydate.txt
/home/elliot/all.txt
/home/elliot/Mars.txt
/home/elliot/output.txt
/home/elliot/planets.txt
/home/elliot/error.txt
/home/elliot/animals.txt
/home/ghost.txt
```

The find command

- The -type option searches for file type; for example, to search for all the directories in /home/elliot/dir1, you can run:

```
root@ubuntu-linux:~# find /home/elliot/dir1 -type d  
/home/elliot/dir1  
/home/elliot/dir1/cities  
/home/elliot/dir1/directory2
```

- Notice it only listed the directories in /home/elliot/dir1. To list regular files instead, you can run:

```
root@ubuntu-linux:~# find /home/elliot/dir1 -type f  
/home/elliot/dir1/cities/paris  
/home/elliot/dir1/cities/london  
/home/elliot/dir1/cities/berlin  
/home/elliot/dir1/directory2/file1.txt  
/home/elliot/dir1/directory2/file3.txt  
/home/elliot/dir1/directory2/file2.txt
```

The find command

- To search for both regular files and directories, you can use a comma:

```
root@ubuntu-linux:~# find /home/elliot/dir1 -type d,f  
/home/elliot/dir1  
/home/elliot/dir1/cities  
/home/elliot/dir1/cities/paris  
/home/elliot/dir1/cities/london  
/home/elliot/dir1/cities/berlin  
/home/elliot/dir1/directory2  
/home/elliot/dir1/directory2/file1.txt  
/home/elliot/dir1/directory2/file3.txt  
/home/elliot/dir1/directory2/file2.txt
```

The find command

- Now as the root user create the two files large.txt and LARGE.TXT in /root:

```
root@ubuntu-linux:~# touch large.txt LARGE.TXT
```

- Let's say you forgot where these two files are located; in this case, you can use / as your starting-point:

```
root@ubuntu-linux:~# find / -name large.txt  
/root/large.txt
```

The find command

- Notice it only listed the location of large.txt, What if you wanted the other file LARGE.TXT as well? In this case, You can use the -iname option, which makes the search case insensitive:

```
root@ubuntu-linux:~# find / -iname large.txt  
/root/LARGE.TXT  
/root/large.txt
```

- Let's append the line "12345" to the file large.txt:

```
root@ubuntu-linux:~# echo 12345 >> large.txt
```

- Notice the size of the files large.txt and LARGE.txt:

```
root@ubuntu-linux:~# du -b large.txt LARGE.TXT  
6 large.txt  
0 LARGE.TXT
```

The find command

- For example, to search for empty files under the /root directory, you can run the command:

```
root@ubuntu-linux:~# find /root -size 0c  
/root/LARGE.TXT
```

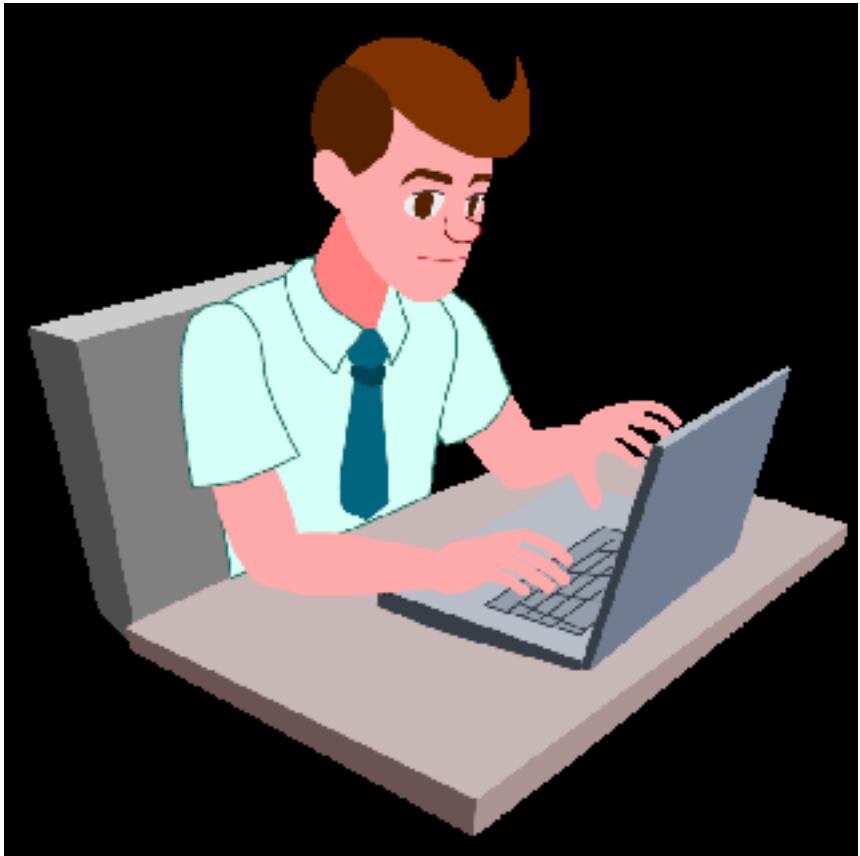
- As you can see, it listed LARGE.TXT as it has zero characters; 0c means zero characters (or bytes).
- Now, if you want to search for files of size 6 bytes under /root, you can run:

```
root@ubuntu-linux:~# find /root -size 6c  
/root/large.txt
```

Knowledge check

For the following exercises, open up your Terminal and try to solve the following tasks:

1. Use the locate command to find the path of the file boot.log.
2. Find all the files that are bigger than 50 MB in size.
3. Find all the files that are between 70 MB and 100 MB in size.
4. Find all the files that are owned by the user smurf.
5. Find all the files that are owned by the group developers.

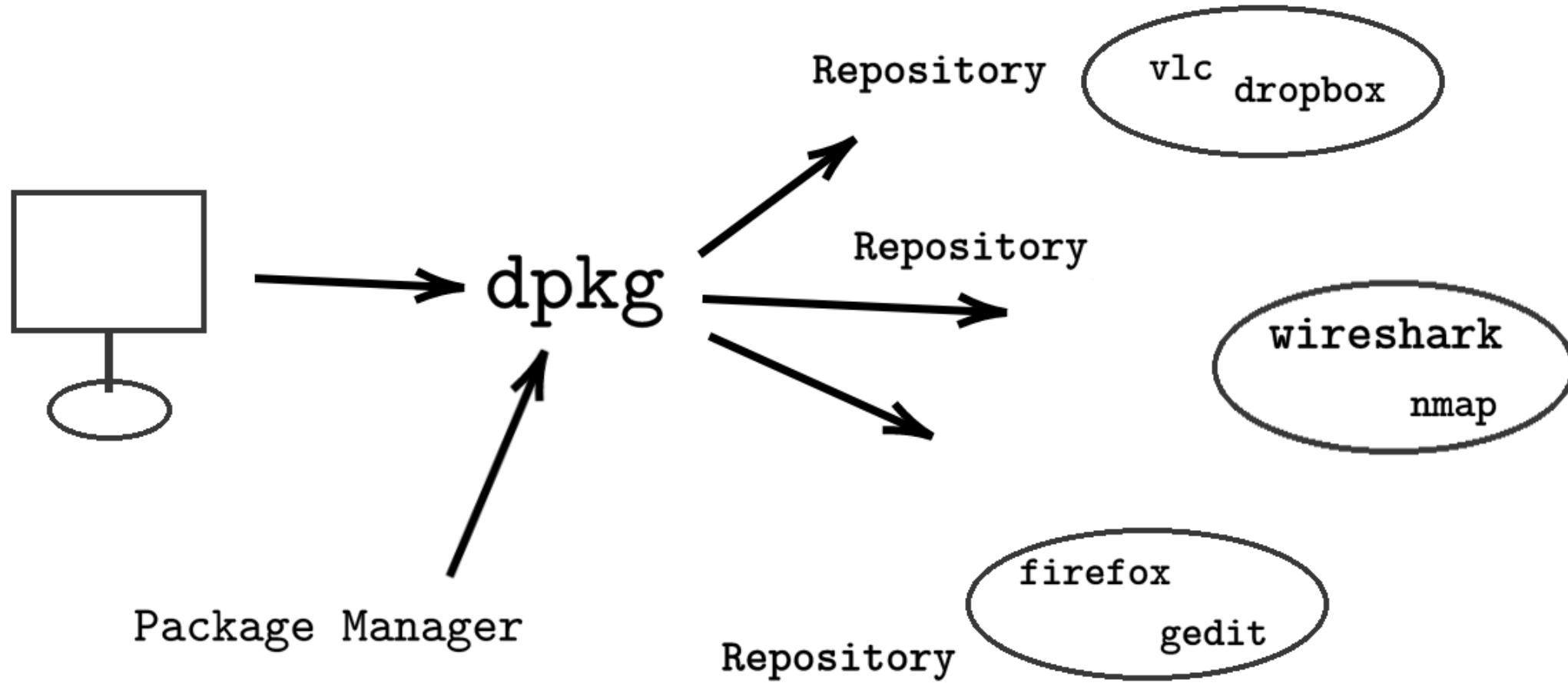


"Complete Lab"



12. Managing and Identifying Software Packages

Where do packages come from?



How to download packages

- On Ubuntu and other Debian Linux distributions, you can use the command-line utility apt-get to manage packages. Behind the scenes, apt-get makes use of the package manager dpkg.
- To download a package, you can run the command apt-get download followed by the package name:

`apt-get download package_name`

- As the root user, change to the /tmp directory:
`root@ubuntu-linux:~# cd /tmp`

How to download packages

- To download the cmatrix package, you can run the command:

```
root@ubuntu-linux:/tmp# apt-get download cmatrix
Get:1 http://ca.archive.ubuntu.com/ubuntu bionic/universe amd64
cmatrix amd64
1.2a-5build3 [16.1 kB]
Fetched 16.1 kB in 1s (32.1 kB/s)
```

- The cmatrix package will be downloaded in /tmp:

```
root@ubuntu-linux:/tmp# ls
cmatrix_1.2a-5build3_amd64.deb
```

How to download packages

```
root@ubuntu-linux:/tmp# dpkg -c cmatrix_1.2a-5build3_amd64.deb
drwxr-xr-x root/root    0 2018-04-03 06:17 .
drwxr-xr-x root/root    0 2018-04-03 06:17 ./usr/
drwxr-xr-x root/root    0 2018-04-03 06:17 ./usr/bin/
-rwxr-xr-x root/root 18424 2018-04-03 06:17 ./usr/bin/cmatrix
drwxr-xr-x root/root    0 2018-04-03 06:17 ./usr/share/
drwxr-xr-x root/root    0 2018-04-03 06:17 ./usr/share/consolefonts/
-rw-r--r-- root/root 4096 1999-05-13 08:55 ./usr/share/consolefonts/matrix.fnt
drwxr-xr-x root/root    0 2018-04-03 06:17 ./usr/share/doc/
drwxr-xr-x root/root    0 2018-04-03 06:17 ./usr/share/doc/cmatrix/
-rw-r--r-- root/root 2066 2000-04-03 19:29 ./usr/share/doc/cmatrix/README
-rw-r--r-- root/root 258 1999-05-13 09:12 ./usr/share/doc/cmatrix/TODO
-rw-r--r-- root/root 1128 2018-04-03 06:17 ./usr/share/doc/cmatrix/copyright
drwxr-xr-x root/root    0 2018-04-03 06:17 ./usr/share/man/
drwxr-xr-x root/root    0 2018-04-03 06:17 ./usr/share/man/man1/
-rw-r--r-- root/root  932 2018-04-03 06:17 ./usr/share/man/man1/cmatrix.1.gz
drwxr-xr-x root/root    0 2018-04-03 06:17 ./usr/share/menu/
-rw-r--r-- root/root  392 2018-04-03 06:17 ./usr/share/menu/cmatrix
```

How to download packages

- Notice that we only downloaded the package, but we didn't install it yet.
- Nothing will happen if you run the cmatrix command:

```
root@ubuntu-linux:/tmp# cmatrix  
bash: /usr/bin/cmatrix: No such file or directory
```

How to install packages

- You can use the -i option with the dpkg command to install a downloaded package:

```
root@ubuntu-linux:/tmp# dpkg -i cmatrix_1.2a-5build3_amd64.deb
```

Selecting previously unselected package cmatrix.

(Reading database ... 178209 files and directories currently installed.) Preparing to unpack cmatrix_1.2a-5build3_amd64.deb ...

Unpacking cmatrix (1.2a-5build3) ...

Setting up cmatrix (1.2a-5build3) ...

Processing triggers for man-db (2.8.3-2ubuntu0.1) ...

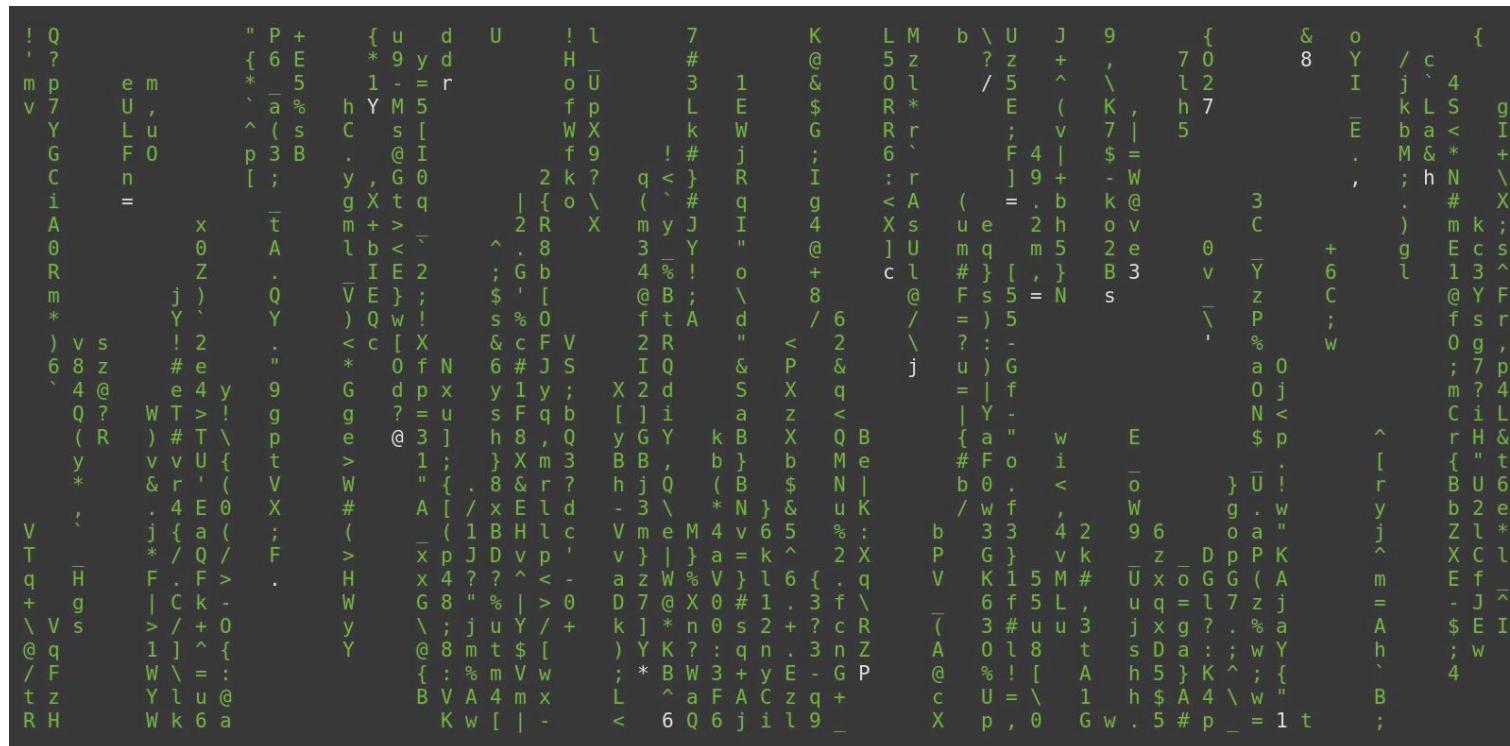
```
root@ubuntu-linux:/tmp#
```

- And that's it! Now run the cmatrix command:

```
root@ubuntu-linux:/tmp# cmatrix
```

How to install packages

- You will see the matrix running on your terminal like in the following image:



The image shows a terminal window displaying a matrix of characters and symbols. The characters are arranged in a grid pattern, where each character's position is determined by its ASCII value. The matrix includes letters, numbers, punctuation, and various symbols. The colors of the characters vary, creating a visual effect similar to the 'matrix' from the movie.

How to install packages

- We have taken the long way to install the cmatrix package.
- We first downloaded the package, and then we installed it.
- You can install a package right away (without downloading it) by running the command apt-get install followed by the package name:

`apt-get install package_name`

How to install packages

- For example, you can install the GNOME Chess game by running the command:

```
root@ubuntu-linux:/tmp# apt-get install gnome-chess
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  bbchess crafty fairymax fruit glaurung gnuchess phalanx sjeng stockfish toga2
The following NEW packages will be installed:
  gnome-chess
0 upgraded, 1 newly installed, 0 to remove and 357 not upgraded.
Need to get 0 B/1,514 kB of archives.
After this operation, 4,407 kB of additional disk space will be used.
Selecting previously unselected package gnome-chess.
(Reading database ... 178235 files and directories currently installed.) Preparing to unpack .../gnome-
chess_1%3a3.28.1-1_amd64.deb ...
Unpacking gnome-chess (1:3.28.1-1) ...
Processing triggers for mime-support (3.60ubuntu1) ...
Processing triggers for desktop-file-utils (0.23-1ubuntu3.18.04.2) ...
Processing triggers for libglib2.0-0:amd64 (2.56.3-0ubuntu0.18.04.1) ...
Setting up gnome-chess (1:3.28.1-1) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for gnome-menus (3.13.3-11ubuntu1.1) ...
Processing triggers for hicolor-icon-theme (0.17-2) ...
```

How to install packages

- Now you can start the game by running the gnome-chess command:

```
root@ubuntu-linux:/tmp# gnome-chess
```



How to remove packages

- You can easily remove a package by running the command apt-get remove followed by the package name:

`apt-get remove package_name`

- For example, if you are tired of the matrix lifestyle and have decided to remove the cmatrix package, you can run:

```
root@ubuntu-linux:/tmp# apt-get remove cmatrix
```

Reading package lists... Done

Building dependency tree

Reading state information... Done

The following packages will be REMOVED:

 cmatrix

0 upgraded, 0 newly installed, 1 to remove and 357 not upgraded.

After this operation, 49.2 kB disk space will be freed.

Do you want to continue? [Y/n] y

(Reading database ... 178525 files and directories currently installed.)

Removing cmatrix (1.2a-5build3) ...

Processing triggers for man-db (2.8.3-2ubuntu0.1) ...

How to remove packages

- Now, if you run the cmatrix command, you will get an error:

```
root@ubuntu-linux:/tmp# cmatrix
```

Command 'cmatrix' not found, but can be installed with:

```
apt install cmatrix
```

How to remove packages

```
root@ubuntu-linux:/tmp# apt-get purge gnome-chess
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  hoichess
Use 'apt autoremove' to remove it.
The following packages will be REMOVED:
  gnome-chess*
0 upgraded, 0 newly installed, 1 to remove and 357 not upgraded.
After this operation, 4,407 kB disk space will be freed.
Do you want to continue? [Y/n] y
(Reading database ... 178515 files and directories currently installed.)
Removing gnome-chess (1:3.28.1-1) ...
Processing triggers for mime-support (3.60ubuntu1) ...
Processing triggers for desktop-file-utils (0.23-1ubuntu3.18.04.2) ...
Processing triggers for libglib2.0-0:amd64 (2.56.3-0ubuntu0.18.04.1) ... Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for gnome-menus (3.13.3-11ubuntu1.1) ...
Processing triggers for hicolor-icon-theme (0.17-2) ...
(Reading database ... 178225 files and directories currently installed.)
Purging configuration files for gnome-chess (1:3.28.1-1) ...
```

How to search for packages

- Sometimes you are unsure of a package name.
- Then, in this case, you can't install it until you look it up.
- You can search for a package by using the command apt-cache search followed by your search term or keyword:

`apt-cache search keyword`

How to search for packages

```
root@ubuntu-linux:/tmp# apt-cache search shark
dopewars - drug-dealing game set in streets of New York City
dopewars-data - drug-dealing game set in streets of New York City - data files forensics-extra - Forensics Environment - extra console components (metapackage)
kernelshark - Utilities for graphically analyzing function tracing in the kernel libcrypto++-dev - General purpose cryptographic library - C++ development
libshark-dev - development files for Shark
libshark0 - Shark machine learning library
libwireshark-data - network packet dissection library -- data files
libwireshark-dev - network packet dissection library -- development files libwireshark10 - network packet dissection library -- shared library
libwritetap-dev - network packet capture library -- development files
libwsutil-dev - network packet dissection utilities library -- development files libwsutil8 - network packet dissection utilities library -- shared library
netmate - netdude clone that shows pcap dump lines in network header style plowshare-modules - plowshare drivers for various file sharing websites
shark-doc - documentation for Shark
tcpextract - extract files from network traffic based on file signatures
tshark - network traffic analyzer - console version
wifite - Python script to automate wireless auditing using aircrack-ng tools wireshark - network traffic analyzer - meta-package
wireshark-common - network traffic analyzer - common files
wireshark-dev - network traffic analyzer - development tools
wireshark-doc - network traffic analyzer - documentation
wireshark-gtk - network traffic analyzer - GTK+ version
wireshark-qt - network traffic analyzer - Qt version
zeitgeist-explorer - GUI application for monitoring and debugging zeitgeist forensics-extra-gui - Forensics Environment - extra GUI components (metapackage) horst
- Highly Optimized Radio Scanning Tool
libvirt-wireshark - Wireshark dissector for the libvirt protocol
libwritetap7 - network packet capture library -- shared library
libwscodecs1 - network packet dissection codecs library -- shared library minetest-mod-animals - Minetest mod providing animals
nsntrace - perform network trace of a single process by using network namespaces libwireshark11 - network packet dissection library -- shared library
libwritetap8 - network packet capture library -- shared library
libwscodecs2 - network packet dissection codecs library -- shared library libwsutil9 - network packet dissection utilities library -- shared library
```

How to search for packages

```
root@ubuntu-linux:/tmp# apt-cache -n search shark
kernelshark - Utilities for graphically analyzing function tracing in the kernel
libshark-dev - development files for Shark
libshark0 - Shark machine learning library
libwireshark-data - network packet dissection library -- data files
libwireshark-dev - network packet dissection library -- development files
libwireshark10 - network packet dissection library -- shared library
shark-doc - documentation for Shark
tshark - network traffic analyzer - console version
wireshark - network traffic analyzer - meta-package
wireshark-common - network traffic analyzer - common files
wireshark-dev - network traffic analyzer - development tools
wireshark-doc - network traffic analyzer - documentation
wireshark-gtk - network traffic analyzer - GTK+ version
wireshark-qt - network traffic analyzer - Qt version
libndpi-wireshark - extensible deep packet inspection library - wireshark dissector

libvirt-wireshark - Wireshark dissector for the libvirt protocol
libwireshark11 - network packet dissection library -- shared library
```

How to show package information

- To view package information, you can use the command `apt-cache show` followed by the package name:

`apt-cache show package_name`

- For example, to display the `cmatrix` package information, you can run:

```
root@ubuntu-linux:~# apt-cache show cmatrix
Package: cmatrix
Architecture: amd64
Version: 1.2a-5build3
Priority: optional
Section: universe/misc
Origin: Ubuntu
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Original-Maintainer: Diego Fernández Durán <diego@gredi.net>
Bugs: https://bugs.launchpad.net/ubuntu/+filebug
Installed-Size: 48
Depends: libc6 (>= 2.4), libncurses5 (>= 6), libtinfo5 (>= 6)
Recommends: kbd
Suggests: cmatrix-xfont
Filename: pool/universe/c/cmatrix/cmatrix_1.2a-5build3_amd64.deb
Size: 16084
MD5sum: 8dad2a99d74b63cce6eef0046f0ac91
SHA1: 3da3a0ec97807e6f53de7653e4e9f47fd96521c2
SHA256: cd50212101bfd71479af41e7afc47ea822c075ddb1ceed83895f8eaa1b79ce5d Homepage: http://www.asty.org/cmatrix/
Description-en_CA: simulates the display from "The Matrix"
Screen saver for the terminal based in the movie "The Matrix".
 * Support terminal resize.
 * Screen saver mode: any key closes it.
 * Selectable color.
 * Change text scroll rate.
Description-md5: 9af1f58e4b6301a6583f036c780c6ae6
```

How to show package information

- You can use the apt-cache depends command to list package dependencies:
`apt-cache depends package_name`
- For example, to view the list of packages that are needed to be installed for cmatrix to work properly, you can run the command:

```
root@ubuntu-linux:~# apt-cache depends cmatrix
```

`cmatrix`

Depends: libc6

Depends: libcurses5

Depends: libtinfo5

Recommends: kbd

Suggests: cmatrix-xfont

Listing all packages

- You can use the dpkg -l command to list all the packages that are installed on your system:

```
root@ubuntu-linux:~# dpkg -l
```

- You can also use the apt-cache pkgnames command to list all the packages that are available for you to install:

```
root@ubuntu-linux:~# apt-cache pkgnames
libdatrie-doc
libfstrcmp0-dbg
libghc-monadplus-doc
librime-data-sampheng
python-pyao-dbg
fonts-georgewilliams
python3-aptdaemon.test
libcollada2gltfconvert-dev
python3-doc8
r-bioc-hypergraph
.
.
.
.
.
.
```

Listing all packages

- You can pipe the output to the wc -l command to get the total number of available packages:

```
root@ubuntu-linux:~# apt-cache pkgnames | wc -l 64142
```

- You may also be interested to know which repositories (sources) your system used to obtain all these packages.
- These repositories are included in the file /etc/apt/sources.list and in any file with the suffix .list under the directory /etc/apt/sources.list.d/, You can check the man page:

```
root@ubuntu-linux:~# man sources.list
```

Listing all packages

- You can also use the apt-cache policy command to list all the enabled repositories on your system:

```
root@ubuntu-linux:~# apt-cache policy
Package files:
100 /var/lib/dpkg/status
  release a=now
500 http://dl.google.com/linux/chrome/deb stable/main amd64
  Packages release v=1.0,o=Google LLC,a=stable,n=stable,l=Google,c=main,
  b=amd64 origin dl.google.com
100 http://ca.archive.ubuntu.com/ubuntu bionic-backports/main i386
  Packages release v=18.04,o=Ubuntu,a=bionic-backports,n=bionic,l=Ubuntu,
  c=main,b=i386 origin ca.archive.ubuntu.com
100 http://ca.archive.ubuntu.com/ubuntu bionic-backports/main amd64
  Packages release v=18.04,o=Ubuntu,a=bionic-backports,n=bionic,l=Ubuntu,
  c=main,b=amd64 origin ca.archive.ubuntu.com
500 http://ca.archive.ubuntu.com/ubuntu bionic/multiverse i386
  Packages release v=18.04,o=Ubuntu,a=bionic,n=bionic,
  l=Ubuntu,c=multiverse,b=i386 origin ca.archive.ubuntu.com
500 http://ca.archive.ubuntu.com/ubuntu bionic/multiverse amd64
  Packages release v=18.04,o=Ubuntu,a=bionic,n=bionic,l=Ubuntu,
  c=multiverse,b=amd64 origin ca.archive.ubuntu.com
500 http://ca.archive.ubuntu.com/ubuntu bionic/universe i386
  Packages release v=18.04,o=Ubuntu,a=bionic,n=bionic,l=Ubuntu,
  c=universe,b=i386 origin ca.archive.ubuntu.com
500 http://ca.archive.ubuntu.com/ubuntu bionic/universe amd64
  Packages release v=18.04,o=Ubuntu,a=bionic,n=bionic,l=Ubuntu,
  c=universe,b=amd64 origin ca.archive.ubuntu.com
500 http://ca.archive.ubuntu.com/ubuntu bionic/restricted i386
  Packages release v=18.04,o=Ubuntu,a=bionic,n=bionic,l=Ubuntu,
  c=restricted,b=i386 origin ca.archive.ubuntu.com
500 http://ca.archive.ubuntu.com/ubuntu bionic/restricted amd64
  Packages release v=18.04,o=Ubuntu,a=bionic,n=bionic,l=Ubuntu,
  c=restricted,b=amd64 origin ca.archive.ubuntu.com
500 http://ca.archive.ubuntu.com/ubuntu bionic/main i386
  Packages release v=18.04,o=Ubuntu,a=bionic,
  n=bionic,l=Ubuntu,c=main,b=i386 origin ca.archive.ubuntu.com
500 http://ca.archive.ubuntu.com/ubuntu bionic/main amd64
  Packages release v=18.04,o=Ubuntu,a=bionic,n=bionic,
  l=Ubuntu,c=main,b=amd64 origin ca.archive.ubuntu.com
Pinned packages:
```

Listing all packages

- If you are eager to know which repository provides a specific package, you can use the apt-cache policy command followed by the package name:

`apt-cache policy package_name`

- For example, to know which repository provides the cmatrix package, you can run:

```
root@ubuntu-linux:~# apt-cache policy cmatrix  
cmatrix:
```

Installed: 1.2a-5build3

Candidate: 1.2a-5build3

Version table:

```
*** 1.2a-5build3 500
```

```
500 http://ca.archive.ubuntu.com/ubuntu bionic/universe amd64 Packages
```

```
100 /var/lib/dpkg/status
```

Patching your system

- If a newer release for a package is available, then you can upgrade it using the apt-get install --only-upgrade command followed by the package name:

`apt-get install --only-upgrade package_name`

- For example, you can upgrade the nano package by running the command:

```
root@ubuntu-linux:~# apt-get install --only-upgrade nano
```

Reading package lists... Done

Building dependency tree

Reading state information... Done

nano is already the newest version (2.9.3-2).

The following package was automatically installed and is no longer required:

 hoichess

Use 'apt autoremove' to remove it.

0 upgraded, 0 newly installed, 0 to remove and 357 not upgraded.

Patching your system

- The first command apt-get update will update the list of available packages and their versions, but it doesn't do any installation or upgrade:

```
root@ubuntu-linux:~# apt-get update
Ign:1 http://dl.google.com/linux/chrome/deb stable InRelease
Hit:2 http://ca.archive.ubuntu.com/ubuntu bionic InRelease
Hit:3 http://ppa.launchpad.net/linuxuprising/java/ubuntu bionic InRelease
Hit:4 http://dl.google.com/linux/chrome/deb stable Release
Hit:5 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:6 http://ca.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:8 http://ca.archive.ubuntu.com/ubuntu bionic-backports InRelease
Reading package lists... Done
```

Patching your system

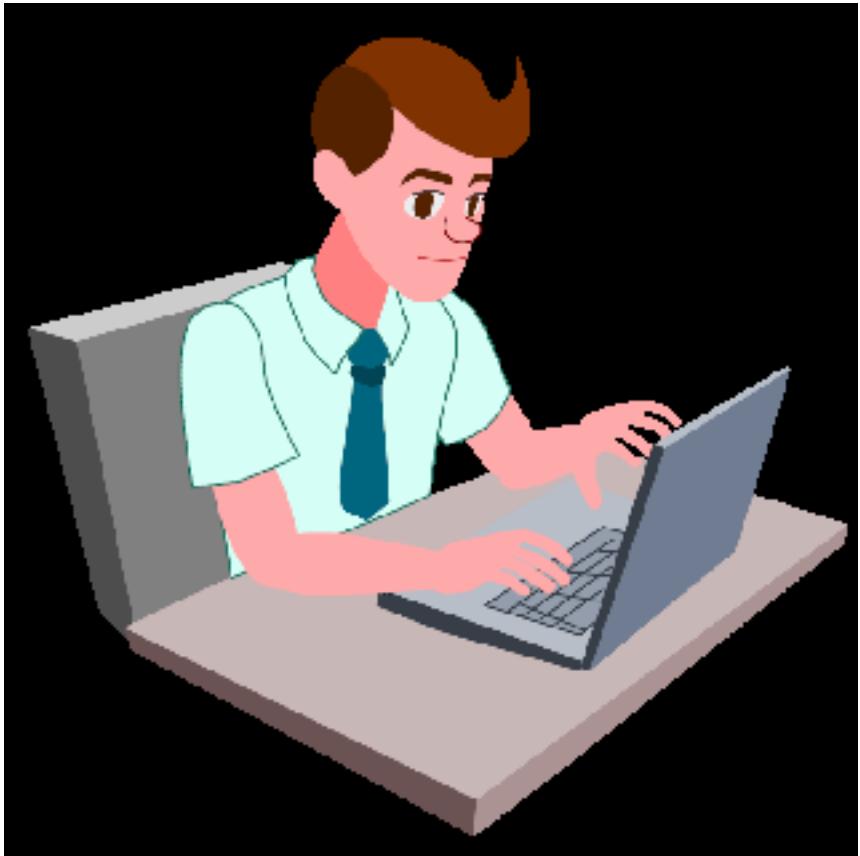
- The second command apt-get upgrade will upgrade all the installed packages on your system:

```
root@ubuntu-linux:~# apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following package was automatically installed and is no longer required:
  hoichess
Use 'apt autoremove' to remove it.
The following packages have been kept back:
  gstreamer1.0-gl libcogl20 libgail-3-0 libgl1-mesa-dri libgstreamer-gl1.0-0
  libreoffice-calc libreoffice-core libreoffice-draw libreoffice-gnome
  libreoffice-gtk3
  libwayland-egl1-mesa libxatracker2 linux-generic linux-headers-generic
  software-properties-common software-properties-gtk ubuntu-desktop
The following packages will be upgraded:
  apt apt-utils aptdaemon aptdaemon-data aspell base-files bash bind9-host bluez
  python2.7-minimal python3-apt python3-aptdaemon python3-aptdaemon.gtk3widgets
  python3-problem-report python3-update-manager python3-urllib3 python3.6
342 upgraded, 0 newly installed, 0 to remove and 30 not upgraded.
Need to get 460 MB of archives.
After this operation, 74.3 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Knowledge check

For the following exercises, open up your Terminal and try to solve the following tasks:

1. Install the tmux package on your system.
2. List all the dependencies of the vim package.
3. Install the cowsay package on your system.
4. Remove the cowsay package along with all its configuration files.
5. Upgrade all the packages on your system (patch your system).



"Complete Lab"

13. Controlling Processes

- To list all the processes that are owned by a specific user, you can run the command ps -u followed by the username:

`ps -u username`

- For example, to list all the processes that are owned by elliot, you can run:

```
root@ubuntu-linux:~# ps -u elliot
```

PID	TTY	TIME	CMD
1365	?	00:00:00	systemd
1366	?	00:00:00	(sd-pam)
1379	?	00:00:00	gnome-keyring-d
1383	tty2	00:00:00	gdm-x-session
1385	tty2	00:00:18	Xorg
1389	?	00:00:00	dbus-daemon
1393	tty2	00:00:00	gnome-session-b
1725	?	00:00:00	ssh-agent
1797	?	00:00:00	gvfsd

What is a process?

What is a process?

- You can use the ps -e command to list all the processes that are running on your system:

```
root@ubuntu-linux:~# ps -e
PID TTY      TIME CMD
 1 ?        00:00:01 systemd
 2 ?        00:00:00 kthreadd
 4 ?        00:00:00 kworker/0:0H
 6 ?        00:00:00 mm_percpu_wq
 7 ?        00:00:00 ksoftirqd/0
 8 ?        00:00:00 rcu_sched
 9 ?        00:00:00 rcu_bh
10 ?        00:00:00 migration/0
11 ?        00:00:00 watchdog/0
12 ?        00:00:00 cpuhp/0
13 ?        00:00:00 kdevtmpfs
.
.
.
.
```

What is a process?

- You can also use the -f option to get more information:

```
root@ubuntu-linux:~# ps -ef
UID      PID  PPID C STIME TTY      TIME     CMD
root        1    0 11:23    ? 00:00:01 /sbin/init splash
root        2    0 11:23    ? 00:00:00 [kthreadd]
root        4    2 0 11:23    ? 00:00:00 [kworker/0:0H]
root        6    2 0 11:23    ? 00:00:00 [mm_percpu_wq]
root        7    2 0 11:23    ? 00:00:00 [ksoftirqd/0]
root        8    2 0 11:23    ? 00:00:01 [rcu_sched]
root        9    2 0 11:23    ? 00:00:00 [rcu_bh]
root       10    2 0 11:23    ? 00:00:00 [migration/0]
elliot 1835 1393 1 11:25 tty2 00:00:58 /usr/bin/gnome-shell
elliot 1853 1835 0 11:25 tty2 00:00:00 ibus-daemon --xim --panel disable
elliot 1857 1365 0 11:25    ? 00:00:00 /usr/lib/gnome-shell/gnome-shell
elliot 1865 1853 0 11:25 tty2 00:00:00 /usr/lib/ibus/ibus-dconf
elliot 1868    1 0 11:25 tty2 00:00:00 /usr/lib/ibus/ibus-x11 --kill-daemon
elliot 1871 1365 0 11:25    ? 00:00:00 /usr/lib/ibus/ibus-portal
.
.
.
```

Parent process versus child process

- To get the PID of a process, you can use the pgrep command followed by the process name:

```
pgrep process_name
```

- For example, to get the PID of your terminal process, you can run:

```
elliott@ubuntu-linux:~$ pgrep terminal  
10009
```

- The PID of my terminal is 10009. Now, let's get the PID of the bash process:

```
elliott@ubuntu-linux:~$ pgrep bash  
10093
```

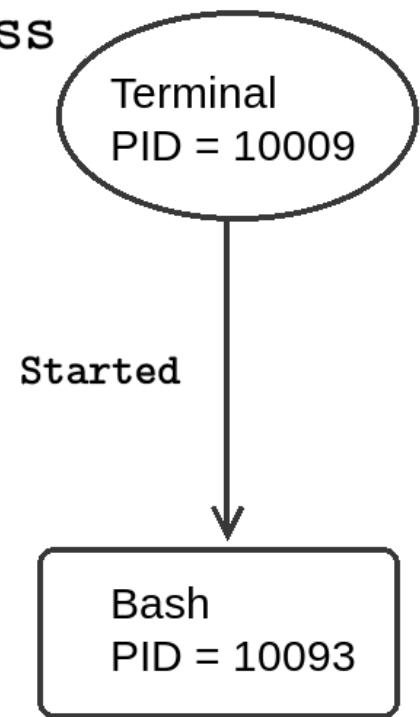
Parent process versus child process

- For the PID of my bash shell is 10093.
- Now, you can get the information of your bash process by using the -p option followed by the bash PID:

```
elliot@ubuntu-linux:~$ ps -fp 10093
UID  PID  PPID C STIME TTY   TIME CMD
elliot 10093 10009 0 13:37 pts/1 00:00:00 bash
```

Parent Process

Child Process



Parent process versus child process

- The top command is a very useful command that you can use to view processes' information in real time.
- You can check its man page to learn how to use it:

elliot@ubuntu-linux:~\$ man top

- The output for the preceding command is shown in the following screenshot:

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM
1385	elliot	20	0	442196	94152	44012	S	0.3	2.3
1835	elliot	20	0	3049584	349108	94900	S	0.3	8.6
10194	elliot	20	0	110076	3516	2500	S	0.3	0.1
10301	elliot	20	0	49112	3800	3124	S	0.3	0.1
10321	elliot	20	0	48884	3696	3076	R	0.3	0.1
1	root	20	0	159952	9196	6688	S	0.0	0.2
2	root	20	0	0	0	0	S	0.0	0.0
4	root	0	-20	0	0	0	I	0.0	0.0
6	root	0	-20	0	0	0	I	0.0	0.0
7	root	20	0	0	0	0	S	0.0	0.0
8	root	20	0	0	0	0	I	0.0	0.0
9	root	20	0	0	0	0	I	0.0	0.0
10	root	rt	0	0	0	0	S	0.0	0.0
11	root	rt	0	0	0	0	S	0.0	0.0
12	root	20	0	0	0	0	S	0.0	0.0

Foreground versus background processes

There are two types of processes in Linux:

Foreground processes

Background processes

Foreground versus background processes

- The yes command outputs any string that follows it repeatedly until killed:

```
elliot@ubuntu-linux:~$ whatis yes
```

```
yes (1)          - output a string repeatedly until killed
```

- For example, to output the word hello repeatedly on your terminal, you can run the command:

```
elliot@ubuntu-linux:~$ yes hello
.
.
.
```

Foreground versus background processes

- You can kill the process by hitting the Ctrl + C key combination as follows:

```
hello
```

```
^C
```

```
elliot@ubuntu-linux:~$
```

- As soon as you hit Ctrl + C, the process will be killed, and you can continue using your terminal.
- Let's do another example; you can use the firefox command to start up Firefox from your terminal:

```
elliot@ubuntu-linux:~$ firefox
```

Foreground versus background processes

- You can start up Firefox as a background process by adding the ampersand character as follows:

```
elliott@ubuntu-linux:~$ firefox &
[1] 3468
elliott@ubuntu-linux:~$
```

Sending signals to processes

- You can interact and communicate with processes via signals.
- There are various signals, and each signal serves a different purpose.
- To list all available signals, you can run the kill -L command:

```
elliot@ubuntu-linux:~$ kill -L
1) SIGHUP 2) SIGINT 3) SIGQUIT 4) SIGILL 5) SIGTRAP
6) SIGABRT 7) SIGBUS 8) SIGFPE 9) SIGKILL 10) SIGUSR1
11) SIGSEGV 12) SIGUSR2 13) SIGPIPE 14) SIGALRM 15) SIGTERM
16) SIGSTKFLT 17) SIGCHLD 18) SIGCONT 19) SIGSTOP 20) SIGTSTP
21) SIGTTIN 22) SIGTTOU 23) SIGURG 24) SIGXCPU 25) SIGXFSZ
26) SIGVTALRM 27) SIGPROF 28) SIGWINCH 29) SIGIO 30) SIGPWR
31) SIGSYS 34) SIGRTMIN 35) SIGRTMIN+1 36) SIGRTMIN+2 37) SIGRTMIN+3
38) SIGRTMIN+4 39) SIGRTMIN+5 40) SIGRTMIN+6 41) SIGRTMIN+7 42) SIGRTMIN+8
43) SIGRTMIN+9 44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9 56) SIGRTMAX-8 57) SIGRTMAX-7
58) SIGRTMAX-6 59) SIGRTMAX-5 60) SIGRTMAX-4 61) SIGRTMAX-3 62) SIGRTMAX-2
63) SIGRTMAX-1 64) SIGRTMAX
```

Sending signals to processes

- To see how signals work, let's first start Firefox as a background process:

```
elliott@ubuntu-linux:~$ firefox &  
[1] 4218
```

- Notice that the PID of Firefox is 4218 on my system. I can kill (terminate) Firefox by sending a SIGKILL signal as follows:

```
elliott@ubuntu-linux:~$ kill -SIGKILL 4218  
[1]+ Killed firefox
```

- This will immediately shut down Firefox.
- You can also use the numeric value of the SIGKILL signal instead:

```
elliott@ubuntu-linux:~$ kill -9 4218
```

Sending signals to processes

- In general, the syntax for the kill command is as follows:

`kill -SIGNAL PID`

- Let's start Firefox again as a background process:

```
elliott@ubuntu-linux:~$ firefox &  
[1] 4907
```

Sending signals to processes

- Notice that the PID of Firefox is 4907 on my system.
- Now go ahead and start playing a YouTube video on Firefox.
- After you have done that, go back to your terminal and send the SIGSTOP signal to Firefox:

```
elliot@ubuntu-linux:~$ kill -SIGSTOP 4907
```

- You will notice that Firefox becomes unresponsive and your YouTube video is stopped; no problem – we can fix that by sending the SIGCONT signal to Firefox:

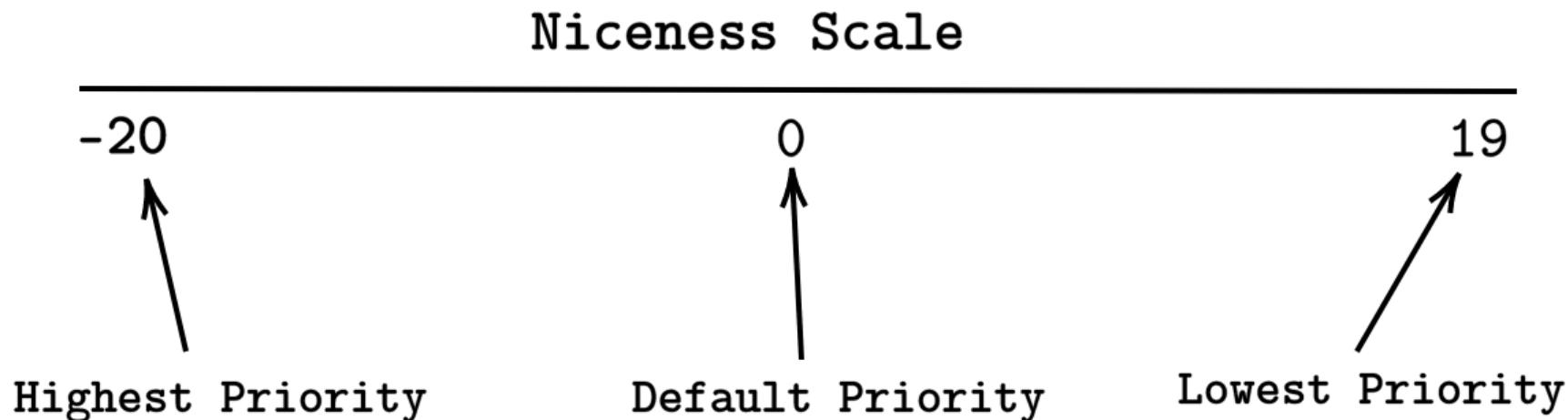
```
elliot@ubuntu-linux:~$ kill -SIGCONT 4907
```

Sending signals to processes

- You can use process names instead of process identifiers with the pkill command.
- For example, to close your terminal process, you can run the command:
`elliott@ubuntu-linux:~$ pkill -9 terminal`
- Now let's do something funny; open your terminal and run the command:
`elliott@ubuntu-linux:~$ pkill -SIGSTOP terminal`
- There are many other signals that you can send to processes; check the following man page to understand the use of each signal:
`elliott@ubuntu-linux:~$ man signal`

Working with process priority

- Each process has a priority that is determined by the niceness scale, which ranges from -20 to 19.
- The lower the nice value, the higher the priority of a process, so a nice value of -20 gives the highest priority to a process.
- On the other hand, a nice value of 19 gives the lowest priority to a process:



Viewing a process priority

- Start Firefox as a background process:

```
elliott@ubuntu-linux:~$ firefox &  
[1] 6849
```

- You can use the ps command to view a process' nice value:

```
elliott@ubuntu-linux:~$ ps -o nice -p 6849  
NI  
0
```

Setting priorities for new processes

- You can use the nice command to start a process with your desired priority.
- The general syntax of the nice command goes as follows:
`nice -n -20 →19 process`
- Let's say you are about to upgrade all the packages on your system; it would be wise to give such a process the highest priority possible.
- To do that, you can run the following command as the root user:

```
root@ubuntu-linux:~# nice -n -20 apt-get upgrade
```

Changing a process priority

- You can use the renice command to change the priority of a running process.
- We have already seen that Firefox was running with a default process priority of zero; let's change Firefox's priority and give it the lowest priority possible:

```
root@ubuntu-linux:~# renice -n 19 -p 6849  
6849 (process ID) old priority 0, new priority 19
```

The /proc directory

- Every process in Linux is represented by a directory in /proc.
- For example, if your Firefox process has a PID of 6849, then the directory /proc/6849 will represent the Firefox process:

```
root@ubuntu-linux:~# pgrep firefox
```

```
6849
```

```
root@ubuntu-linux:~# cd /proc/6849
```

```
root@ubuntu-linux:/proc/6849#
```

- Inside a process' directory, you can find a lot of valuable and insightful information about the process.
- For example, you will find a soft link named exe that points to the process' executable file:

```
root@ubuntu-linux:/proc/6849# ls -l exe
```

```
lrwxrwxrwx 1 elliot elliot 0 Nov 21 18:02 exe -> /usr/lib/firefox/firefox
```

- You will also find the status file, which stores various pieces of information about a process; these include the process state, the PPID, the amount of memory used by the process, and so on:

```
root@ubuntu-linux:/proc/6849# head status
```

```
Name: firefox
```

```
Umask: 0022
```

```
State: S (sleeping) Tgid: 6849
```

```
Ngid: 0
```

```
Pid: 6849
```

```
PPid: 1990
```

```
TracerPid: 0
```

```
Uid: 1000 1000 1000 1000
```

```
Gid: 1000 1000 1000 1000
```

The /proc directory

The /proc directory

- The limits file displays the current limits set for the process:

```
root@ubuntu-linux:/proc/7882# cat limits
Limit           Soft Limit  Hard Limit  Units
Max cpu time    unlimited  unlimited  seconds
Max file size   unlimited  unlimited  bytes
Max data size   unlimited  unlimited  bytes
Max stack size  8388608   unlimited  bytes
Max core file size 0        unlimited  bytes
Max resident set unlimited  unlimited  bytes
Max processes   15599     15599     processes
Max open files  4096      4096      files
Max locked memory 16777216 16777216  bytes
Max address space unlimited  unlimited  bytes
Max file locks   unlimited  unlimited  locks
Max pending signals 15599   15599   signals
Max msgqueue size 819200   819200   bytes
Max nice priority 0        0
Max realtime priority 0     0
Max realtime timeout unlimited  unlimited  us
```

The /proc directory

- The fd directory will show you all the files that the process is currently using on your system:

```
root@ubuntu-linux:/proc/6849# cd fd
root@ubuntu-linux:/proc/6849/fd# ls -l | tail
lrwx----- 1 elliot elliot 64 Nov 21 18:12 83 -> /home/elliot/.mozilla/firefox/places.sqlite-wal
lr-x----- 1 elliot elliot 64 Nov 21 18:12 84 -> /home/elliot/.mozilla/firefox/wkgfiatj.default
/favicons.sqlite
lrwx----- 1 elliot elliot 64 Nov 21 18:12 85 -> /home/elliot/.mozilla/firefox/wkgfiatj.default
/favicons.sqlite-wal
lrwx----- 1 elliot elliot 64 Nov 21 18:12 86 -> /home/elliot/.mozilla/firefox/wkgfiatj.default
/content-prefs.sqlite
lrwx----- 1 elliot elliot 64 Nov 21 18:12 88 -> /home/elliot/.mozilla/firefox/wkgfiatj.default
/webappsstore.sqlite
lr-x----- 1 elliot elliot 64 Nov 21 18:12 89 -> /usr/lib/firefox/browser/features
/formautofill@mozilla.org.xpi
lr-x----- 1 elliot elliot 64 Nov 21 18:12 9 -> /dev/shm/org.mozilla.ipc.6849.5 (deleted)
lrwx----- 1 elliot elliot 64 Nov 21 18:12 90 -> /home/elliot/.mozilla/firefox/wkgfiatj.default
/webappsstore.sqlite-wal
lr-x----- 1 elliot elliot 64 Nov 21 18:12 92 -> /home/elliot/.mozilla/firefox/wkgfiatj.default
/webappsstore.sqlite
lrwx----- 1 elliot elliot 64 Nov 21 18:12 93 -> /home/elliot/.mozilla/firefox/wkgfiatj.default
/webappsstore.sqlite-wal
```

The /proc directory

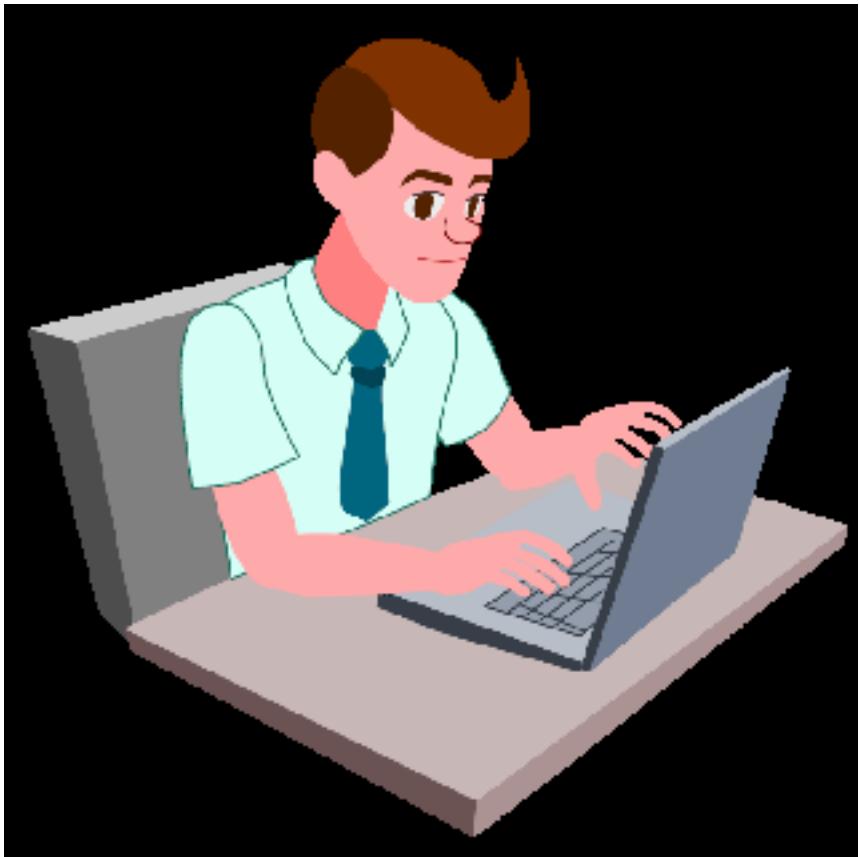
- You can also use the lsof command to list all the files a process is using:

```
root@ubuntu-linux:~# lsof -p 6849 | tail
lsof: WARNING: can't stat() fuse.gvfsd-fuse file system /run/user/1000/gvfs
      Output information may be incomplete.
firefox 6849 elliot  164u      unix 0xfffff918255ae1c00      0t0  77045 type=SEQPACKET
firefox 6849 elliot  165u      unix 0xfffff918255ae1800      0t0  77046 type=SEQPACKET
firefox 6849 elliot  166r      REG        0,23    58086      48 /dev/shm/org.mozilla.ipc.6849.41
firefox 6849 elliot  168u      unix 0xfffff918255ae2000      0t0  77049 type=STREAM
firefox 6849 elliot  170r      REG        0,23    21518      49 /dev/shm/org.mozilla.ipc.6849.42
firefox 6849 elliot  172r      REG        0,23     170      50 /dev/shm/org.mozilla.ipc.6849.43
firefox 6849 elliot  174r      REG        0,23    1918      51 /dev/shm/org.mozilla.ipc.6849.44
firefox 6849 elliot  176r      REG        0,23    1772      52 /dev/shm/org.mozilla.ipc.6849.45
firefox 6849 elliot  178r      REG        0,23   20920      53 /dev/shm/org.mozilla.ipc.6849.46
firefox 6849 elliot  180r      REG        0,23    5808      54 /dev/shm/org.mozilla.ipc.6849.47
```

Knowledge check

For the following exercises, open up your Terminal and try to solve the following tasks:

1. List the process ID of your running terminal.
2. List the parent process ID of your running terminal.
3. Use the kill command to close your terminal.
4. Start Firefox as a background process.
5. Change Firefox's priority to a maximum priority.



"Complete Lab"

14. The Power of Sudo (su and sudo Commands)



Examples of privileged commands

- You would find most of the commands that require root privileges in the directories /sbin and /usr/sbin.
- Let's switch to user smurf:

```
elliot@ubuntu-linux:~$ su - smurf
```

Password:

```
smurf@ubuntu-linux:~$
```

- Now let's see if smurf can add a new user to the system:

```
smurf@ubuntu-linux:~$ useradd bob
```

```
useradd: Permission denied.
```

Examples of privileged commands

- User smurf gets a permission denied error, That's because the useradd command is a privileged command.
- OK fine! Let's try installing the terminator package, which is a pretty cool Terminal emulator I must say:

```
smurf@ubuntu-linux:~$ apt-get install terminator
E: Could not open lock file /var/lib/dpkg/lock-frontend - open
(13: Permission denied)
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-
frontend),
are you root?
```

Granting access with sudo

- User smurf is now very sad as he can't add user bob or install the terminator package on the system.
- You can use the visudo command to grant user smurf the permissions to run the two privileged commands he wants.
- Run the visudo command as the root user:

```
root@ubuntu-linux:~# visudo
```

Granting access with sudo

- Will open up the file /etc/sudoers so you can edit it:

```
# 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  
# Host alias specification  
# User alias specification  
# Cmnd alias specification  
# User privilege specification  
root ALL=(ALL:ALL) ALL  
# Members of the admin group may gain root privileges  
%admin 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
```

Granting access with sudo

- All the lines that begin with the hash characters are comments, so only focus on these lines:

```
root  ALL=(ALL:ALL) ALL  
%admin ALL=(ALL) ALL  
%sudo ALL=(ALL:ALL) ALL
```

- The first line root ALL=(ALL:ALL) ALL is a rule that grants user root the permission to run all the commands on the system.
- We can now add a rule to grant user smurf the permission to run the useradd command.
- The syntax specification for a rule in the /etc/sudoers file is as follows:

```
user hosts=(user:group) commands
```

Granting access with sudo

- Now add the following rule to the /etc/sudoers file:

```
smurf  ALL=(ALL)    /usr/sbin/useradd
```

- The ALL keyword means no restrictions. Notice that you also have to include the full path of the commands. Now, save and exit the file then switch to user smurf:

```
root@ubuntu-linux:~# su - smurf  
smurf@ubuntu-linux:~$
```

- Now precede the useradd command with sudo as follows:

```
smurf@ubuntu-linux:~$ sudo useradd bob  
[sudo] password for smurf:  
smurf@ubuntu-linux:~$
```

Granting access with sudo

- It will prompt user smurf for his password; enter it, and just like that!
User bob is added:

```
smurf@ubuntu-linux:~$ id bob  
uid=1005(bob) gid=1005(bob) groups=1005(bob)  
smurf@ubuntu-linux:~$
```

- Cool! So smurf can now add users to the system; however, he still can't install any packages on the system:

```
smurf@ubuntu-linux:~$ sudo apt-get install terminator  
Sorry, user smurf is not allowed to execute '/usr/bin/apt-get install  
terminator' as root on ubuntu-linux.
```

Granting access with sudo

- Now let's fix that, Switch back to the root user and run the visudo command to edit the sudo rule for user smurf:

```
smurf ALL=(ALL) NOPASSWD: /usr/sbin/useradd, /usr/bin/apt-get install terminator
```

- Now, save and exit then switch back to user smurf and try to install the terminator package:

```
smurf@ubuntu-linux:~$ sudo apt-get install terminator
```

Reading package lists... Done

Building dependency tree

Reading state information... Done

The following packages were automatically installed and are no longer required:

gsfonts-x11 java-common

Use 'sudo apt autoremove' to remove them.

The following NEW packages will be installed:

terminator

Granting access with sudo

- Success! Notice that the sudo rule grants smurf permission only to install the terminator package.
- He will get an error if he tries to install any other package:

```
smurf@ubuntu-linux:~$ sudo apt-get install cmatrix
Sorry, user smurf is not allowed to execute '/usr/bin/apt-get install
cmatrix'
as root on ubuntu-linux.
```

User and command aliases

- You can use user aliases to reference multiple users in the /etc/sudoers file.
- For example, you can create a user alias MANAGERS that includes users smurf and bob as follows:

User_Alias MANAGERS = smurf,bob

- You can use a command alias to group multiple commands together. For example, you can create a command alias USER_CMDS that includes the commands useradd, userdel, and usermod:

Cmnd_Alias USER_CMDS = /usr/sbin/useradd, /usr/sbin/userdel,
/usr/sbin/usermod

- Now you can use both aliases:

MANAGERS ALL=(ALL) USER_CMDS

Group privileges

- You can also specify groups in the /etc/sudoers file.
- The group name is preceded by the percentage character as follows:
`%group hosts=(user:group) commands`

- The following rule will grant the developers group permission to install any package on the system:

```
%developers ALL=(ALL) NOPASSWD: /usr/bin/apt-get install
```

- The following rule will grant the developers group permission to run any command on the system:

```
%developers ALL=(ALL) NOPASSWD: ALL
```

Listing user privileges

- You can use the command sudo -lU to display a list of the sudo commands a user can run:

`sudo -lU username`

- For example, you can run the command:

`root@ubuntu-linux:~# sudo -lU smurf`

Matching Defaults entries for smurf on ubuntu-linux:

`env_reset, mail_badpass`

User smurf may run the following commands on ubuntu-linux:

(ALL) NOPASSWD: /usr/sbin/useradd, /usr/bin/apt-get install terminator

Listing user privileges

- To list all the sudo commands that can be run by user smurf.
- If a user is not allowed to run any sudo commands, the output of the command sudo-lU will be as follows:

```
root@ubuntu-linux:~# sudo -lU rachel  
User rachel is not allowed to run sudo on ubuntu-linux.
```

visudo versus /etc/sudoers

- First, run the visudo command and add the following line:

THISLINE=WRONG

- Now try to save and exit:

root@ubuntu-linux:~# visudo

>>> /etc/sudoers: syntax error near line 14 <<<

What now?

Options are:

(e)dit sudoers file again

e(x)it without saving changes to sudoers file

(Q)uit and save changes to sudoers file (DANGER!)

What now?

visudo versus /etc/sudoers

- Why is this important? Well, if you saved the file with an error in it, all the sudo rules in /etc/sudoers will not work!
- Let's hit Q to save the changes and then try to list the sudo commands that can be run by user smurf:

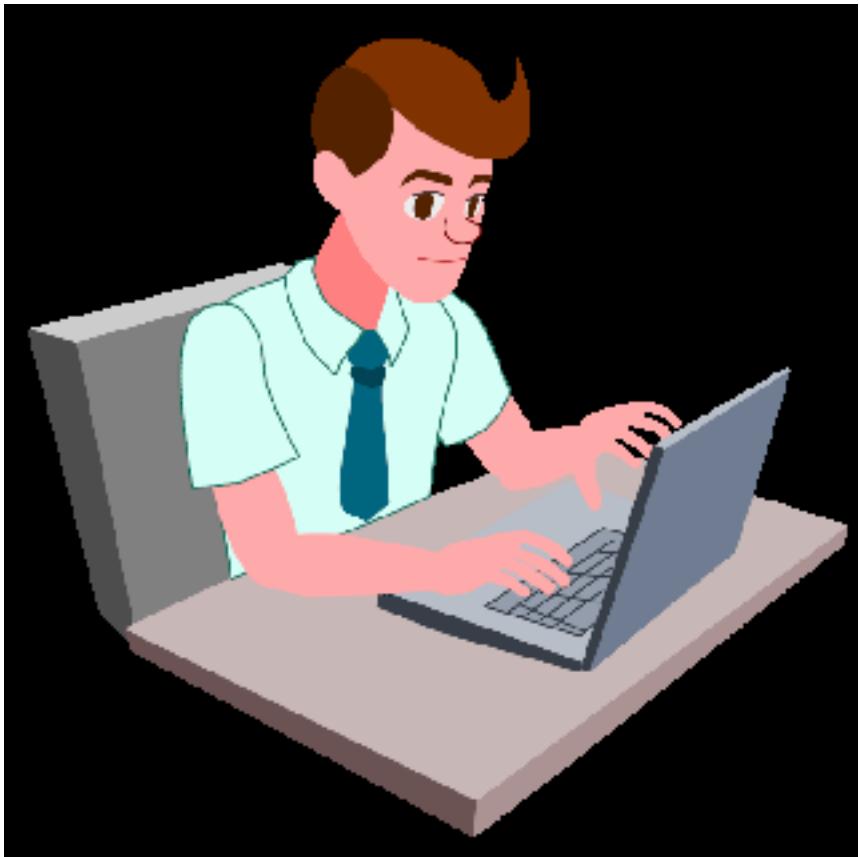
What now? Q

```
root@ubuntu-linux:~# sudo -lU smurf
>>> /etc/sudoers: syntax error near line 14 <<<
sudo: parse error in /etc/sudoers near line 14
sudo: no valid sudoers sources found, quitting
sudo: unable to initialize policy plugin
```

Knowledge check

For the following exercises, open up your Terminal and try to solve the following tasks:

1. Add a sudo rule so that user smurf can run the fdisk command.
2. Add a sudo rule so that the developers group can run the apt-get command.
3. List all the sudo commands of user smurf.



"Complete Lab"

15. Basic Networking

Testing network connectivity

- An easy way to check whether you have internet access on your Linux machine is by trying to reach any remote host (server) on the internet.
- This can be done by using the ping command. In general, the syntax of the ping command is as follows:

`ping [options] host`

Testing network connectivity

- For example, to test whether you can reach google.com, you can run the following command:

```
root@ubuntu-linux:~# ping google.com
PING google.com (172.217.1.14) 56(84) bytes of data.
64 bytes from iad23s25-in-f14.1e100.net (172.217.1.14): icmp_seq=1 ttl=55 time=38.7 ms
64 bytes from iad23s25-in-f14.1e100.net (172.217.1.14): icmp_seq=2 ttl=55 time=38.7 ms
64 bytes from iad23s25-in-f14.1e100.net (172.217.1.14): icmp_seq=3 ttl=55 time=40.4 ms
64 bytes from iad23s25-in-f14.1e100.net (172.217.1.14): icmp_seq=4 ttl=55 time=36.6 ms
64 bytes from iad23s25-in-f14.1e100.net (172.217.1.14): icmp_seq=5 ttl=55 time=40.8 ms
64 bytes from iad23s25-in-f14.1e100.net (172.217.1.14): icmp_seq=6 ttl=55 time=38.6 ms
64 bytes from iad23s25-in-f14.1e100.net (172.217.1.14): icmp_seq=7 ttl=55 time=38.9 ms
64 bytes from iad23s25-in-f14.1e100.net (172.217.1.14): icmp_seq=8 ttl=55 time=37.1 ms
^C
--- google.com ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 66ms
rtt min/avg/max/mdev = 36.555/38.724/40.821/1.344 ms
```

Testing network connectivity

- You can use the -c option to specify the number of packets you want to send to a host.
- For example, to only send three packets to google.com, you can run the following command:

```
root@ubuntu-linux:~# ping -c 3 google.com
PING google.com (172.217.1.14) 56(84) bytes of data.
```

```
64 bytes from iad23s25-in-f14.1e100.net (172.217.1.14): icmp_seq=1 ttl=55 time=39.3 ms
64 bytes from iad23s25-in-f14.1e100.net (172.217.1.14): icmp_seq=2 ttl=55 time=49.7 ms
64 bytes from iad23s25-in-f14.1e100.net (172.217.1.14): icmp_seq=3 ttl=55 time=40.8 ms
```

--- google.com ping statistics ---

```
3 packets transmitted, 3 received, 0% packet loss, time 59ms rtt min/avg/max/mdev =
39.323/43.267/49.708/4.595 ms
```

Testing network connectivity

- If you are not connected to the internet, you will get the following output from the ping command:

```
root@ubuntu-linux:~# ping google.com  
ping: google.com: Name or service not known
```

Listing your network interfaces

- You can list the available network interfaces on your system by viewing the contents of the /sys/class/net directory:

```
root@ubuntu-linux:~# ls /sys/class/net
eth0 lo wlan0
```

The ip command

- You can also use the ip link show command to view the available network interfaces on your system:

```
root@ubuntu-linux:~# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN mode DEFAULT group default qlen 1000
    link/ether f0:de:f1:d3:e1:e1 brd ff:ff:ff:ff:ff:ff
3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DORMANT group default qlen 1000
    link/ether 10:0b:a9:6c:89:a0 brd ff:ff:ff:ff:ff:ff
```

The nmcli command

- Another method that I prefer is using the nmcli device status command:

```
root@ubuntu-linux:~# nmcli device status
DEVICE TYPE STATE CONNECTION
wlan0 wifi    connected SASKTEL0206-5G
eth0 ethernet unavailable --
lo   loopback unmanaged --
```

Checking your IP address

- There are many different ways you can use to check your machine's IP address.
- You can use the old-school (yet still popular) ifconfig command followed by the name of your network interface that is connected to the internet:

```
root@ubuntu-linux:~# ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 172.16.1.73 netmask 255.255.255.0 broadcast 172.16.1.255
        inet6 fe80::3101:321b:5ec3:cf9 prefixlen 64 scopeid 0x20<link>
        ether 10:0b:a9:6c:89:a0 txqueuelen 1000 (Ethernet)
        RX packets 265 bytes 27284 (26.6 KiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 165 bytes 28916 (28.2 KiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Checking your IP address

- You can also use the -a option to list all network interfaces:

```
root@ubuntu-linux:~# ifconfig -a
eth0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
      ether f0:de:f1:d3:e1:e1 txqueuelen 1000 (Ethernet)
      RX packets 0 bytes 0 (0.0 B)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 0 bytes 0 (0.0 B)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
      device interrupt 20 memory 0xf2500000-f2520000

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
      inet 127.0.0.1 netmask 255.0.0.0
      inet6 ::1 prefixlen 128 scopeid 0x10<host>
      loop txqueuelen 1000 (Local Loopback)
      RX packets 4 bytes 156 (156.0 B)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 4 bytes 156 (156.0 B)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 172.16.1.73 netmask 255.255.255.0 broadcast 172.16.1.255
      inet6 fe80::3101:321b:5ec3:cf9 prefixlen 64 scopeid 0x20<link>
      ether 10:0b:a9:6c:89:a0 txqueuelen 1000 (Ethernet)
      RX packets 482 bytes 45500 (44.4 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 299 bytes 57788 (56.4 KiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Checking your IP address

- You can also use the newer ip command to check your machine's IP address.
- For example, you can run the ip address show command to list and show the status of all your network interfaces:

```
root@ubuntu-linux:~# ip address show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN
    link/ether f0:de:f1:d3:e1:e1 brd ff:ff:ff:ff:ff:ff
3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
    link/ether 10:0b:a9:6c:89:a0 brd ff:ff:ff:ff:ff:ff
        inet 172.16.1.73/24 brd 172.16.1.255 scope global dynamic
            noprefixroute wlan0 valid_lft 85684sec preferred_lft 85684sec
        inet6 fe80::3101:321b:5ec3:cf9/64 scope link noprefixroute
            valid_lft forever preferred_lft forever
```

Checking your gateway address

- Let's start with the first command, route -n:

```
root@ubuntu-linux:~# route -n Kernel IP routing table
Destination Gateway   Genmask   Flags Metric Ref Use Iface
0.0.0.0 172.16.1.254 0.0.0.0   UG     600   0 0 wlan0
172.16.1.0 0.0.0.0   255.255.255.0 U      600   0 0 wlan0
```

- You can see from the output that my default gateway IP address is 172.16.1.254. Now let's try the second command, netstat -rn:

```
root@ubuntu-linux:~# netstat -rn
Kernel IP routing table
Destination   Gateway   Genmask   Flags MSS Window irtt Iface
0.0.0.0 172.16.1.254 0.0.0.0   UG    0 0 0 wlan0
172.16.1.0 0.0.0.0   255.255.255.0 U      0 0 0 wlan0
```

Checking your gateway address

- The output almost looks identical. Now the output differs a little bit with the third command, ip route:

```
root@ubuntu-linux:~# ip route
default via 172.16.1.254 dev wlan0 proto dhcp metric 600
172.16.1.0/24 dev wlan0 proto kernel scope link src 172.16.1.73 metric 600
```

- The default gateway IP address is displayed on the first line: default via 172.16.1.254.
- You should also be able to ping your default gateway:

```
root@ubuntu-linux:~# ping -c 2 172.16.1.254
PING 172.16.1.254 (172.16.1.254) 56(84) bytes of data.
64 bytes from 172.16.1.254: icmp_seq=1 ttl=64 time=1.38 ms
64 bytes from 172.16.1.254: icmp_seq=2 ttl=64 time=1.62 ms

--- 172.16.1.254 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 3ms rtt min/avg/max/mdev =
1.379/1.501/1.624/0.128 ms
```

Flying with traceroute

- You are now ready to leave your house to go to work.
- You must go through different streets that eventually lead to your destination, right? Well, this is very similar to when you try to reach a host (website) on the internet; there is a route that you take that starts with your default gateway and ends with your destination.
- You can use the traceroute command to trace the route to any destination.
- The general syntax of the traceroute command is as follows:
traceroute destination

Flying with traceroute

- For example, you can trace the route from your machine to google.com by running the following command:

```
root@ubuntu-linux:~# traceroute google.com
traceroute to google.com (172.217.1.14), 30 hops max, 60 byte packets
1 172.16.1.254 (172.16.1.254) 15.180 ms 15.187 ms 15.169 ms
2 207-47-195-169.ngai.static.sasknet.sk.ca (207.47.195.169) 24.059 ms
3 142.165.0.110 (142.165.0.110) 50.060 ms 54.305 ms 54.903 ms
4 72.14.203.189 (72.14.203.189) 53.720 ms 53.997 ms 53.948 ms
5 108.170.250.241 (108.170.250.241) 54.185 ms 35.506 ms 108.170.250.225
6 216.239.35.233 (216.239.35.233) 37.005 ms 35.729 ms 38.655 ms
7 yyz10s14-in-f14.1e100.net (172.217.1.14) 41.739 ms 41.667 ms 41.581 ms
```

Breaking your DNS

- Every time you enter a domain name on your browser, the DNS translates (resolves) the domain name to its corresponding IP address.

The IP address of your DNS server is stored in the file /etc/resolv.conf:

```
root@ubuntu-linux:~# cat /etc/resolv.conf
```

```
# Generated by NetworkManager
```

```
nameserver 142.165.200.5
```

- I am using the DNS server 142.165.200.5, which is provided by my Internet Service Provider (ISP). You can use the nslookup command to see DNS in action.

- The general syntax of the nslookup command is as follows:

```
nslookup domain_name
```

Breaking your DNS

- The nslookup command uses DNS to obtain the IP address of a domain name.
- For example, to get the IP address of facebook.com, you can run the following command:

```
root@ubuntu-linux:~# nslookup facebook.com
```

```
Server: 142.165.200.5
```

```
Address: 142.165.200.5#53
```

Non-authoritative answer:

```
Name: facebook.com
```

```
Address: 157.240.3.35
```

```
Name: facebook.com
```

```
Address: 2a03:2880:f101:83:face:b00c:0:25de
```

Breaking your DNS

- You can also ping facebook.com:

```
root@ubuntu-linux:~# ping -c 2 facebook.com
```

```
PING facebook.com (157.240.3.35) 56(84) bytes of data.
```

```
64 bytes from edge-star-mini-shv-01-sea1.facebook.com (157.240.3.35):
```

```
icmp_seq=1 ttl=55 time=34.6 ms
```

```
64 bytes from edge-star-mini-shv-01-sea1.facebook.com (157.240.3.35):
```

```
icmp_seq=2 ttl=55 time=33.3 ms
```

```
--- facebook.com ping statistics ---
```

```
2 packets transmitted, 2 received, 0% packet loss, time 2ms
```

```
rtt min/avg/max/mdev = 33.316/33.963/34.611/0.673 ms
```

Breaking your DNS

- Now let's break things! My mum once told me that I have to break things so I can understand how they work.
- Let's see what life is without DNS by emptying the file /etc/resolv.conf:

```
root@ubuntu-linux:~# echo > /etc/resolv.conf  
root@ubuntu-linux:~# cat /etc/resolv.conf
```

```
root@ubuntu-linux:~#
```

- Now let's do nslookup on facebook.com:

```
root@ubuntu-linux:~# nslookup facebook.com
```

Breaking your DNS

- You will see that it hangs as it is unable to resolve domain names anymore, Now let's try to ping facebook.com:

```
root@ubuntu-linux:~# ping facebook.com  
ping: facebook.com: Temporary failure in name resolution
```

- You get the error message Temporary failure in name resolution, which is a fancy way of saying that your DNS is broken! However, you can still ping facebook.com by using its IP address:

```
root@ubuntu-linux:~# ping -c 2 157.240.3.35  
PING 157.240.3.35 (157.240.3.35) 56(84) bytes of data.  
64 bytes from 157.240.3.35: icmp_seq=1 ttl=55 time=134 ms  
64 bytes from 157.240.3.35: icmp_seq=2 ttl=55 time=34.4 ms  
--- 157.240.3.35 ping statistics ---  
2 packets transmitted, 2 received, 0% packet loss, time 2ms  
rtt min/avg/max/mdev = 34.429/84.150/133.872/49.722 ms
```

Breaking your DNS

- Let's fix our DNS, but this time we will not use the DNS server of our ISP; instead, we will use Google's public DNS server 8.8.8.8:

```
root@ubuntu-linux:~# echo "nameserver 8.8.8.8" > /etc/resolv.conf
```

```
root@ubuntu-linux:~# cat /etc/resolv.conf
```

```
nameserver 8.8.8.8
```

- Now let's do an nslookup on facebook.com again:

```
root@ubuntu-linux:~# nslookup facebook.com Server: 8.8.8.8
```

```
Address: 8.8.8.8#53
```

```
Non-authoritative answer:
```

```
Name: facebook.com
```

```
Address: 31.13.80.36
```

```
Name: facebook.com
```

```
Address: 2a03:2880:f10e:83:face:b00c:0:25de
```

Changing your hostname

- Every website has a domain name that uniquely identifies it over the internet; similarly, a computer has a hostname that uniquely identifies it over a network.
- Your computer's hostname is stored in the file /etc/hostname:

```
root@ubuntu-linux:~# cat /etc/hostname  
ubuntu-linux
```

Changing your hostname

- You can use hostnames to reach other computers in the same network (subnet).
- For example, I have another computer with the hostname backdoor that is currently running, and I can ping it:

```
root@ubuntu-linux:~# ping backdoor
PING backdoor (172.16.1.67) 56(84) bytes of data.
64 bytes from 172.16.1.67 (172.16.1.67): icmp_seq=1 ttl=64 time=3.27 ms
64 bytes from 172.16.1.67 (172.16.1.67): icmp_seq=2 ttl=64 time=29.3 ms
64 bytes from 172.16.1.67 (172.16.1.67): icmp_seq=3 ttl=64 time=51.4 ms
^C
--- backdoor ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 20ms
rtt min/avg/max/mdev = 3.272/27.992/51.378/19.662 ms
```

Changing your hostname

- Notice that backdoor is on the same network (subnet) and has an IP address of 172.16.1.67. I can also ping myself:

```
root@ubuntu-linux:~# ping ubuntu-linux
PING ubuntu-linux (172.16.1.73) 56(84) bytes of data.
64 bytes from 172.16.1.73 (172.16.1.73): icmp_seq=1 ttl=64 time=0.025 ms
64 bytes from 172.16.1.73 (172.16.1.73): icmp_seq=2 ttl=64 time=0.063 ms
^C
--- ubuntu-linux ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 14ms
rtt min/avg/max/mdev = 0.025/0.044/0.063/0.019 ms
```

Changing your hostname

- You can use the hostnamectl command to view and set your computer's hostname:

```
root@ubuntu-linux:~# hostnamectl
```

Static hostname: ubuntu-linux

Icon name: computer-vm

Chassis: vm

Machine ID: 106fd80252e541faafa4e54a250d1216

Boot ID: c5508514af114b4b80c55d4267c25dd4

Virtualization: oracle

Operating System: Ubuntu 18.04.3 LTS

Kernel: Linux 4.15.0-66-generic

Architecture: x86-64

Changing your hostname

- To change your computer's hostname, you can use the hostnamectl set-hostname command followed by the new hostname:

```
hostnamectl set-hostname new_hostname
```

- For example, you can change the hostname of your computer to myserver by running the following command:

```
root@ubuntu-linux:~# hostnamectl set-hostname myserver
```

```
root@ubuntu-linux:~# su -
```

```
root@myserver:~#
```

- Keep in mind that you need to open a new shell session so that your shell prompt displays the new hostname. You can also see that the file /etc/hostname is updated as it contains the new hostname:

```
root@ubuntu-linux:~# cat /etc/hostname
```

```
myserver
```

Restarting your network interface

- You can use the ifconfig command to bring down (disable) a network interface; you have to follow the network interface name with the down flag as follows:

`ifconfig interface_name down`

- For example, I can bring down my Wi-Fi interface, wlan0, by running the following command:

`root@myserver:~# ifconfig wlan0 down`

Restarting your network interface

- You can use the up flag to bring up (enable) a network interface:
`ifconfig interface_name up`
- For example, I can bring back up my Wi-Fi interface by running the following command:

```
root@myserver:~# ifconfig wlan0 up
```

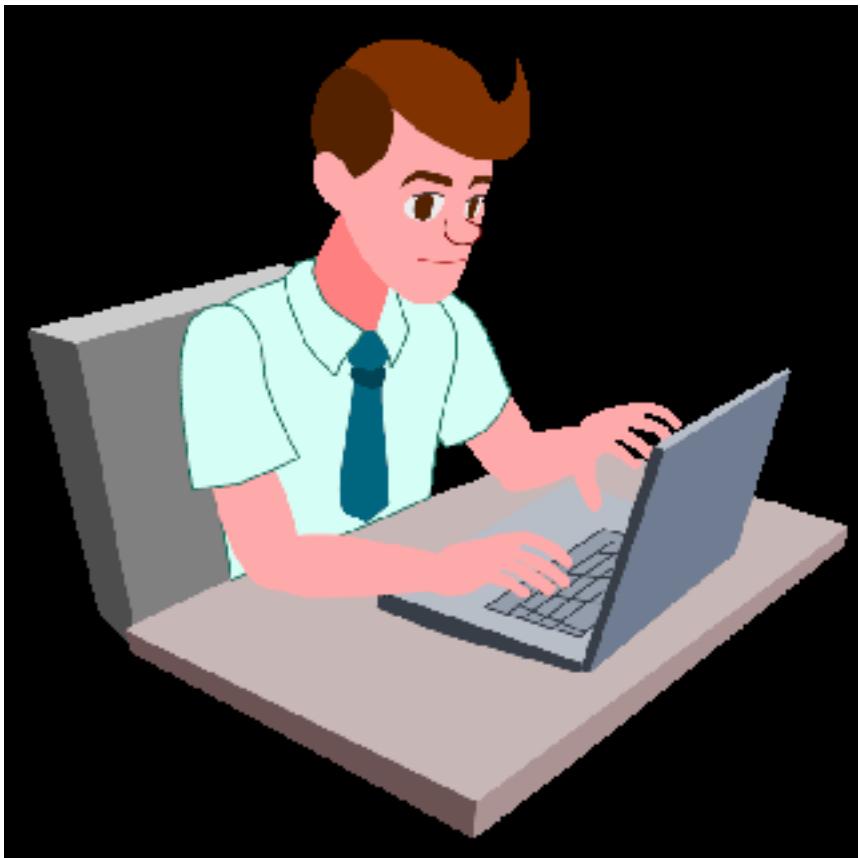
- You may also want to restart all your network interfaces at the same time.
- This can be done by restarting the NetworkManager service as follows:

```
root@myserver:~# systemctl restart NetworkManager
```

Knowledge check

For the following exercises, open up your Terminal and try to solve the following tasks:

1. Change your hostname to darkarmy.
2. Display the IP address of your default gateway.
3. Trace the route from your machine to www.ubuntu.com.
4. Display the IP address of your DNS.
5. Display the IP address of www.distrowatch.com.



"Complete Lab"

16. Shell Scripting Overview



Creating simple scripts

- Our first bash script will be a simple script that will output the line "Hello Friend!" to the screen.
- In Elliot's home directory, create a file named hello.sh and insert the following two lines:

```
elliot@ubuntu-linux:~$ cat hello.sh
#!/bin/bash
echo "Hello Friend!"
```

- Now we need to make the script executable:

```
elliot@ubuntu-linux:~$ chmod a+x hello.sh
```

- And finally, run the script:

```
elliot@ubuntu-linux:~$ ./hello.sh
Hello Friend!
```

The PATH variable

- You may have noticed that I used ./hello.sh to run the script; you will get an error if you omit the leading .:/

```
elliott@ubuntu-linux:~$ hello.sh  
hello.sh: command not found
```

- The shell can't find the command hello.sh.
- When you run a command on your terminal, the shell looks for that command in a set of directories that are stored in the PATH variable.
- You can use the echo command to view the contents of your PATH variable:

```
elliott@ubuntu-linux:~$ echo $PATH  
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

The PATH variable

- The colon character separates the path of each of the directories.
- You don't need to include the full path of any command or script (or any executable) that resides in these directories.
- All the commands you have learned so far reside in /bin and /sbin, which are both stored in your PATH variable. As a result, you can run the pwd command:

```
elliot@ubuntu-linux:~$ pwd  
/home/elliot
```

- There is no need to include its full path:

```
elliot@ubuntu-linux:~$ /bin/pwd  
/home/elliot
```

The PATH variable

- The good news is that you can easily add a directory to your PATH variable.
- For example, to add /home/elliot to your PATH variable, you can use the export command as follows:

```
elliot@ubuntu-linux:~$ export PATH=$PATH:/home/elliot
```

- Now you don't need the leading ./ to run the hello.sh script:

```
elliot@ubuntu-linux:~$ hello.sh
```

Hello Friend!

- It will run because the shell is now looking for executable files in the /home/elliot directory as well:

```
elliot@ubuntu-linux:~$ echo $PATH
```

```
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/home/elliot
```

The PATH variable

- Alright! Now let's create a few more bash scripts.
- We will create a script named hello2.sh that prints out "Hello Friend!" then displays your current working directory:

```
elliott@ubuntu-linux:~$ cat hello2.sh
```

```
#!/bin/bash
```

```
echo "Hello Friend!"
```

```
pwd
```

- Now let's run it:

```
elliott@ubuntu-linux:~$ hello2.sh
```

```
-bash: /home/elliott/hello2.sh: Permission denied
```

The PATH variable

- Shoot! I forgot to make it executable:

```
elliot@ubuntu-linux:~$ chmod a+x hello2.sh
```

```
elliot@ubuntu-linux:~$ ./hello2.sh
```

Hello Friend!

/home/elliot

Reading user input

- Let's create a better version of our hello.sh script.
- We will let the user input his/her name and then we will greet the user; create a script named greet.sh with the following lines:

```
elliott@ubuntu-linux:~$ cat greet.sh
```

```
#!/bin/bash
```

```
echo "Please enter your name:"
```

```
read name
```

```
echo "Hello $name!"
```

- Now make the script executable and then run it:

```
elliott@ubuntu-linux:~$ chmod a+x greet.sh
```

```
elliott@ubuntu-linux:~$ ./greet.sh
```

```
Please enter your name:
```

Reading user input

- When you run the script, it will prompt you to enter your name; I entered Elliot as my name:

```
elliot@ubuntu-linux:~$ ./greet.sh
```

```
Please enter your name:
```

```
Elliot
```

```
Hello Elliot!
```

Reading user input

- Let's create another script that reads a filename from the user and then outputs the size of the file in bytes; we will name our script size.sh:

```
elliott@ubuntu-linux:~$ cat size.sh
#!/bin/bash
echo "Please enter a file path:"
read file
filesize=$(du -bs $file | cut -f1)
echo "The file size is $filesize bytes"
```

- And never forget to make the script executable:

```
elliott@ubuntu-linux:~$ chmod a+x size.sh
```

Reading user input

- Now let's run the script:

```
elliott@ubuntu-linux:~$ size.sh
```

Please enter a file path

```
/home/elliott/size.sh
```

The file size is 128 bytes

- I used size.sh as the file path, and the output was 128 bytes; is that true? Let's check:

```
elliott@ubuntu-linux:~$ du -bs size.sh
```

```
128 size.sh
```

Reading user input

- Indeed it is; notice in the script the following line:
`filesize=$(du -bs $file | cut -f1)`
- It stores the result of the command `du -bs $file | cut -f1` in the variable `filesize`:

```
elliot@ubuntu-linux:~$ du -bs size.sh | cut -f1  
128
```

- Also notice that the command `du -bs $file | cut -f1` is surrounded by parentheses and a dollar sign (on the left); this is called command substitution.
- In general, the syntax of command substitution goes as follows:
`var=$(command)`

Passing arguments to scripts

- For example, let's create a bash script named size2.sh that does the same thing as the script size.sh, but instead of reading the file from the user, we will pass it to the script size2.sh as an argument:

```
elliott@ubuntu-linux:~$ cat size2.sh
#!/bin/bash
filesize=$(du -bs $1 | cut -f1)
echo "The file size is $filesize bytes"
```

- Now let's make the script executable:

```
elliott@ubuntu-linux:~$ chmod a+x size2.sh
```

Passing arguments to scripts

- Finally, you can run the script:

```
elliot@ubuntu-linux:~$ size2.sh /home/elliot/size.sh
```

```
The file size is 128 bytes
```

- You will get the same output as size.sh.

Notice that we provided the file path

- /home/elliot/size.sh as an argument to the script size2.sh.

Passing arguments to scripts

- We only used one argument in the script size2.sh, and it is referenced by \$1.
- You can pass multiple arguments as well; let's create another script size3.sh that takes two files (two arguments) and outputs the size of each file:

```
elliott@ubuntu-linux:~$ cat size3.sh
#!/bin/bash
filesize1=$(du -bs $1 | cut -f1)
filesize2=$(du -bs $2 | cut -f1)
echo "$1 is $filesize1 bytes"
echo "$2 is $filesize2 bytes"
```

Passing arguments to scripts

- Now make the script executable and run it:

```
elliott@ubuntu-linux:~$ size3.sh /home/elliott/size.sh  
/home/elliott/size3.sh  
/home/elliott/size.sh is 128 bytes  
/home/elliott/size3.sh is 136 bytes
```

- Awesome! As you can see, the first argument is referenced by \$1, and the second argument is referenced by \$2. So in general:

bash_script.sh argument1 argument2 argument3 ...

\$1 \$2 \$3

Using the if condition

- In general, the syntax of the if condition is as follows:

```
if [ condition is true ]; then  
    do this ...  
fi
```

- For example, let's create a script empty.sh that will examine whether a file is empty or not:

```
elliot@ubuntu-linux:~$ cat empty.sh  
#!/bin/bash  
filesize=$(du -bs $1 | cut -f1)  
if [ $filesize -eq 0 ]; then  
    echo "$1 is empty!"  
fi
```

Using the if condition

- Now let's make the script executable and also create an empty file named zero.txt:

```
elliot@ubuntu-linux:~$ chmod a+x empty.sh  
elliot@ubuntu-linux:~$ touch zero.txt
```

- Now let's run the script on the file zero.txt:

```
elliot@ubuntu-linux:~$ ./empty.sh zero.txt  
zero.txt is empty!
```

Using the if condition

- As you can see, the script correctly detects that zero.txt is an empty file; that's because the test condition is true in this case as the file zero.txt is indeed zero bytes in size:

```
if [ $filesize -eq 0 ];
```

- We used -eq to test for equality.
- Now if you run the script on a non-empty file, there will be no output:

```
elliot@ubuntu-linux:~$ ./empty.sh size.sh
```

```
elliot@ubuntu-linux:~$
```

Using the if condition

- We need to modify the script empty.sh so that it displays an output whenever it's passed a non-empty file; for that, we will use the if-else statement:

```
if [ condition is true ]; then
    do this ...
else
    do this instead ...
fi
```

Using the if condition

- Let's edit the empty.sh script by adding the following else statement:

```
elliott@ubuntu-linux:~$ cat empty.sh
#!/bin/bash
filesize=$(du -bs $1 | cut -f1)
if [ $filesize -eq 0 ]; then
echo "$1 is empty!"
else
echo "$1 is not empty!"
fi
```

Using the if condition

- Now let's rerun the script:

```
elliott@ubuntu-linux:~$ ./empty.sh size.sh
```

```
size.sh is not empty!
```

```
elliott@ubuntu-linux:~$ ./empty.sh zero.txt
```

```
zero.txt is empty!
```

- You can also use the elif (else-if) statement to create multiple test conditions:

```
if [ condition is true ]; then
```

```
    do this ...
```

```
elif [ condition is true]; then
```

```
    do this instead ...
```

```
fi
```

Using the if condition

- Let's create a script filetype.sh that detects a file type.
- The script will output whether a file is a regular file, a soft link, or a directory:

```
elliott@ubuntu-linux:~$ cat filetype.sh
```

```
#!/bin/bash
file=$1
if [ -f $1 ]; then
echo "$1 is a regular file"
elif [ -L $1 ]; then
echo "$1 is a soft link"
elif [ -d $1 ]; then
echo "$1 is a directory"
fi
```

Using the if condition

- Now let's make the script executable and also create a soft link to /tmp named tempfiles:

```
elliot@ubuntu-linux:~$ chmod a+x filetype.sh
```

```
elliot@ubuntu-linux:~$ ln -s /tmp tempfiles
```

- Now run the script on any directory:

```
elliot@ubuntu-linux:~$ ./filetype.sh /bin
```

```
/bin is a directory
```

- It correctly detects that /bin is a directory. Now run the script on any regular file:

```
elliot@ubuntu-linux:~$ ./filetype.sh zero.txt
```

```
zero.txt is a regular file
```

Using the if condition

- It correctly detects that zero.txt is a regular file. Finally, run the script on any soft link:

```
elliot@ubuntu-linux:~$ ./filetype.sh tempfiles
```

```
tempfiles is a soft link
```

- It correctly detects that tempfiles is a soft link.
- The following man page contains all the test conditions:

```
elliot@ubuntu-linux:~$ man test
```

Using the for loop

- The for loop has a few different syntaxes.
- If you are familiar with C++ or C programming, then you will recognize the following for loop syntax:

```
for ((initialize ; condition ; increment)); do  
// do something  
done
```

- Using the aforementioned C-style syntax; the following for loop will print out "Hello World" twenty times:

```
for ((i = 0 ; i < 20 ; i++)); do  
    echo "Hello World"  
done
```

Using the for loop

- Now let's create a script hello20.sh that has the for loop we just discussed:

```
elliott@ubuntu-linux:~$ cat hello20.sh
#!/bin/bash
for ((i = 0 ; i < 20 ; i++)); do
    echo "Hello World"
done
```

Using the for loop

- Now make the script executable and run it:

```
elliott@ubuntu-linux:~$ chmod a+x hello20.sh
elliott@ubuntu-linux:~$ hello20.sh
Hello World
```

Using the for loop

- It outputs the line "Hello World" twenty times as we expected. Instead of the C-style syntax, you can also use the range syntax with the for loop:

```
for i in {1..20}; do  
    echo "Hello World"  
done
```

- This will also output "Hello World" 20 times, This range syntax is particularly useful when working with a list of files.
- To demonstrate, create the following five files:

```
elliot@ubuntu-linux:~$ touch one.doc two.doc three.doc four.doc  
five.doc
```

Using the for loop

- Now let's say we want to rename the extension for all five files from .doc to .document.
- We can create a script rename.sh that has the following for loop:

```
#!/bin/bash
for i in /home/elliot/*.doc; do
    mv $i $(echo $i | cut -d. -f1).document
done
```

- Make the script executable and run it:

```
#!/bin/bash
elliot@ubuntu-linux:~$ chmod a+x rename.sh
elliot@ubuntu-linux:~$ ./rename.sh
elliot@ubuntu-linux:~$ ls *.document
five.document four.document one.document three.document two.document
```

- The while loop is another popular and intuitive loop.
- The general syntax for a while loop is as follows:

```
while [ condition is true ]; do  
    // do something  
done
```

- For example, we can create a simple script numbers.sh that prints the numbers from one to ten:

```
elliot@ubuntu-linux:~$ cat numbers.sh  
#!/bin/bash  
number=1  
while [ $number -le 10 ]; do  
    echo $number  
    number=$((number+1))  
done
```

Using the while loop

Using the while loop

- Make the script executable and run it:

```
elliott@ubuntu-linux:~$ chmod a+x numbers.sh
elliott@ubuntu-linux:~$ ./numbers.sh
1
2
3
4
5
6
7
8
9
10
```

Using the while loop

- The script is simple to understand; we first initialized the variable number to 1:

```
number=1
```

- Then we created a test condition that will keep the while loop running as long as the variable number is less than or equal to 10:

```
while [ $number -le 10 ]; do
```

- Notice that to evaluate an arithmetic expression, it needs to be within double parentheses as \${((arithmetic-expression))}:

```
echo $number
```

```
number=$((number+1))
```

Using the while loop

- For example, to generate a random permutation of the numbers between 1 and 10, you can run the following command:

```
elliott@ubuntu-linux:~$ shuf -i 1-10
1
6
5
2
10
8
3
9
7
4
```

Using the while loop

- Now we can use the -n option to select one number out of the permutation.
- This number will be random as well. So to generate a random number between 1 and 10, you can run the following command:

```
elliott@ubuntu-linux:~$ shuf -i 1-10 -n 1
```

6

Using the while loop

- Here is our lovely handcrafted script game.sh:

```
elliott@ubuntu-linux:~$ cat game.sh
#!/bin/bash
random=$(shuf -i 1-10 -n 1) #generate a random number between 1 and 10.
echo "Welcome to the Number Guessing Game"
echo "The lucky number is between 1 and 10."
echo "Can you guess it?"
tries=1
while [ true ]; do
echo -n "Enter a Number between 1-10: "
read number
if [ $number -gt $random ]; then
echo "Too high!"
elif [ $number -lt $random ]; then
echo "Too low!"
else
echo "Correct! You got it in $tries tries"
break #exit the loop
fi
tries=$((tries+1))
done
```

Using the while loop

- Now make the script executable and run it to start the game:

```
elliott@ubuntu-linux:~$ chmod a+x game.sh
elliott@ubuntu-linux:~$ game.sh
Welcome to the Number Guessing Game
The lucky number is between 1 and 10.
Can you guess it?
Enter a Number between 1-10: 4
Too low!
Enter a Number between 1-10: 7
Too low!
Enter a Number between 1-10: 9
Too high!
Enter a Number between 1-10: 8
Correct! You got it in 4 tries
```

Using the while loop

- It took me four tries in my first attempt at the game; I bet you can easily beat me!
- Let's go over our game script line by line. We first generate a random number between 1 and 10 and assign it to the variable random:
`random=$(shuf -i 1-10 -n 1) #generate a random number between 1 and 10.`
- We then print three lines that explain the game to the player:
`echo "Welcome to the Number Guessing Game"
echo "The lucky number is between 1 and 10."
echo "Can you guess it?"`

Using the while loop

- Next, we initialize the variable tries to 1 so that we can keep track of how many guesses the player took:

`tries=1`

- We then enter the game loop:

`while [true]; do`

- The first thing we do in the game loop is that we ask the player to enter a number between 1 and 10:

```
echo -n "Enter a Number between 1-10: "
read number
```

Using the while loop

- We then test to see if the number the player has entered is greater than, less than, or equal to the random number:

```
if [ $number -gt $random ]; then
echo "Too high!"
elif [ $number -lt $random ]; then
echo "Too low!"
else
echo "Correct! You got it in $tries tries"
break #exit the loop
fi
```

Using the while loop

- Finally, we increment the number of tries by 1 for each incorrect guess (high or low):

```
tries=$((tries+1))
```

Using the until loop

- Both the for and while loops run as long as the test condition is true.
- On the flip side, the until loop keeps running as long as the test condition is false.
- That's to say, it stops running as soon as the test condition is true.
- The general syntax of an until loop is as follows:

```
until [condition is true]; do  
  [commands]  
done
```

Using the until loop

- For example, we can create a simple script 3x10.sh that prints out the first ten multiples of 3:

```
elliott@ubuntu-linux:~$ cat 3x10.sh
#!/bin/bash
counter=1
until [ $counter -gt 10 ]; do
echo $((counter * 3))
counter=$((counter+1))
done
```

Using the until loop

- Now make the script executable and then run it:

```
elliot@ubuntu-linux:~$ chmod a+x 3x10.sh
elliot@ubuntu-linux:~$ 3x10.sh
3
6
9
12
15
18
21
24
27
30
```

Using the until loop

- The script is easy to understand, but you might scratch your head a little bit trying to understand the test condition of the until loop:

```
until [ $counter -gt 10 ]; do
```

- Notice that we can achieve the same result with a while loop that has the opposite test condition.
- You simply negate the test condition of the until loop and you will get the while loop equivalent:

```
while [ $counter -le 10 ]; do
```

Bash script functions

- When your scripts get bigger and bigger, things can get very messy, To overcome this problem, you can use bash functions.
- The idea behind functions is that you can reuse parts of your scripts, which in turn produces better organized and readable scripts.
- The general syntax of a bash function is as follows:

```
function_name () {  
    <commands>  
}
```

Bash script functions

- Let's create a function named hello that prints out the line "Hello World".
- We will put the hello function in a new script named fun1.sh:

```
elliott@ubuntu-linux:~$ cat fun1.sh
#!/bin/bash
```

```
hello () {
echo "Hello World"
}
hello # Call the function hello()
hello # Call the function hello()
hello # Call the function hello()
```

Bash script functions

- Now make the script executable and run it:

```
elliot@ubuntu-linux:~$ chmod a+x fun1.sh
```

```
elliot@ubuntu-linux:~$ ./fun1.sh
```

Hello World

Hello World

Hello World

Passing function arguments

- Functions can also take arguments the same way a script can take arguments.
- To demonstrate, we will create a script math.sh that has two functions add and sub:

```
elliott@ubuntu-linux:~$ cat math.sh
#!/bin/bash

add () {
echo "$1 + $2 =" $(( $1+$2 ))
}

sub () {
echo "$1 - $2 =" $(( $1-$2 ))
}

add 7 2
sub 7 2
```

Passing function arguments

- Make the script executable and then run it:

```
elliot@ubuntu-linux:~$ chmod a+x math.sh
```

```
elliot@ubuntu-linux:~$ ./math.sh
```

```
7 + 2 = 9
```

```
7 - 2 = 5
```

No browsing for you

- We will conclude this lesson with a pretty cool bash script noweb.sh that makes sure no user is having fun browsing the web on the Firefox browser:

```
elliott@ubuntu-linux:~$ cat noweb.sh
#!/bin/bash

shutdown_firefox() {
killall firefox 2> /dev/null
}

while [ true ]; do
shutdown_firefox
sleep 10 #wait for 10 seconds
done
```

No browsing for you

- Now open Firefox as a background process:

```
elliott@ubuntu-linux:~$ firefox &  
[1] 30436
```

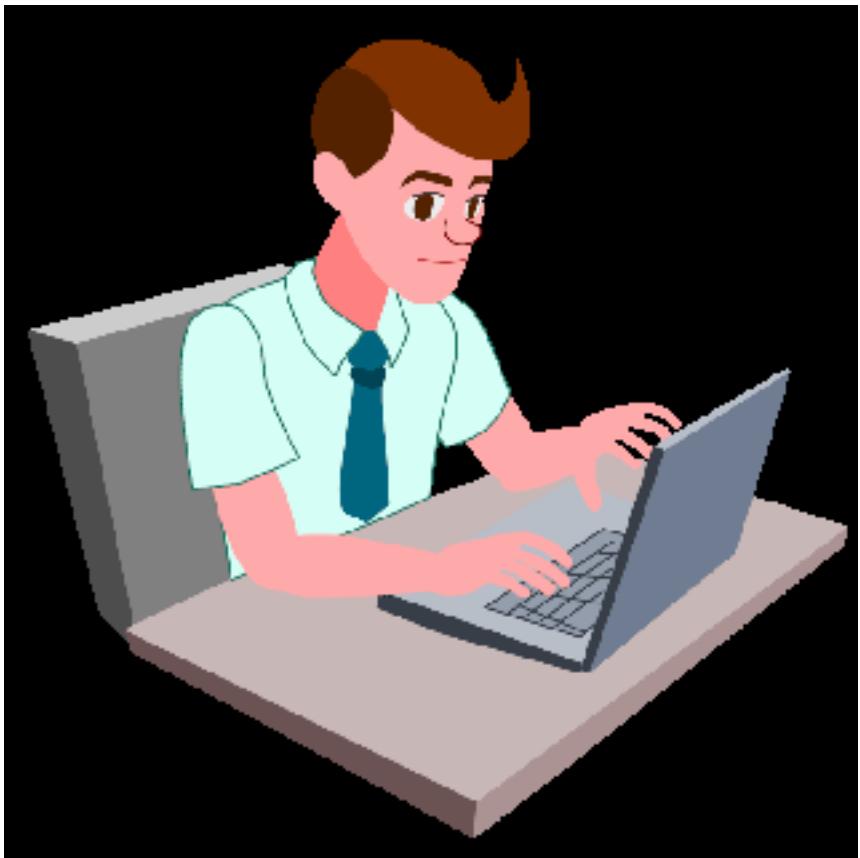
- Finally, make the script executable and run the script in the background:

```
elliott@ubuntu-linux:~$ chmod a+x noweb.sh  
elliott@ubuntu-linux:~$ ./noweb.sh &  
[1] 30759
```

Knowledge check

For the following exercises, open up your terminal and try to solve the following tasks:

1. Create a bash script that will display the calendar of the current month.
2. Modify your script so it displays the calendar for any year (passed as an argument).
3. Modify your script so it displays the calendar for all the years from 2000 to 2020.

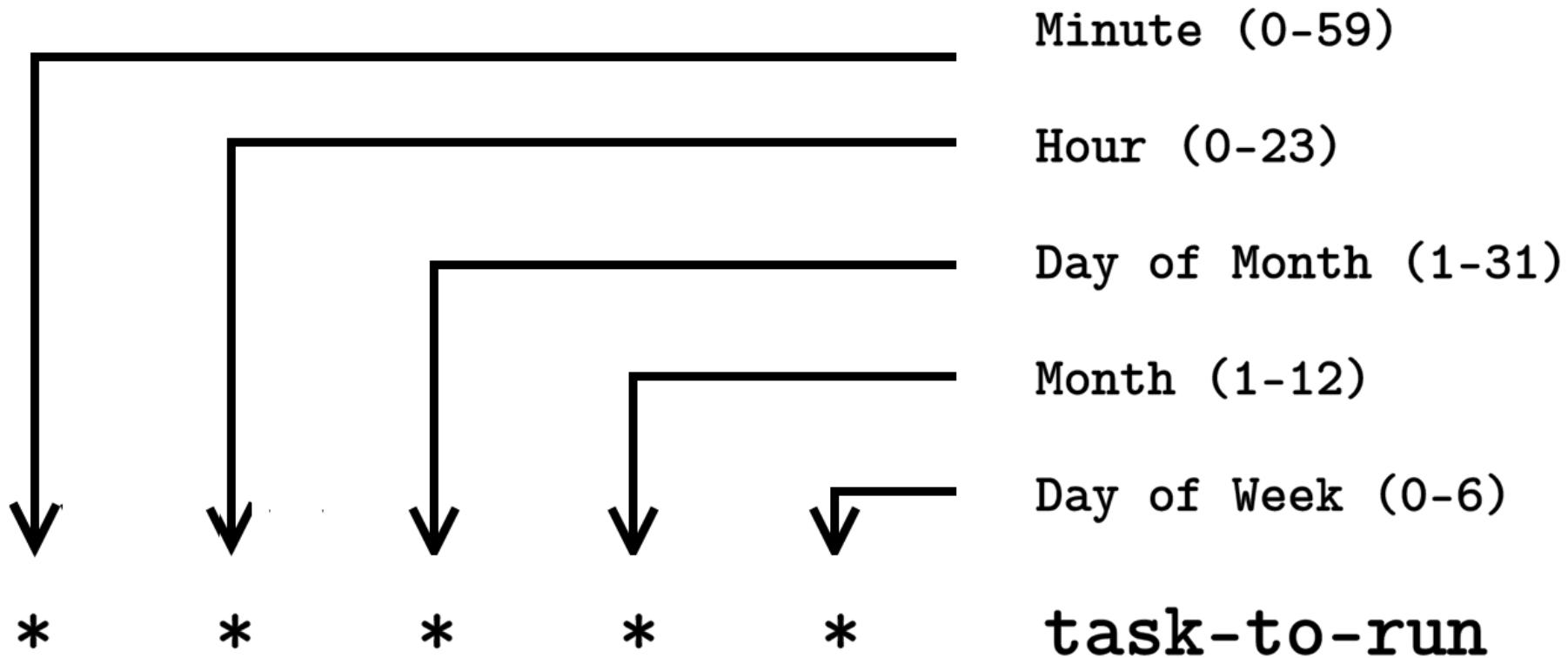


"Complete Lab"

17. Controlling Processes: cron and crontab

Our first cron job

- The following diagram shows you the typical format for a cron job:



Our first cron job

- Cron jobs are user-specific, and so each user has their own list of cron jobs.
- For example, the user elliot can run the command crontab -l to display his their of cron jobs:

```
elliot@ubuntu-linux:~$ crontab -l  
no crontab for elliot
```

- Currently, the user elliot doesn't have any cron jobs.
- You can run the command crontab -e to edit or create cron jobs:

```
elliot@ubuntu-linux:~$ crontab -e
```

Our first cron job

- Now add the following line and then save and exit:
`* * * * * echo "A minute has passed." >> /home/elliot/minutes.txt`
- After you exit, you will see the message: "crontab: installing new crontab":

```
elliot@ubuntu-linux:~$ crontab -e  
crontab: installing new crontab
```

- Finally, the user elliot can list their cron jobs to verify that the new cron job is scheduled:

```
elliot@ubuntu-linux:~$ crontab -l  
* * * * * echo "A minute has passed." >> /home/elliot/minutes.txt
```

Our first cron job

- Now, wait for a few minutes and then check the contents of the file /home/elliot/minutes.txt:

```
elliott@ubuntu-linux:~$ cat /home/elliott/minutes.txt
```

A minute has passed.

Run every five minutes

- Let's create another cron job that will run every five minutes.
- For example, you may want to create a cron job that checks the load average on your system every five minutes.

- Run the command crontab -e to add a new cron job:

```
elliot@ubuntu-linux:~$ crontab -e
```

- Now add the following line and then save and exit:

```
*/5 * * * * uptime >> /home/elliot/load.txt
```

Run every five minutes

- Finally, let's view the list of installed cron jobs to verify that the new cron job is scheduled:

```
elliot@ubuntu-linux:~$ crontab -e  
crontab: installing new crontab  
elliot@ubuntu-linux:~$ crontab -l  
* * * * * echo "A minute has passed" >> /home/elliot/minutes.txt  
*/5 * * * * uptime >> /home/elliot/load.txt
```

Run every five minutes

- Hang around for five or ten minutes and then check the contents of the file /home/elliot/load.txt.
- If you don't have a stopwatch, run the command sleep 300 and wait until it finishes:

```
elliot@ubuntu-linux:~$ sleep 300
```

- I made myself some green tea, and then came back after ten minutes and viewed the file /home/elliot/load.txt:

```
elliot@ubuntu-linux:~$ cat /home/elliot/load.txt
```

```
14:40:01 up 1 day, 5:13, 2 users, load average: 0.41, 0.40, 0.37
```

```
14:45:01 up 1 day, 5:18, 2 users, load average: 0.25, 0.34, 0.35
```

More cron examples

- You can also schedule your cron job to run at multiple time intervals.
- For example, the following cron job will run every hour on Sunday at the minutes 5, 20, and 40:

5,20,40 * * * sun task-to-run

- You can also specify a time range. For example, a cron job that will run at 6:30 PM on weekdays (Monday -> Friday) will have the following format:

30 18 * * 1-5 task-to-run

- To see more cron examples, you can check the fifth section of the crontab man page:

elliot@ubuntu-linux:~\$ man 5 crontab

Automating system patching

- Let's switch to the root user and then create a bash script named auto_patch.sh in /root:

```
root@ubuntu-linux:~# cat auto_patch.sh
#!/bin/bash
apt-get -y update
apt-get -y upgrade
shutdown -r now
```

Automating system patching

- Now make the script executable:

```
root@ubuntu-linux:~# chmod +x auto_patch.sh
```

- Finally, you need to schedule a cron job to run the auto_patch.sh script.
- Let's assume the system is scheduled to update on Saturday at 01:00 AM.
- In this case, you can create the following cron job:

```
0 1 * * sat /root/auto_patch.sh
```

Running a job once

- We first need to install the at package:

```
root@ubuntu-linux:~# apt-get -y install at
```

- Now you can schedule to run the auto_patch.sh script this coming Saturday at 01:00 AM with the following command:

```
root@ubuntu-linux:~# at 01:00 AM Sat -f /root/patch.sh
```

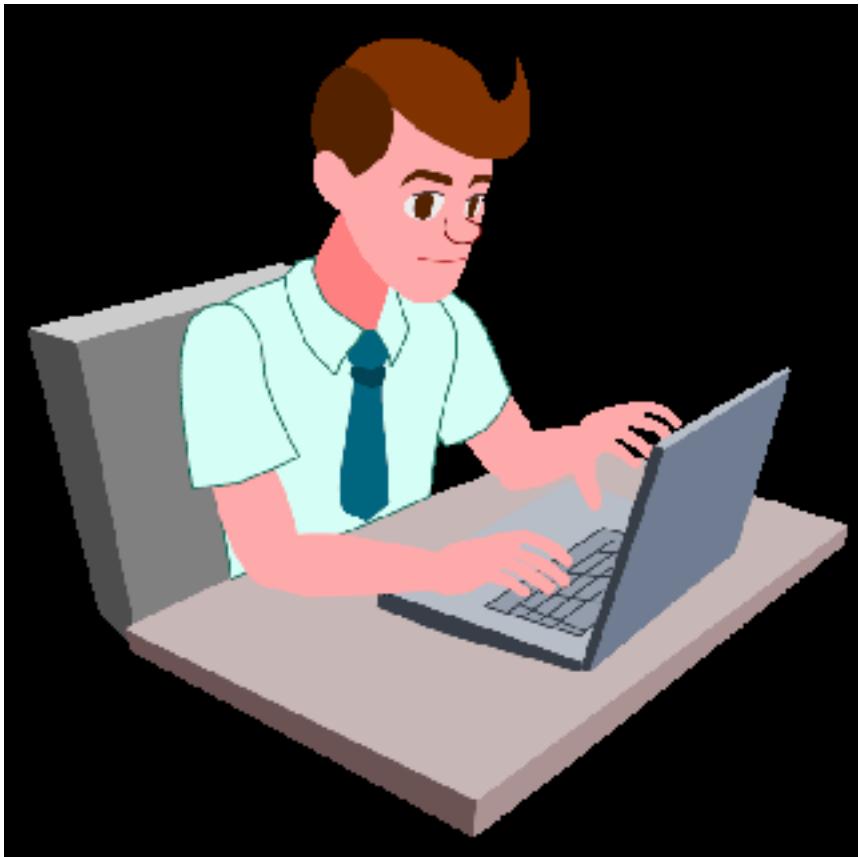
- Remember, at jobs only run once, so after Saturday, the auto_patch.sh script will not run again.
- You can learn more about at by reading its man page:

```
root@ubuntu-linux:~# man at
```

Knowledge check

For the following exercises, open up your terminal and try to solve the following tasks:

1. Create a cron job for the root user that will run every 10 minutes. The cron job will simply append the line "10 minutes have passed!" to the file /root/minutes.txt.
2. Create a cron job for the root user that will run every Christmas (25th of December at 1 AM). The cron job will simply append the line "Merry Christmas!" to the file /root/holidays.txt.



"Complete Lab"

18. System Backups

Creating an archive

- Let's create a backup for all the bash scripts in the /home/elliot directory.
- As the root user, create a directory named backup in /root:
`root@ubuntu-linux:~# mkdir /root/backup`

- To create an archive, we use the tape archive command tar.
- The general syntax to create an archive is as follows:

`tar -cf archive_name files`

Creating an archive

- To do that, we first change to the /home/elliot directory:

```
root@ubuntu-linux:~# cd /home/elliot  
root@ubuntu-linux:/home/elliot#
```

- Then we run the command:

```
root@ubuntu-linux:/home/elliot# tar -cf /root/backup/scripts.tar *.sh
```

- This will create the archive file scripts.tar in /root/backup, and there will be no command output:

```
root@ubuntu-linux:/home/elliot# ls -l /root/backup/scripts.tar  
-rw-r--r-- 1 root root 20480 Nov 1 23:12 /root/backup/scripts.tar
```

Creating an archive

- We could have also added the verbose option -v to see the files that are being archived:

```
root@ubuntu-linux:/home/elliot# tar -cvf /root/backup/scripts.tar *.sh  
3x10.sh  
detect.sh  
empty.sh  
filetype.sh  
fun1.sh  
game.sh  
hello20.sh  
hello2.sh  
hello3.sh  
hello.sh  
math.sh  
mydate.sh  
noweb.sh  
numbers.sh  
rename.sh  
size2.sh  
size3.sh  
size.sh
```

Viewing archive contents

- You may want to see the contents of an archive.
- To do that, you can use the `-t` option along with the `-f` option followed by the archive you wish to view:

`tar -tf archive`

- For example, to view the contents of the archive `scripts.tar` that we just created, you can run the command:

```
root@ubuntu-linux:/home/elliott# tar -tf /root/backup/scripts.tar
3x10.sh
detect.sh
empty.sh
filetype.sh
fun1.sh
game.sh
hello20.sh
hello2.sh
hello3.sh
hello.sh
math.sh
mydate.sh
noweb.sh
numbers.sh
rename.sh
size2.sh
size3.sh
size.sh
```

Extracting archive files

- You may also want to extract files from an archive.
- To demonstrate, let's create a directory named myscripts in /root:
`root@ubuntu-linux:/# mkdir /root/myscripts`
- To extract files from an archive, we use the **-x** option along with the **-f** option, followed by the archive name.
- Then, we use the **-C** option followed by the destination directory as follows:
`tar -xf archive -C destination`
- So to extract all the files in the scripts.tar archive to the /root/myscripts directory, you can run the following command:
`root@ubuntu-linux:/# tar -xf /root/backup/scripts.tar -C /root/myscripts`

Extracting archive files

- Now let's verify that the files were indeed extracted to the /root/myscripts directory:

```
root@ubuntu-linux:/# ls /root/myscripts  
3x10.sh  
empty.sh  
fun1.sh  
hello20.sh  
hello3.sh  
math.sh  
noweb.sh  
rename.sh  
size3.sh  
detect.sh  
filetype.sh  
game.sh  
hello2.sh  
hello.sh  
mydate.sh  
numbers.sh  
size2.sh  
size.sh
```

Compressing with gzip

- The most popular compression method on Linux is arguably gzip, and the upside is that it's really fast.
- You can compress an archive file with gzip by using the -z option with the tar command as follows:

```
tar -czf compressed_archive archive_name
```

- So to compress the scripts.tar archive into a gzip-compressed archive named scripts.tar.gz, you first need to change to the /root/backup directory and then run the following command:

```
root@ubuntu-linux:~/backup# tar -czf scripts.tar.gz scripts.tar
```

Compressing with gzip

- Now if you list the contents of the backup directory, you will see the newly created gzip-compressed archive scripts.tar.gz:

```
root@ubuntu-linux:~/backup# ls  
scripts.tar scripts.tar.gz
```

- Now let's run the file command on both archives:

```
root@ubuntu-linux:~/backup# file scripts.tar  
scripts.tar: POSIX tar archive (GNU)  
root@ubuntu-linux:~/backup# file scripts.tar.gz  
scripts.tar.gz: gzip compressed data, last modified: Sat Nov 2 22:13:44 2019,  
from Unix
```

Compressing with gzip

- The file command detects the type of both archives.
- Now let's compare the size (in bytes) of both archives:

```
root@ubuntu-linux:~/backup# du -b scripts.tar scripts.tar.gz
```

```
20480 scripts.tar
```

```
1479 scripts.tar.gz
```

- The compressed archive scripts.tar.gz is way smaller in size as we expected compared to the uncompressed archive scripts.tar.
- If you want to extract the files in the compressed archive scripts.tar.gz to /root/myscripts, you can run:

```
root@ubuntu-linux:~/backup# tar -xf scripts.tar.gz -C /root/myscripts
```

Compressing with bzip2

- You can compress an archive with bzip2 compression by using the -j option with the tar command as follows:

```
tar -cjf compressed_archive archive_name
```

- Notice the only difference here is that we use the -j option for bzip2 compression instead of -z for gzip compression.
- So to compress the scripts.tar archive into a bzip2-compressed archive named scripts.tar.bz2, you first need to change to the /root/backup directory and then run the following command:

```
root@ubuntu-linux:~/backup# tar -cjf scripts.tar.bz2 scripts.tar
```

Compressing with bzip2

- Now if you list the contents of the backup directory, you will see the newly created bzip2-compressed archive scripts.tar.bz2:

```
root@ubuntu-linux:~/backup# ls  
scripts.tar scripts.tar.bz2 scripts.tar.gz
```

- Let's run the file command on the bzip2-compressed archive scripts.tar.bz2:

```
root@ubuntu-linux:~/backup# file scripts.tar.bz2  
scripts.tar.bz2: bzip2 compressed data, block size = 900k
```

Compressing with bzip2

- It correctly detects the type of compression method used for the archive scripts.tar.bz2.
- Awesome – now let's compare the size (in bytes) of the gzip-compressed archive scripts.tar.gz and the bzip2-compressed archive scripts.tar.bz2:

```
root@ubuntu-linux:~/backup# du -b scripts.tar.bz2 scripts.tar.gz
```

```
1369 scripts.tar.bz2
```

```
1479 scripts.tar.gz
```

- Notice that the bzip2-compressed archive scripts.tar.bz2 is smaller than the gzip-compressed archive scripts.tar.gz.
- If you want to extract the files in the compressed archive scripts.tar.bz2 to /root/myscripts, you can run:

```
root@ubuntu-linux:~/backup# tar -xf scripts.tar.bz2 -C /root/myscripts
```

Compressing with xz

- You can compress an archive with xz compression by using the -J option with the tar command as follows:

```
tar -cJf compressed_name archive_name
```

- Notice here we use the uppercase letter J with xz compression.
- So to compress the scripts.tar archive into an xz-compressed archive named scripts.tar.xz, you first need to change to the /root/backup directory and then run the following command:

```
root@ubuntu-linux:~/backup# tar -cJf scripts.tar.xz scripts.tar
```

Compressing with xz

- Now if you list the contents of the backup directory, you will see the newly created xz-compressed archive scripts.tar.xz:

```
root@ubuntu-linux:~/backup# ls  
scripts.tar scripts.tar.bz2 scripts.tar.gz scripts.tar.xz
```

- Let's run the file command on the xz-compressed archive scripts.tar.xz:

```
root@ubuntu-linux:~/backup# file scripts.tar.xz  
scripts.tar.xz: XZ compressed data
```

Measuring performance

- You can use the time command to measure the time it takes a command (or a program) to finish executing.
- The general syntax for the time command is as follows:

`time command_or_program`

- For example, to measure how long it takes for the date command to finish executing, you can run the following command:

```
root@ubuntu-linux:~# time date
```

```
Sun Nov 3 16:36:33 CST 2019
```

```
real 0m0.004s
```

```
user 0m0.003s
```

```
sys 0m0.000s
```

Measuring performance

- The gzip compression method is the fastest of all three compression methods; well, let's see if I am lying or telling the truth! Change to the /root/backup directory:

```
root@ubuntu-linux:~# cd /root/backup  
root@ubuntu-linux:~/backup#
```

- Now let's see how long it takes to create a gzip-compressed archive file for all the files in /boot:

```
root@ubuntu-linux:~/backup# time tar -czf boot.tar.gz /boot  
real 0m4.717s  
user 0m4.361s  
sys 0m0.339s
```

Measuring performance

- On my system, it took gzip 4.717 seconds to run! Now let's measure the time it takes to create a bzip2-compressed archive of the same directory /boot:

```
root@ubuntu-linux:~/backup# time tar -cjf boot.tar.bz2 /boot  
real 0m19.306s  
user 0m18.809s  
sys 0m0.359s
```

- It took bzip2 an enormous 19.306 seconds to run! You can see how gzip compression is much faster than bzip2.
- Now let's see the time it takes to create an xz-compressed archive of the same directory /boot:

```
root@ubuntu-linux:~/backup# time tar -cJf boot.tar.xz /boot  
real 0m53.745s  
user 0m52.679s  
sys 0m0.873s
```

Measuring performance

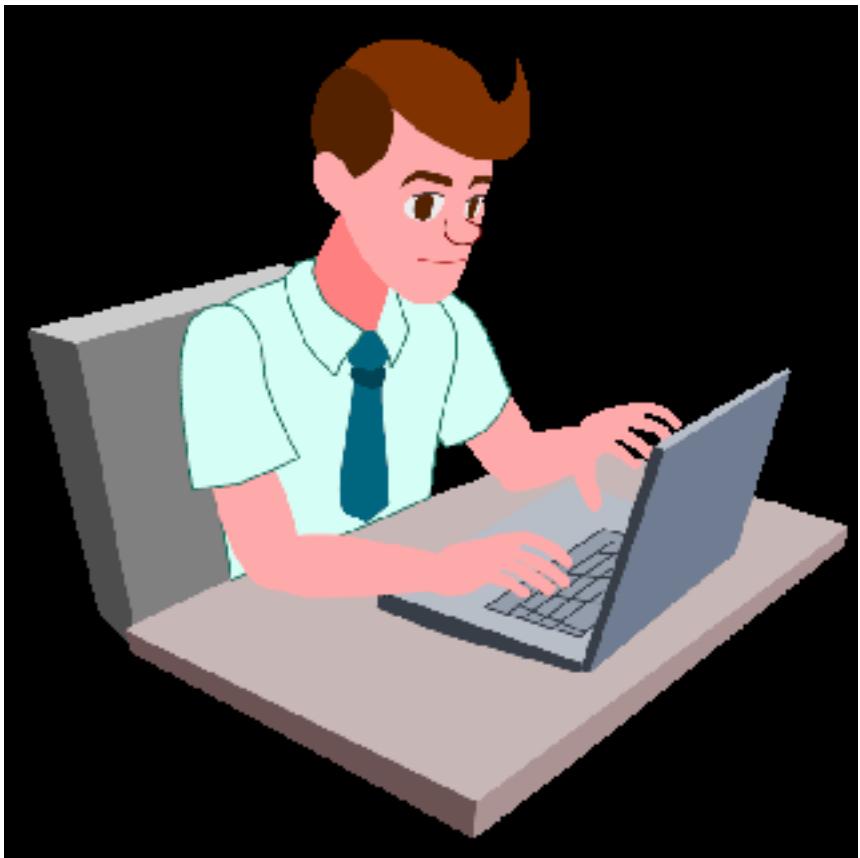
- Finally, let's check the size (in bytes) of the three compressed archives:

```
root@ubuntu-linux:~/backup# du -b boot.*  
97934386 boot.tar.bz2  
98036178 boot.tar.gz  
94452156 boot.tar.xz
```

Knowledge check

For the following exercises, open up your Terminal and try to solve the following tasks:

1. Create a gzip archive named var.tar.gz in /root for all the files in /var.
2. Create a bzip2 archive named tmp.tar.bz2 in /root for all the files in /tmp.
3. Create an xz archive named etc.tar.xz in /root for all the files in /etc.



"Complete Lab"

19. Creating Aliases

Your first alias

- Let's assume that you always forget that the command free -h displays the memory information of your system:

```
elliot@ubuntu-linux:~$ free -h
```

	total	used	free	shared	buff/cache	available
Mem:	3.9G	939M	2.2G	6.6M	752M	2.7G
Swap:	947M	0B	947M			

- The alias command instructs the shell to replace one string (word) with another.
- Well, how is this useful? Let me show you; if you run the following command:

```
elliot@ubuntu-linux:~$ alias memory="free -h"
```

Your first alias

- Then every time you enter memory, your shell will replace it with free -h:

```
elliot@ubuntu-linux:~$ memory
```

	total	used	free	shared	buff/cache	available
Mem:	3.9G	936M	2.2G	6.6M	756M	2.7G
Swap:	947M	0B	947M			

- Wow! So now you have achieved your dream! You can create an alias for any Linux command that you are having trouble remembering.
- Notice that the general format of the alias command is as follows:
`alias alias_name="command(s)_to_run"`

One alias for multiple commands

- You can use a semicolon to run multiple commands on the same line.
- For example, to create a new directory named newdir and change to newdir all at once, you can run the following command:

```
elliot@ubuntu-linux:~$ mkdir newdir; cd newdir
```

```
elliot@ubuntu-linux:~/newdir$
```

- So you use a semicolon to separate each command.
- In general, the syntax for running multiple commands on the same line is as follows:

```
command1; command2; command3; command4; ....
```

One alias for multiple commands

- We often like to check the calendar and the date at the same time, right? For that, we will create an alias named date so that every time we run date, it will run both the date and calendar commands:

```
elliot@ubuntu-linux:~$ alias date="date;cal"
```

- Now let's run date and see what's up:

```
[elliot@ubuntu-linux:~$ date
Mon Nov  4 13:34:04 CST 2019
November 2019
Su Mo Tu We Th Fr Sa
                    1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
```

Listing all aliases

- You should also know that aliases are user-specific.
- So the aliases created by elliot will not work for user smurf; take a look:

```
elliot@ubuntu-linux:~$ su - smurf
```

Password:

```
smurf@ubuntu-linux:~$ date
```

Mon Nov 4 13:33:36 CST 2019

```
smurf@ubuntu-linux:~$ memory
```

Command 'memory' not found, did you mean:

 command 'lmemory' from deb lmemory

Try: apt install <deb name>

Listing all aliases

- As you can see, smurf can't use the aliases of user Elliot, So every user has their own set of aliases.
- Now, let's exit back to user elliot:

```
smurf@ubuntu-linux:~$ exit
```

```
logout
```

```
elliot@ubuntu-linux:~$ memory
```

	total	used	free	shared	buff/cache	available
Mem:	3.9G	937M	2.0G	6.6M	990M	2.7G
Swap:	947M	0B	947M			

Listing all aliases

- You can run the alias command to list all the aliases that can be used by the currently logged-in user:

```
elliott@ubuntu-linux:~$ alias  
alias date='date;cal'  
alias egrep='egrep --color=auto'  
alias fgrep='fgrep --color=auto'  
alias grep='grep --color=auto'  
alias l='ls -CF'  
alias la='ls -A'  
alias ll='ls -alF'  
alias ls='ls --color=auto'  
alias memory='free -h'
```

Creating a permanent alias

- Open a new Terminal session, then try and run the two aliases we have created:

```
elliot@ubuntu-linux:~$ date  
Mon Nov 4 13:43:46 CST 2019  
elliot@ubuntu-linux:~$ memory
```

Command 'memory' not found, did you mean:
 command 'lmemory' from deb lmemory
Try: sudo apt install <deb name>

Creating a permanent alias

- You can see, they are gone! They are not even in your list of aliases anymore:

```
elliott@ubuntu-linux:~$ alias  
alias egrep='egrep --color=auto'  
alias fgrep='fgrep --color=auto'  
alias grep='grep --color=auto'  
alias l='ls -CF'  
alias la='ls -A'  
alias ll='ls -alF'  
alias ls='ls --color=auto'
```

Creating a permanent alias

- To create a permanent alias for a user, you need to include it in the hidden .bashrc file in the user's home directory.
- So to permanently add our two aliases back, you have to add the following two lines at the very end of the /home/el- liot/.bashrc file:

alias memory = "free -h"

alias date = "date;cal"

- You can do it by running the following two echo commands:

```
elliott@ubuntu-linux:~$ echo 'alias memory="free -h"' >>  
/home/elliott/.bashrc
```

```
elliott@ubuntu-linux:~$ echo 'alias date="date;cal"' >>  
/home/elliott/.bashrc
```

Creating a permanent alias

- After you add both aliases to the /home/elliot/.bashrc file, you need to run the source command on the /home/elliot/.bashrc file for the change to take effect in the current session:

```
elliot@ubuntu-linux:~$ source /home/elliot/.bashrc
```

- Now you can use your two aliases, memory and date, forever without worrying that they will disappear after you close your current Terminal session:

```
elliot@ubuntu-linux:~$ memory
              total        used        free      shared  buff/cache   available
Mem:       3.8G       233M      3.3G        672K      282M       3.4G
Swap:          0B         0B         0B

elliot@ubuntu-linux:~$ date
Mon Nov  4 13:35:59 CST 2019
November 2019
Su Mo Tu We Th Fr Sa
      1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
```

Removing an alias

- Let's create another temporary alias named lastline that will display the last line in a file:

```
elliot@ubuntu-linux:~$ alias lastline="tail -n 1"
```

- Now let's try our new alias on the /home/elliot/.bashrc file:

```
elliot@ubuntu-linux:~$ lastline /home/elliot/.bashrc  
alias date="date;cal"
```

- Alright! It works well, Now, if you wish to delete the alias, then you can run the unalias command followed by the alias name:

```
elliot@ubuntu-linux:~$ unalias lastline
```

Removing an alias

- Now the lastline alias has been deleted:

```
elliott@ubuntu-linux:~$ lastline /home/elliott/.bashrc  
lastline: command not found
```

- You can also use the unalias command to temporarily deactivate a permanent alias, For example, if you run the following command:

```
elliott@ubuntu-linux:~$ unalias memory
```

- Now, the permanent alias memory will not work in the current Terminal session:

```
elliott@ubuntu-linux:~$ memory
```

Command 'memory' not found, did you mean:

 command 'lmemory' from deb lmemory

Try: sudo apt install <deb name>

Adding safety nets

- You can also use aliases to protect against dumb mistakes.
- For example, to protect against removing important files by mistake, you can add the following alias:

```
elliot@ubuntu-linux:~$ alias rm="rm -i"
```

- Now you will be asked to confirm each time you attempt to remove a file:

```
elliot@ubuntu-linux:~$ rm *
rm: remove regular file '3x10.sh'?
```

Go crazy with aliases

- You can also have some fun with aliases and make users go crazy; take a look at this alias:

```
elliot@ubuntu-linux:~$ alias nano="vi"
```

- Now when user elliot tries to open the nano editor, the vi editor will open instead! User elliot can overcome this dilemma by typing in the full path of the nano editor.
- Here is another funny alias:

```
elliot@ubuntu-linux:~$ alias exit="echo No I am not exiting ..."
```

Removing an alias

- Now look what will happen when user elliot tries to exit the Terminal:

```
elliot@ubuntu-linux:~$ exit
```

```
No I am not exiting ...
```

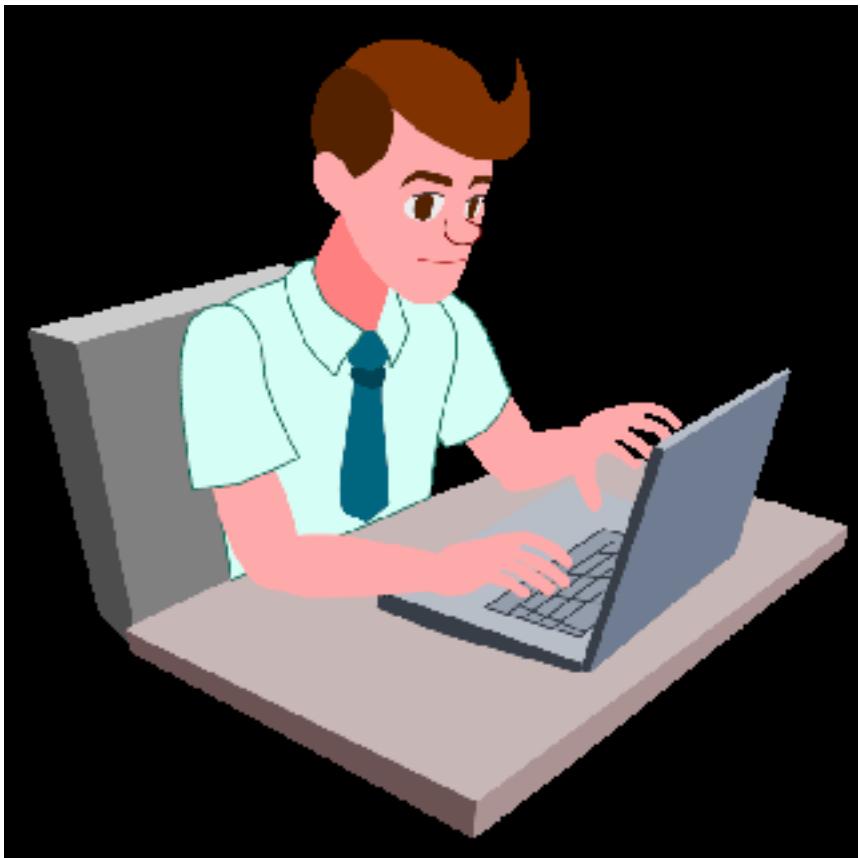
```
elliot@ubuntu-linux:~$ exit
```

```
No I am not exiting ...
```

Knowledge check

For the following exercises, open up your Terminal and try to solve the following tasks:

1. Create a temporary alias called ins for the apt-get install command.
2. Create a temporary alias called packages for the dpkg -l command.
3. Create a permanent alias called clean that will remove all the files in the /tmp directory.



"Complete Lab"



20. File and Disk Management Tools

Where are your devices?

- The terminal you are working on right now is, in fact, a device.
- If you run the w command, you will see the name of the terminal you are connected to in the second column of the output.

```
elliot@ubuntu-linux:~$ w
11:38:59 up 17 min, 1 user, load average: 0.00, 0.00, 0.02
USER  TTY      FROM          LOGIN@ IDLE JCPU PCPU WHAT
elliot pts/0    172.16.1.67    11:22  0.00s 0.06s 0.00s w
```

Where are your devices?

- In my case, it is pts/0; pts is short for pseudoterminal slave.

- Now, this terminal is represented by the file /dev/pts/0:

```
elliott@ubuntu-linux:~$ ls -l /dev/pts/0  
crw----- 1 elliot tty 136, 0 Nov 7 11:40 /dev/pts/0
```

- I will echo the line Hello Friend to /dev/pts/0 and pay close attention to what will happen:

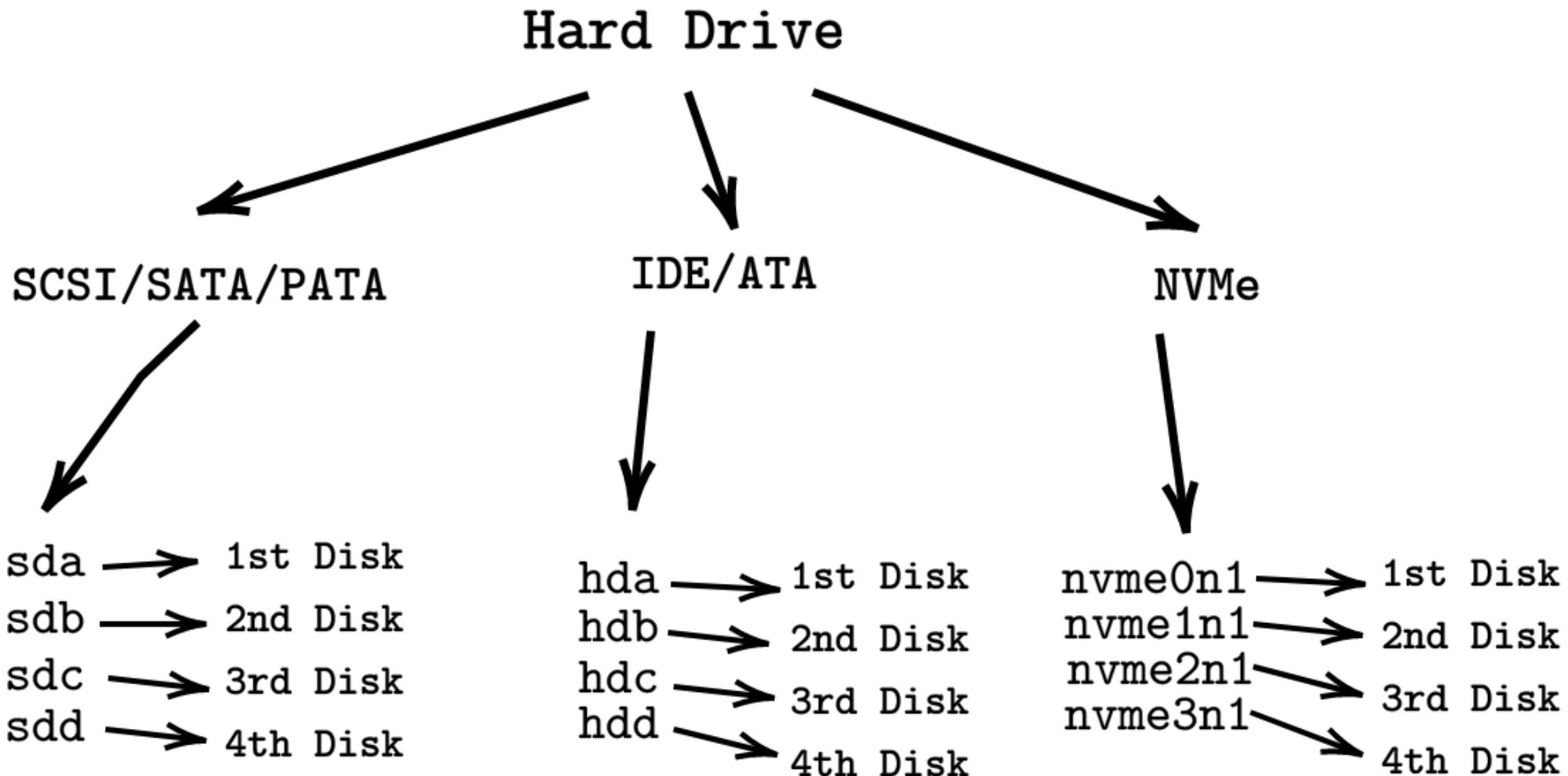
```
elliott@ubuntu-linux:~$ echo "Hello Friend" > /dev/pts/0  
Hello Friend
```

Where is your hard disk?

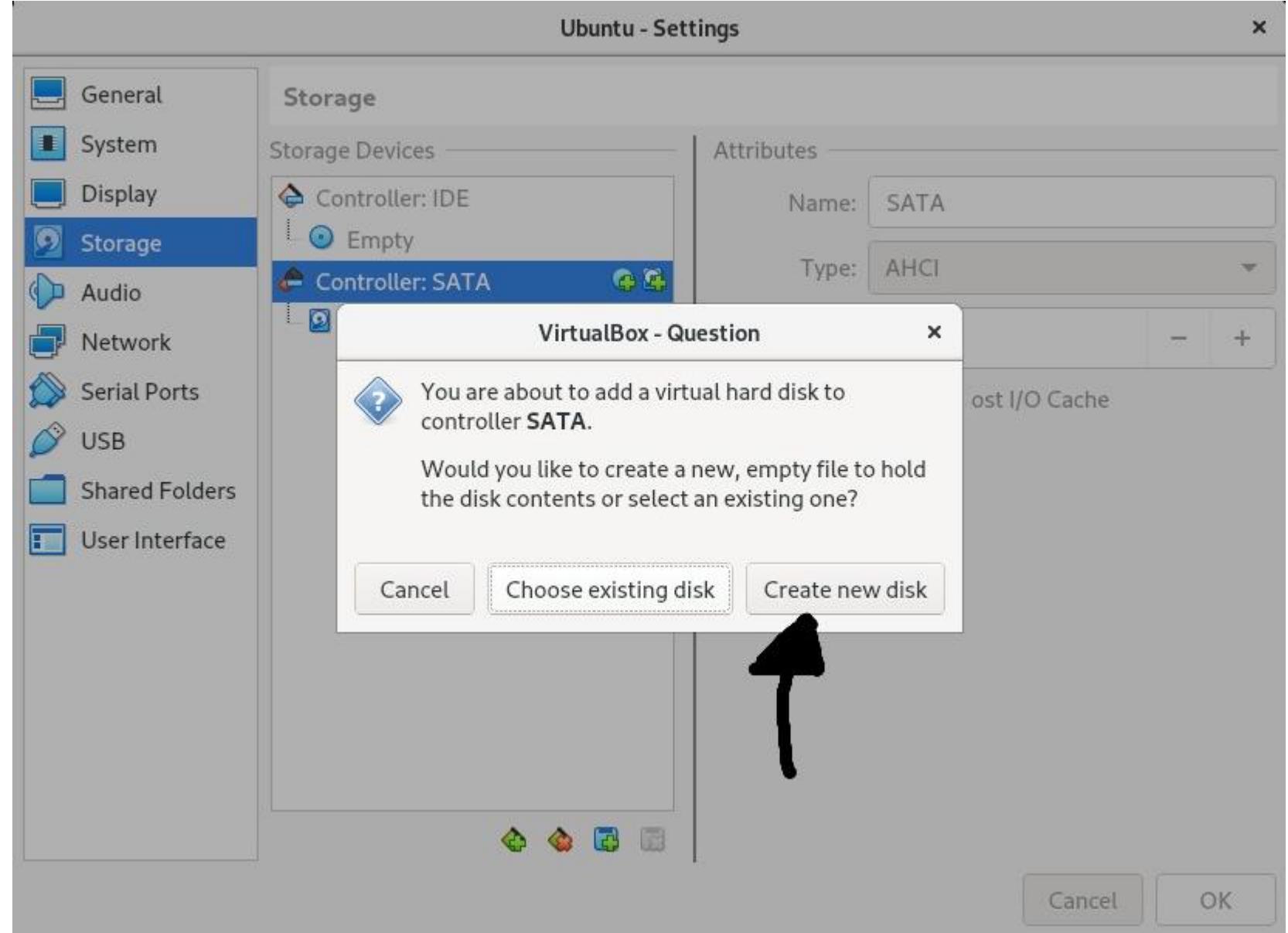
- To know which file represents your hard disk; you need to run the command lsblk, which is short for list block:

```
elliott@ubuntu-linux:~$ lsblk
NAME  MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda    8:0    0 20G  0 disk
| sda1  8:1    0 20G  0 part /
sr0    11:0   1 1024M 0 rom
```

Where is your hard disk?



Adding disks to your virtual machine



Adding disks to your virtual machine

- You should be able to see your new disk as soon as your virtual machine starts:

```
elliott@ubuntu-linux:~$ lsblk
NAME  MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda    8:0    0 20G  0 disk
| sda1   8:1    0 20G  0 part /
sdb    8:16   0 10G  0 disk
sr0    11:0   1 1024M 0 rom
```

Creating new disk partitions

- The first thing you may want to do is to create a new partition.
- To create a new partition, we use the fdisk command followed by the disk name:
`fdisk [options] device`
- To create a new partition on the /dev/sdb disk; you can run the following command:
`root@ubuntu-linux:~# fdisk /dev/sdb`

Welcome to fdisk (util-linux 2.31.1).

Changes will remain in memory only, until you decide to write them.

Be careful before using the write command.

Device does not contain a recognized partition table.

Created a new DOS disklabel with disk identifier 0xb13d9b6a.

Command (m for help):

Creating new disk partitions

- Opens up the fdisk utility.
- If you are unsure what to do; you can enter m for help:

```
Command (m for help): m
Help:
  DOS (MBR)
    a  toggle a bootable flag
    b  edit nested BSD disklabel
    c  toggle the dos compatibility flag

  Generic
    d  delete a partition
    F  list free unpartitioned space l  list known partition types
    n  add a new partition
    p  print the partition table t  change a partition type
    v  verify the partition table
    i  print information about a partition

  Save & Exit
    w  write table to disk and exit
    q  quit without saving changes

  Create a new label
    g  create a new empty GPT partition table
    G  create a new empty SGI (IRIX) partition table
    o  create a new empty DOS  partition table
    s  create a new empty Sun partition table
```

Creating new disk partitions

- We want to create a new partition so enter n:

Command (m for help): n

Partition type

p primary (0 primary, 0 extended, 4 free)

e extended (container for logical partitions)

Select (default p):

- It will then ask you if you want a primary partition or an extended partition.

- We would accept the default selection (primary) so just hit Enter:

Using default response p.

Partition number (1-4, default 1):

Creating new disk partitions

- It will then ask you to select a partition number.
- We will also accept the default, which is partition number 1, so just hit Enter.
- Notice that you can create up to four primary partitions on a given disk:

Partition number (1-4, default 1):

First sector (2048-20971519, default 2048):

- You will then be prompted to choose the sector you would want your new partition to start at; hit Enter to accept the default (2048):

First sector (2048-20971519, default 2048):

Last sector, +sectors or +size{K,M,G,T,P} (2048-20971519, default 20971519):

Creating new disk partitions

- Now you will be asked to choose the size of your new partition; I want a 2 GB partition so I would type +2G and then hit Enter:

Last sector, +sectors or +size{K,M,G,T,P} (2048-20971519, default 20971519): +2G

Created a new partition 1 of type 'Linux' and of size 2 GiB.

Command (m for help):

- Finally, you have to save the configuration by hitting w:

Command (m for help): w

The partition table has been altered.

Calling ioctl() to re-read partition table.

Syncing disks.

Creating new disk partitions

- Now you can run lsblk to see the new partition you just created:

```
root@ubuntu-linux:~# lsblk
NAME  MAJ:MIN RM SIZE  RO TYPE MOUNTPOINT
sda    8:0    0 20G  0 disk
| sda1  8:1    0 20G  0 part /
sdb    8:16   0 10G  0 disk
| sdb1  8:17   0  2G  0 part
sr0   11:0   1 1024M 0 rom
```

Creating new disk partitions

- You can see the 2 GB partition sdb1 is listed under sdb.
- You can also use the -l option with the fdisk command to print out the partition table of your disk:

```
root@ubuntu-linux:~# fdisk -l /dev/sdb
```

```
Disk /dev/sdb: 10 GiB, 10737418240 bytes, 20971520 sectors
```

```
Units: sectors of 1 * 512 = 512 bytes
```

```
Sector size (logical/physical): 512 bytes / 512 bytes
```

```
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

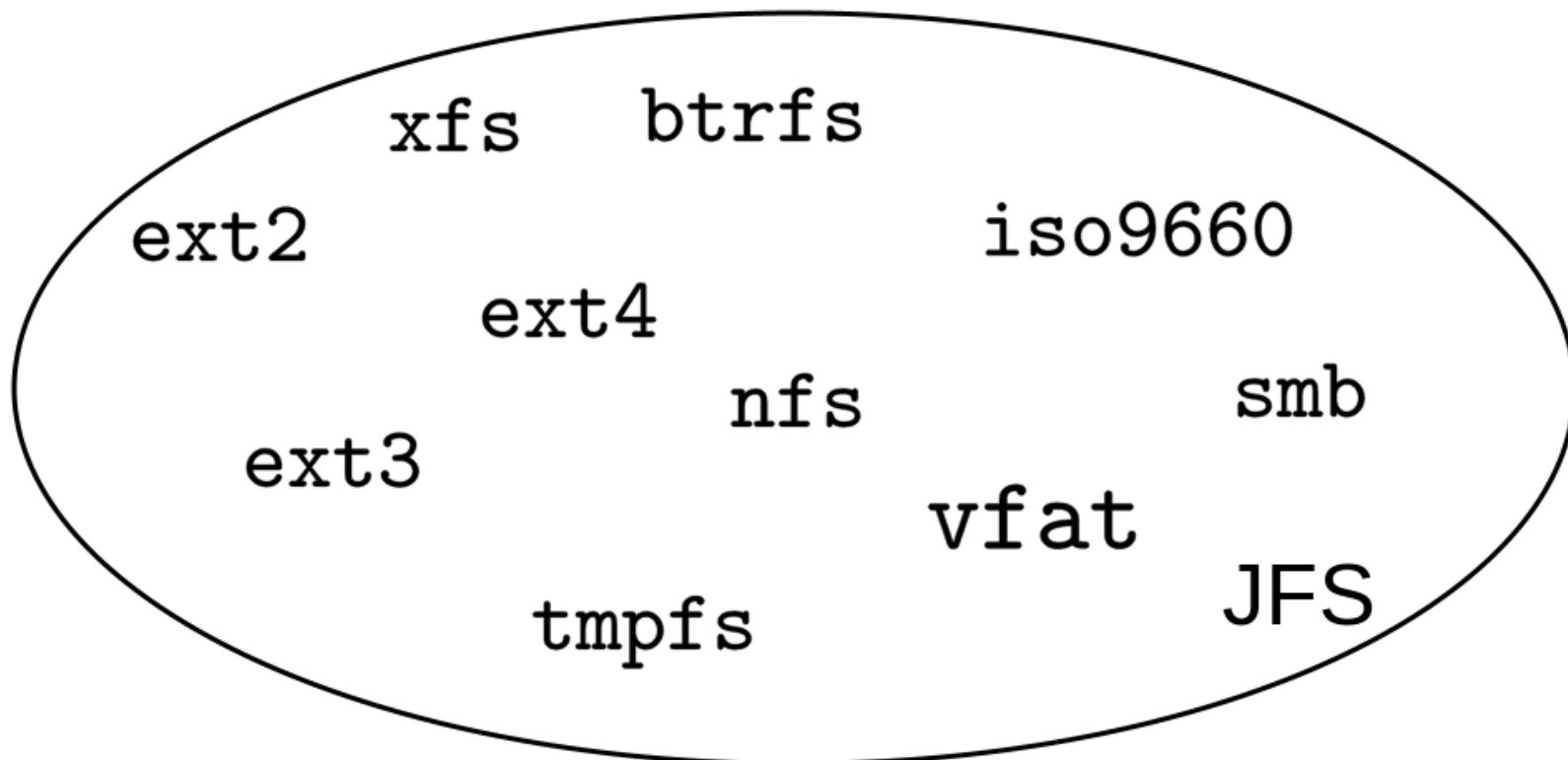
```
Disklabel type: dos
```

```
Disk identifier: 0xb13d9b6a
```

Device	Boot	Start	End	Sectors	Size	Id	Type
/dev/sdb1		2048	4196351	4194304	2G	83	Linux

Creating new filesystems

Linux filesystems



Creating new filesystems

- You can read the description of each Linux filesystem type in the filesystems man page:

```
root@ubuntu-linux:~# man filesystems
```

- To create a filesystem, we use the mkfs command, which is short for make filesystem.
- The general syntax for the mkfs command is as follows:

```
mkfs --type [fstype] disk_or_partition
```

Creating new filesystems

- Now let's create an ext4 filesystem on our new partition /dev/sdb1:

```
root@ubuntu-linux:~# mkfs --type ext4 /dev/sdb1  
mke2fs 1.44.1 (24-Mar-2018)
```

Creating filesystem with 524288 4k blocks and 131072 inodes

Filesystem UUID: 61d947bb-0cd1-41e1-90e0-c9895b6de428

Superblock backups stored on blocks:

32768, 98304, 163840, 229376, 294912

Allocating group tables: done

Writing inode tables: done

Creating journal (16384 blocks): done

Writing superblocks and filesystem accounting information: done

Creating new filesystems

- We have created an ext4 filesystem on our partition /dev/sdb1.
- We can verify our work by running the file -s command on the /dev/sdb1 partition:

```
root@ubuntu-linux:~# file -s /dev/sdb1
/dev/sdb1: Linux rev 1.0 ext4 filesystem data,
UUID=61d947bb-0cd1-41e1-90e0-c9895b6de428 (extents) (64bit) (large
files) (huge files)
```

Creating new filesystems

- You can use the wipefs command to remove (wipe out) a filesystem.
- For example, if you want to remove the ext4 filesystem that we just created on /dev/sdb1, you can run the following command:

```
root@ubuntu-linux:~# wipefs -a /dev/sdb1  
/dev/sdb1: 2 bytes were erased at offset 0x00000438 (ext4): 53 ef
```

- Now if you rerun file -s on the /dev/sdb1 partition, you will see there is no filesystem signature:

```
root@ubuntu-linux:~# file -s /dev/sdb1  
/dev/sdb1: data
```

Creating new filesystems

- Let's recreate an ext4 filesystem on /dev/sdb1 and keep it this time around:

```
root@ubuntu-linux:~# mkfs --type ext4 /dev/sdb1
```

```
mke2fs 1.44.1 (24-Mar-2018)
```

```
Creating filesystem with 524288 4k blocks and 131072 inodes
```

```
Filesystem UUID: 811aef62-d9ca-4db3-b305-bd896d1c8545
```

```
Superblock backups stored on blocks:
```

```
32768, 98304, 163840, 229376, 294912
```

```
Allocating group tables: done
```

```
Writing inode tables: done
```

```
Creating journal (16384 blocks): done
```

```
Writing superblocks and filesystem accounting information: done
```

Mounting filesystems

- To mount a filesystem, we use the mount command as follows:

```
mount filesystem mount_directory
```

- So let's assume we are going to use the filesystem /dev/sdb1 to store our games.
- In this case, let's create a new directory /games:

```
root@ubuntu-linux:~# mkdir /games
```

- Now the only thing left is to mount our filesystem /dev/sdb1 on the /games directory:

```
root@ubuntu-linux:/# mount /dev/sdb1 /games
```

Mounting filesystems

- We can verify our work by running the lsblk command:

```
root@ubuntu-linux:~# lsblk
NAME  MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda    8:0    0 20G  0 disk
| sda1  8:1    0 20G  0 part /
sdb    8:16   0 10G  0 disk
| sdb1  8:17   0  2G  0 part /games
sr0    11:0   1 1024M 0 rom
```

Mounting filesystems

- You can also use the mount command by itself to list all the mounted filesystems on your system.
- For example, to verify that /dev/sdb1 is mounted on /games, you can run the following command:

```
root@ubuntu-linux:/# mount | grep sdb1  
/dev/sdb1 on /games type ext4 (rw,relatime,data=ordered)
```

Mounting filesystems

- We now have 2 GB available for us to use in /games and you can use the df command to display the filesystem disk space usage:

```
root@ubuntu-linux:~# df -h /games
```

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/sdb1	2.0G	6.0M	1.8G	1%	/games

- Now let's create three files in /games:

```
root@ubuntu-linux:~# cd /games
```

```
root@ubuntu-linux:/games# touch game1 game2 game3
```

Unmounting filesystems

- You can also unmount (the reverse of mounting) a filesystem, As you may have guessed, unmounting refers to the process of detaching a filesystem or a storage device.
- To unmount a filesystem, you can use umount as follows:
umount filesystem
- Change to the /games directory and try to unmount the /dev/sdb1 filesystem:

```
root@ubuntu-linux:/games# umount /dev/sdb1  
umount: /games: target is busy.
```

Unmounting filesystems

- Oops! It is saying that the target is busy! That's because I am inside the mount point /games; I will back up one directory and then try again:

```
root@ubuntu-linux:/games# cd ..  
root@ubuntu-linux:# umount /dev/sdb1
```

Unmounting filesystems

- Now let's verify the filesystem /dev/sdb1 is indeed unmounted:

```
root@ubuntu-linux:/# lsblk
```

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINT
------	---------	----	------	----	------	------------

sda	8:0	0	20G	0	disk	
-----	-----	---	-----	---	------	--

sda1	8:1	0	20G	0	part	/
------	-----	---	-----	---	------	---

sdb	8:16	0	10G	0	disk	
-----	------	---	-----	---	------	--

sdb1	8:17	0	2G	0	part	
------	------	---	----	---	------	--

sr0	11:0	1	1024M	0	rom	
-----	------	---	-------	---	-----	--

```
root@ubuntu-linux:/# mount | grep sdb1
```

- Yup! It is definitely unmounted! Now let's list the contents of the /games directory:

```
root@ubuntu-linux:/# ls /games
```

Unmounting filesystems

- Nothing! But do not panic or worry! The three files we created still exist in the /dev/sdb1 filesystem.
- We need to mount the filesystem again, and you will see the files:

```
root@ubuntu-linux:~# mount /dev/sdb1 /games
root@ubuntu-linux:~# ls /games
game1 game2 game3 lost+found
```

Permanently mounting filesystems

- For example, to mount our `/dev/sdb1` filesystem on `/games` permanently, you need to include the following line in `/etc/fstab`:

```
/dev/sdb1 /games ext4 defaults 0 0
```

- You should add the line to the end of the `/etc/fstab` file:

```
root@ubuntu-linux:~# tail -1 /etc/fstab  
/dev/sdb1 /games ext4 defaults 0 0
```

- Now let's unmount `/dev/sdb1`:

```
root@ubuntu-linux:~# umount /dev/sdb1
```

Permanently mounting filesystems

- Finally, you can now mount /dev/sdb1 permanently by running:

```
root@ubuntu-linux:~# mount /dev/sdb1
```

- Notice we did not specify a mount destination this time; that's because the mount destination is already specified in the /etc/fstab file.
- You can use the -a option with the mount command:

```
root@ubuntu-linux:~# mount -a
```

Permanently mounting filesystems

- To mount all the filesystems that are included in /etc/fstab. It is also used to check for syntax errors.
- For example, if you made a typo in /etc/fstab and wrote /dev/sdx1 instead of /dev/sdb1, it will show you the following error:

```
root@ubuntu-linux:~# mount -a  
mount: /games: special device /dev/sdx1 does not exist.
```

- All the mounts specified in /etc/fstab are permanent and they will survive a system reboot.
- You may also refer to the fstab man page for more information on /etc/fstab:

```
root@ubuntu-linux:~# man fstab
```

Running out of space

- A fast way to create big files in Linux is by using the dd command.
- To demonstrate, let's first change to the /games directory:

```
root@ubuntu-linux:~# cd /games
```

```
root@ubuntu-linux:/games#
```

- Now you can run the following command to create a 1 GB file named bigGame:

```
root@ubuntu-linux:/games# dd if=/dev/zero of=bigGame bs=1G count=1
1+0 records in
1+0 records out
1073741824 bytes (1.1 GB, 1.0 GiB) copied, 1.44297 s, 744 MB/s
```

Running out of space

- We have now already used more than half of the available space in /games:

```
root@ubuntu-linux:/games# df -h /games
```

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/sdb1	2.0G	1.1G	868M	55%	/games

- Now let's attempt to create another file named bigFish of size 3 GB:

```
root@ubuntu-linux:/games# dd if=/dev/zero of=bigFish bs=1G count=3
```

```
dd: error writing 'bigFish': No space left on device
```

```
1+0 records in
```

```
0+0 records out
```

```
1016942592 bytes (1.0 GB, 970 MiB) copied, 1.59397 s, 638 MB/s
```

Running out of space

- We got an error as we ran out of space:

```
root@ubuntu-linux:/games# df -h /games
```

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/sdb1	2.0G	2.0G	0	100%	/games

- Now we can't even create a tiny file with the word Hello in it:

```
root@ubuntu-linux:/games# echo Hello > greeting.txt
```

```
-su: echo: write error: No space left on device
```

Corrupting and fixing filesystems

- The following command will surely corrupt your /dev/sdb1 filesystem:

```
root@ubuntu-linux:/games# dd if=/dev/urandom of=/dev/sdb1  
count=10k
```

- Your /dev/sdb1 filesystem is now corrupted! If you don't believe me, unmount it and then try to mount it back again:

```
root@ubuntu-linux:~# umount /dev/sdb1
```

- OK, it unmounted successfully! Let's see if it will mount:

```
root@ubuntu-linux:~# mount /dev/sdb1 /games  
mount: /games: wrong fs type, bad option, bad superblock on /dev/sdb1,  
missing codepage or helper program, or other error.
```

Corrupting and fixing filesystems

- You can use the file system check command fsck to check and repair filesystems.
- So let's run fsck on our corrupted filesystem:

```
root@ubuntu-linux:~# fsck /dev/sdb1
```

```
fsck from util-linux 2.31.1
```

```
e2fsck 1.44.1 (24-Mar-2018)
```

```
/dev/sdb1 was not cleanly unmounted, check forced.
```

```
fsck.ext4: Inode checksum does not match inode while reading bad  
blocks inode
```

This doesn't bode well, but we'll try to go on...

Pass 1: Checking inodes, blocks, and sizes

Inode 1 seems to contain garbage. Clear<y>?

Corrupting and fixing filesystems

- You can avoid that by using the -y option, which answers an automatic yes to all prompts during the repair process:

```
root@ubuntu-linux:~# fsck -y /dev/sdb1
```

- After it finishes, you can rerun fsck to verify the filesystem is now clean:

```
root@ubuntu-linux:~# fsck /dev/sdb1
fsck from util-linux 2.31.1
e2fsck 1.44.1 (24-Mar-2018)
/dev/sdb1: clean, 11/131072 files, 9769/524288 blocks
```

Installing the LVM package

- Before we start playing with LVM, first, we need to install the lvm2 package:

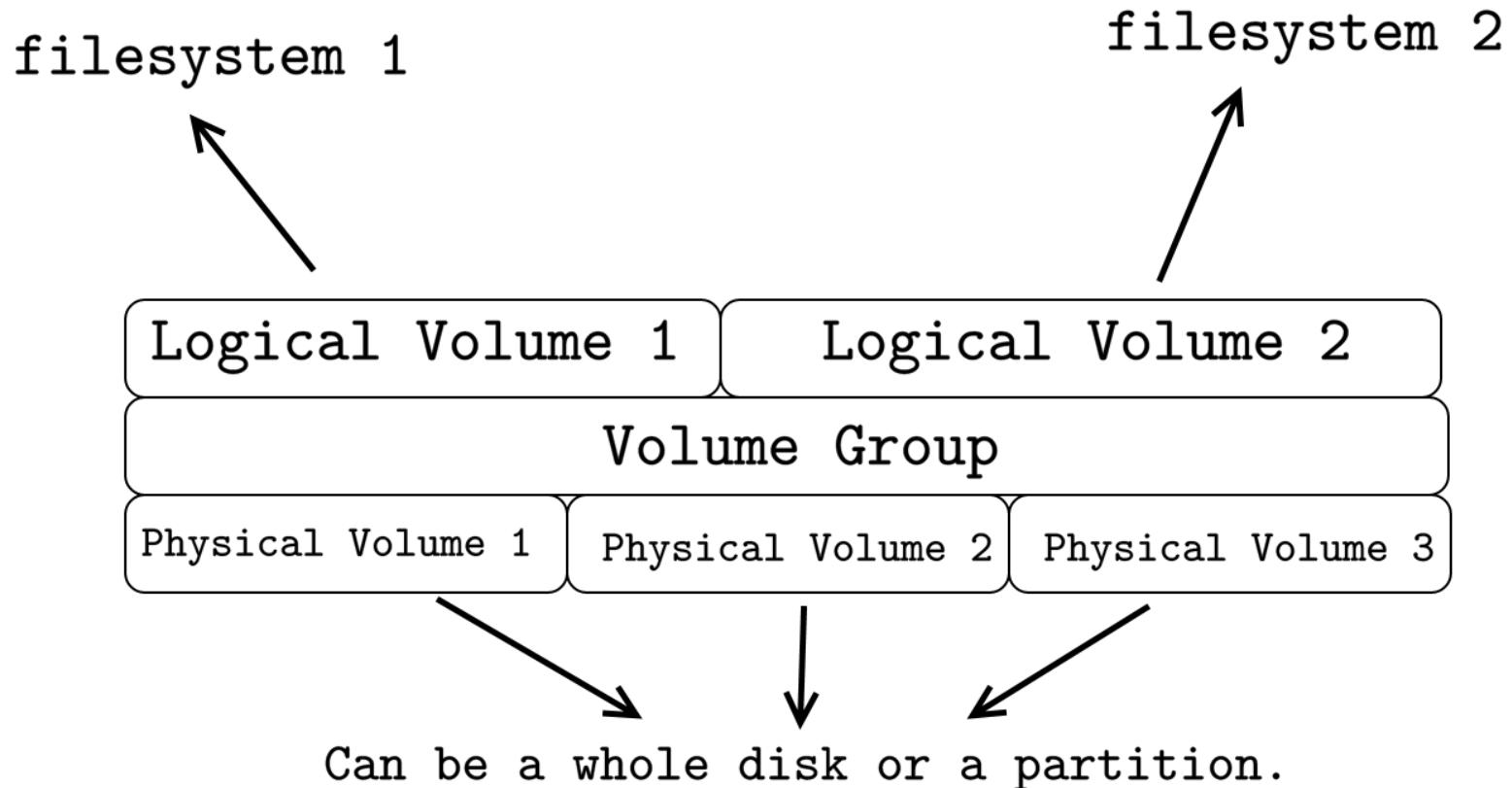
```
root@ubuntu-linux:~# apt-get install lvm2
```

- After the installation is complete, you can run the lvm version command to verify the installation is successful:

```
root@ubuntu-linux:~# lvm version
LVM version: 2.02.176(2) (2017-11-03)
Library version: 1.02.145 (2017-11-03)
Driver version: 4.37.0
```

Three layers of abstraction

- To understand how LVM works, you first need to visualize it.
- LVM is like a cake that is made up of three layers, as shown in Figure.



Creating physical volumes

- The recipe for creating physical volumes is pretty simple; you only need a disk or a partition.
- We have already created a 2 GB partition /dev/sdb1. Now go ahead and create three more partitions under /dev/sdb, each of size 2 GB.
- This is what the end result should look like:

```
root@ubuntu-linux:~# lsblk
NAME   MAJ:MIN   RM   SIZE   RO   TYPE MOUNTPOINT
sda      8:0     0    20G   0   disk
|_ sda1    8:1     0    20G   0   part /
sdb      8:16    0    10G   0   disk
|_ sdb1    8:17    0    2G   0   part /games
|_ sdb2    8:18    0    2G   0   part
|_ sdb3    8:19    0    2G   0   part
|_ sdb4    8:20    0    2G   0   part
sr0     11:0    1  1024M  0   rom
```

Creating physical volumes

- To create a physical volume, we use the pvcreate command followed by a disk or a partition:

`pvcreate disk_or_partition`

- We are going to create three physical volumes: /dev/sdb2, /dev/sdb3, and /dev/sdb4. You can create all three with one command:

```
root@ubuntu-linux:~# pvcreate /dev/sdb2 /dev/sdb3 /dev/sdb4
```

Physical volume "/dev/sdb2" successfully created.

Physical volume "/dev/sdb3" successfully created.

Physical volume "/dev/sdb4" successfully created.

Creating physical volumes

- Cool stuff! You can also use the pvs command to list all physical volumes:

```
root@ubuntu-linux:~# pvs
  PV   VG Fmt Attr PSize PFree
/dev/sdb2  lvm2 --- 2.00g 2.00g
/dev/sdb3  lvm2 --- 2.00g 2.00g
/dev/sdb4  lvm2 --- 2.00g 2.00g
```

Creating volume groups

- To create a volume group, we use the vgcreate command followed by the name of the new volume group and then the physical volumes:

```
vgcreate vg_name PV1 PV2 PV3 ...
```

- Let's create a volume group named myvg that would span /dev/sdb2 and /de- v/sdb3:

```
root@ubuntu-linux:~# vgcreate myvg /dev/sdb2 /dev/sdb3
```

```
Volume group "myvg" successfully created
```

Creating volume groups

- Awesome! You can also use the vgs command to list all volume groups:

```
root@ubuntu-linux:~# vgs
  VG #PV #LV #SN Attr  VSize VFree
myvg  2  0  0 wz--n- 3.99g 3.99g
```

Creating logical volumes

- To create a logical volume, we use the lvcreate command followed by the size of the logical volume, the name of the logical volume, and finally, the volume group name:

```
lvcreate --size 2G --name lv_name vg_name
```

- Let's create a logical volume named mybooks of size 2 GB:

```
root@ubuntu-linux:~# lvcreate --size 2G --name mybooks myvg  
Logical volume "mybooks" created.
```

- Now create another logical volume named myprojects of size 500 MB:

```
root@ubuntu-linux:~# lvcreate --size 500M --name myprojects myvg  
Logical volume "myprojects" created.
```

Creating logical volumes

- You can use the `lvs` command to list all logical volumes:

```
root@ubuntu-linux:~# lvs
```

LV	VG	Attr	LSize	Pool	Origin	Data%	Meta%	Move	Log
mybooks	myvg	-wi-a-----	2.00g						
myprojects	myvg	-wi-a-----	500.00m						

- One final step remains, which is creating filesystems on our logical volumes.
- Your logical volumes are represented in the device mapper directory `/dev/mapper`:

```
root@ubuntu-linux:~# ls /dev/mapper  
myvg-mybooks myvg-myprojects
```

Creating logical volumes

- Let's create an ext4 filesystem on our mybooks logical volume:

```
root@ubuntu-linux:~# mkfs --type ext4 /dev/mapper/myvg-mybooks  
mke2fs 1.44.1 (24-Mar-2018)
```

Creating filesystem with 524288 4k blocks and 131072 inodes

Filesystem UUID: d1b43462-6d5c-4329-b027-7ee2ecef9a

Superblock backups stored on blocks:

32768, 98304, 163840, 229376, 294912

Allocating group tables: done

Writing inode tables: done

Creating journal (16384 blocks): done

Writing superblocks and filesystem accounting information: done

Creating logical volumes

- Similarly, we can create an ext4 filesystem on our myprojects logical volume:

```
root@ubuntu-linux:~# mkfs --type ext4 /dev/mapper/myvg-myprojects  
mke2fs 1.44.1 (24-Mar-2018)
```

Creating filesystem with 512000 1k blocks and 128016 inodes

Filesystem UUID: 5bbb0826-c845-4ef9-988a-d784cc72f258

Superblock backups stored on blocks:

8193, 24577, 40961, 57345, 73729, 204801, 221185, 401409

Allocating group tables: done

Writing inode tables: done

Creating journal (8192 blocks): done

Writing superblocks and filesystem accounting information: done

Creating logical volumes

- We have to mount both filesystems somewhere so we will create two new directories, /books and /projects:

```
root@ubuntu-linux:~# mkdir /books /projects
```

- Now we can mount both filesystems:

```
root@ubuntu-linux:~# mount /dev/mapper/myvg-mybooks /books
```

```
root@ubuntu-linux:~# mount /dev/mapper/myvg-myprojects /projects
```

- We can check the last two lines of the mount command output:

```
root@ubuntu-linux:~# mount | tail -n 2
```

```
/dev/mapper/myvg-mybooks on /books type ext4 (rw,relatime,data=ordered)
```

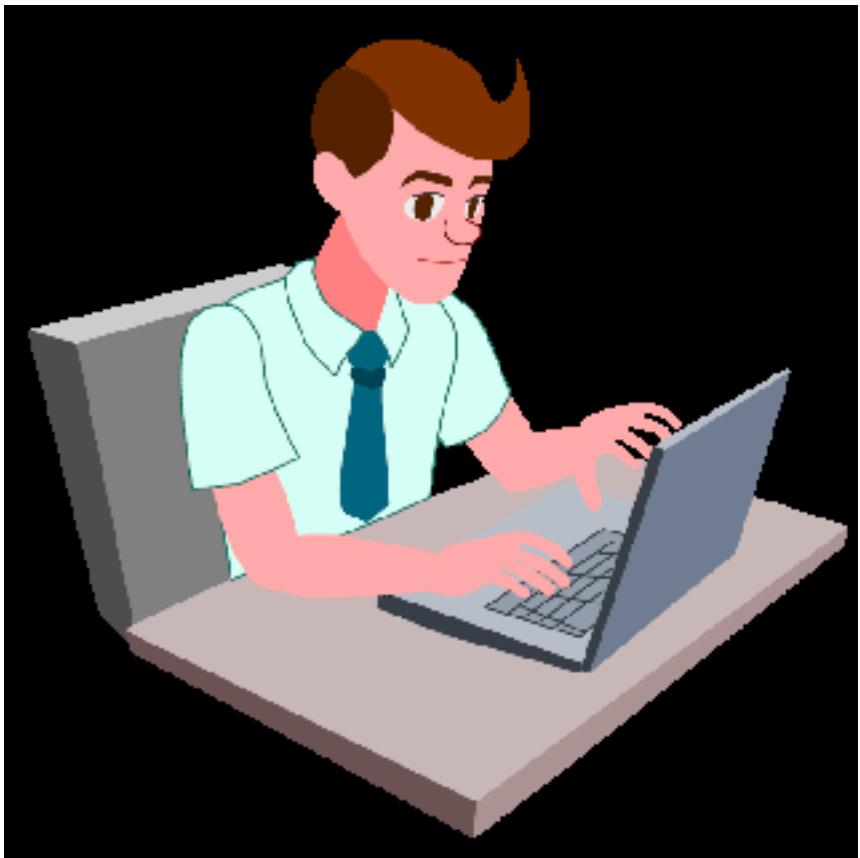
```
/dev/mapper/myvg-myprojects on /projects type ext4
```

```
(rw,relatime,data=ordered)
```

Knowledge check

For the following exercises, open up your Terminal and try to solve the following tasks:

1. Add a new 1 GB disk to your virtual machine.
2. Create three 250 MB partitions on your new disk.
3. Use your three new partitions to create three physical volumes.
4. Create a volume group named `bigvg` that spans all your three physical volumes.
5. Create a logical volume named `biglv` of size 500 MB.
6. Create an ext4 filesystem on the `biglv` logical volume.
7. Mount your filesystem on the `/mnt/wikileaks` directory.



"Complete Lab"

21. LAMP Server Basics

Introduction

- A “LAMP” stack is a group of open source software that is typically installed together in order to enable a server to host dynamic websites and web apps written in PHP.
- This term is an acronym which represents the Linux operating system, with the Apache web server.
- The site data is stored in a MySQL database, and dynamic content is processed by PHP.

Step 1: Installing Apache and Updating the Firewall

- Start by updating the package manager cache.
- If this is the first time you're using sudo within this session, you'll be prompted to provide your user's password to confirm you have the right privileges to manage system packages with apt.

`sudo apt update`

- Then, install Apache with:

`sudo apt install apache2`

Step 1: Installing Apache and Updating the Firewall

- Once the installation is finished, you'll need to adjust your firewall settings to allow HTTP traffic.
- UFW has different application profiles that you can leverage for accomplishing that.
- To list all currently available UFW application profiles, you can run:
`sudo ufw app list`
- You'll see output like this:

[Output](#)

[Available applications:](#)

[Apache](#)

[Apache Full](#)

[Apache Secure](#)

[OpenSSH](#)

Step 1: Installing Apache and Updating the Firewall

- To only allow traffic on port 80, use the Apache profile:

```
sudo ufw allow in "Apache"
```

- You can verify the change with:

```
sudo ufw status
```

Output

Status: active

To	Action	From
--	-----	-----
OpenSSH	ALLOW	Anywhere
Apache	ALLOW	Anywhere
OpenSSH (v6)	ALLOW	Anywhere (v6)
Apache (v6)	ALLOW	Anywhere (v6)

Step 1: Installing Apache and Updating the Firewall

- You can do a spot check right away to verify that everything went as planned by visiting your server's public IP address in your web browser (see the note under the next heading to find out what your public IP address is if you do not have this information already):

http://your_server_ip

Step 1: Installing Apache and Updating the Firewall

- You'll see the default Ubuntu 20.04 Apache web page, which is there for informational and testing purposes.
- It should look something like this:



Apache2 Ubuntu Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in `/usr/share/doc/apache2/README.Debian.gz`**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|   '-- ports.conf
|-- mods-enabled
|   '-- *.load
|   '-- *.conf
|-- conf-enabled
|   '-- *.conf
-- sites-enabled
    '-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective `*-available/` counterparts. These should be managed by using our helpers `a2enmod`, `a2dismod`, `a2ensite`, `a2dissite`, and `a2enconf`, `a2disconf`. See their respective man pages for detailed information.
- The binary is called `apache2`. Due to the use of environment variables, in the default configuration, `apache2` needs to be started/stopped with `/etc/init.d/apache2` or `apache2ctl`. Calling `/usr/bin/apache2` directly will not work with the default configuration.

Document Roots

By default, Ubuntu does not allow access through the web browser to *any* file apart of those located in `/var/www`, `public_html` directories (when enabled) and `/usr/share` (for web applications). If your site is using a web document root located elsewhere (such as in `/srv`) you may need to whitelist your document root directory in `/etc/apache2/apache2.conf`.

The default Ubuntu document root is `/var/www/html`. You can make your own virtual hosts under `/var/www`. This is different to previous releases which provides better security out of the box.

Reporting Problems

Please use the `ubuntu-bug` tool to report bugs in the Apache2 package with Ubuntu. However, check **existing bug reports** before reporting a new bug.

Please report bugs specific to modules (such as PHP and others) to respective packages, not to the web server itself.

How To Find your Server's Public IP Address

- There are a few different ways to do this from the command line.
- First, you could use the iproute2 tools to get your IP address by typing this:

```
ip addr show eth0 | grep inet | awk '{ print $2; }' | sed 's/\.*$//'
```

- This will give you two or three lines back, They are all correct addresses, but your computer may only be able to use one of them, so feel free to try each one.
- An alternative method is to use the curl utility to contact an outside party to tell you how it sees your server, This is done by asking a specific server what your IP address is:

```
curl http://icanhazip.com
```

Step 2 — Installing MySQL

- Again, use apt to acquire and install this software:

```
sudo apt install mysql-server
```

- When prompted, confirm installation by typing Y, and then ENTER.
- When the installation is finished, it's recommended that you run a security script that comes pre-installed with MySQL.
- This script will remove some insecure default settings and lock down access to your database system. Start the interactive script by running:

```
sudo mysql_secure_installation
```

Step 2 — Installing MySQL

- Answer Y for yes, or anything else to continue without enabling.

VALIDATE PASSWORD PLUGIN can be used to test passwords and improve security. It checks the strength of password and allows the users to set only those passwords which are secure enough. Would you like to setup VALIDATE PASSWORD plugin?

Press y|Y for Yes, any other key for No:

Step 2 — Installing MySQL

- Keep in mind that if you enter 2 for the strongest level, you will receive errors when attempting to set any password which does not contain numbers, upper and lowercase letters, and special characters, or which is based on common dictionary words.

There are three levels of password validation policy:

LOW Length ≥ 8

MEDIUM Length ≥ 8 , numeric, mixed case, and special characters

STRONG Length ≥ 8 , numeric, mixed case, special characters and dictionary file

Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG: 1

Step 2 — Installing MySQL

- If you enabled password validation, you'll be shown the password strength for the root password you just entered and your server will ask if you want to continue with that password.
- If you are happy with your current password, enter Y for "yes" at the prompt:

Estimated strength of the password: 100

Do you wish to continue with the password provided?(Press y|Y for Yes, any other key for No) : y

Step 2 — Installing MySQL

When you're finished, test if you're able to log in to the MySQL console by typing:

```
$ sudo mysql
```

This will connect to the MySQL server as the administrative database user **root**, which is inferred by the use of `sudo` when running this command. You should see output like this:

Output

```
Welcome to the MySQL monitor. Commands end with ; or \g.
```

```
Your MySQL connection id is 22
```

```
Server version: 8.0.19-0ubuntu5 (Ubuntu)
```

```
Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.
```

```
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

```
mysql>
```

To exit the MySQL console, type:

```
mysql> exit
```

Step 3 — Installing PHP

- To install these packages, run:

```
sudo apt install php libapache2-mod-php php-mysql
```

- Once the installation is finished, you can run the following command to confirm your PHP version:

```
php -v
```

Output

```
PHP 7.4.3 (cli) (built: Jul 5 2021 15:13:35) ( NTS )
```

```
Copyright (c) The PHP Group
```

```
Zend Engine v3.4.0, Copyright (c) Zend Technologies
```

```
with Zend OPcache v7.4.3, Copyright (c), by Zend Technologies
```

Step 4 — Creating a Virtual Host for your Website

- Create the directory for your_domain as follows:

```
sudo mkdir /var/www/your_domain
```

- Next, assign ownership of the directory with the \$USER environment variable, which will reference your current system user:

```
sudo chown -R $USER:$USER /var/www/your_domain
```

- Then, open a new configuration file in Apache's sites-available directory using your preferred command-line editor.
- Here, we'll use nano:

```
sudo nano /etc/apache2/sites-available/your_domain.conf
```

Step 4 — Creating a Virtual Host for your Website

- This will create a new blank file.
- Paste in the following bare-bones configuration:

`/etc/apache2/sites-available/your_domain.conf`

```
<VirtualHost *:80>
    ServerName your_domain
    ServerAlias www.your_domain
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/your_domain
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Step 4 — Creating a Virtual Host for your Website

- You can now use a2ensite to enable the new virtual host:
`sudo a2ensite your_domain`
- You might want to disable the default website that comes installed with Apache.
- This is required if you're not using a custom domain name, because in this case Apache's default configuration would overwrite your virtual host.
- To disable Apache's default website, type:
`sudo a2dissite 000-default`
- To make sure your configuration file doesn't contain syntax errors, run:
`sudo apache2ctl configtest`

Step 4 — Creating a Virtual Host for your Website

- Finally, reload Apache so these changes take effect:
`sudo systemctl reload apache2`
- Your new website is now active, but the web root
`/var/www/your_domain` is still empty.
- Create an `index.html` file in that location so that we can test that the virtual host works as expected:
`nano /var/www/your_domain/index.html`

Step 4 — Creating a Virtual Host for your Website

- Include the following content in this file:

```
/var/www/your_domain/index.html
<html>
  <head>
    <title>your_domain website</title>
  </head>
  <body>
    <h1>Hello World!</h1>

    <p>This is the landing page of <strong>your_domain</strong>.</p>
  </body>
</html>
```

Step 4 — Creating a Virtual Host for your Website

- Now go to your browser and access your server's domain name or IP address once again:

http://server_domain_or_IP

- You'll see a page like this:

Hello World!

This is the landing page of **your_domain**.

A Note About DirectoryIndex on Apache

- In case you want to change this behavior, you'll need to edit the /etc/apache2/mods-enabled/dir.conf file and modify the order in which the index.php file is listed within the DirectoryIndex directive:

```
sudo nano /etc/apache2/mods-enabled/dir.conf
```

```
/etc/apache2/mods-enabled/dir.conf
```

```
<IfModule mod_dir.c>
```

```
    DirectoryIndex index.php index.html index.cgi index.pl index.xhtml  
    index.htm
```

```
</IfModule>
```

- After saving and closing the file, you'll need to reload Apache so the changes take effect:

```
sudo systemctl reload apache2
```

Step 5 — Testing PHP Processing on your Web Server

- Create a new file named info.php inside your custom web root folder:
`nano /var/www/your_domain/info.php`
- This will open a blank file. Add the following text, which is valid PHP code, inside the file:

```
/var/www/your_domain/info.php
<?php
phpinfo();
```

- When you are finished, save and close the file.
- To test this script, go to your web browser and access your server's domain name or IP address, followed by the script name, which in this case is info.php:

`http://server_domain_or_IP/info.php`

Step 5 – Testing PHP Processing on your Web Server

- You'll see a page similar to this:

PHP Version 7.4.3	
System	Linux sassy-starfish 5.4.0-26-generic #30-Ubuntu SMP Mon Apr 20 16:58:30 UTC 2020 x86_64
Build Date	Mar 26 2020 20:24:23
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.4/apache2
Loaded Configuration File	/etc/php/7.4/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.4/apache2/conf.d
Additional .ini files parsed	/etc/php/7.4/apache2/conf.d/10-mysqlind.ini, /etc/php/7.4/apache2/conf.d/10-opcache.ini, /etc/php/7.4/apache2/conf.d/10-pdo.ini, /etc/php/7.4/apache2/conf.d/20-calendar.ini, /etc/php/7.4/apache2/conf.d/20-ctype.ini, /etc/php/7.4/apache2/conf.d/20-exif.ini, /etc/php/7.4/apache2/conf.d/20-fil.ini, /etc/php/7.4/apache2/conf.d/20-fileinfo.ini, /etc/php/7.4/apache2/conf.d/20-ftp.ini, /etc/php/7.4/apache2/conf.d/20-gettext.ini, /etc/php/7.4/apache2/conf.d/20-iconv.ini, /etc/php/7.4/apache2/conf.d/20-json.ini, /etc/php/7.4/apache2/conf.d/20-mysqli.ini, /etc/php/7.4/apache2/conf.d/20-pdo_mysqli.ini, /etc/php/7.4/apache2/conf.d/20-phar.ini, /etc/php/7.4/apache2/conf.d/20-posix.ini, /etc/php/7.4/apache2/conf.d/20-readline.ini, /etc/php/7.4/apache2/conf.d/20-shmop.ini, /etc/php/7.4/apache2/conf.d/20-sockets.ini, /etc/php/7.4/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.4/apache2/conf.d/20-sysvsem.ini, /etc/php/7.4/apache2/conf.d/20-sysvshm.ini, /etc/php/7.4/apache2/conf.d/20-tokenizer.ini
PHP API	20190902
PHP Extension	20190902
Zend Extension	320190902
Zend Extension Build	API20190902,NTS
PHP Extension Build	API20190902,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2, tlsv1.3
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, convert.iconv.*
This program makes use of the Zend Scripting Language Engine: Zend Engine v3.4.0, Copyright (c) Zend Technologies with Zend OPcache v7.4.3, Copyright (c), by Zend Technologies	
	

Step 5 – Testing PHP Processing on your Web Server

- After checking the relevant information about your PHP server through that page, it's best to remove the file you created as it contains sensitive information about your PHP environment and your Ubuntu server.
- You can use rm to do so:

```
sudo rm /var/www/your_domain/info.php
```

Step 6 — Testing Database Connection from PHP

- First, connect to the MySQL console using the root account:
`sudo mysql`
- To create a new database, run the following command from your MySQL console:
`CREATE DATABASE example_database;`
- Now you can create a new user and grant them full privileges on the custom database you've just created.

Step 6 — Testing Database Connection from PHP

- The following command creates a new user named example_user, using mysql_native_password as default authentication method.
- We're defining this user's password as password, but you should replace this value with a secure password of your own choosing.

```
CREATE USER 'example_user'@'%' IDENTIFIED WITH  
mysql_native_password BY 'password';
```

- Now we need to give this user permission over the example_database database:

```
GRANT ALL ON example_database.* TO 'example_user'@'%';
```

Step 6 — Testing Database Connection from PHP

- Now exit the MySQL shell with:

`exit`

- You can test if the new user has the proper permissions by logging in to the MySQL console again, this time using the custom user credentials:

`mysql -u example_user -p`

- Notice the `-p` flag in this command, which will prompt you for the password used when creating the `example_user` user.
- After logging in to the MySQL console, confirm that you have access to the `example_database` database:

`SHOW DATABASES;`

- Gives you the following output:

Output

```
+-----+  
| Database      |  
+-----+  
| example_database |  
| information_schema |  
+-----+  
2 rows in set (0.000 sec)
```

- Next, we'll create a test table named todo_list. From the MySQL console, run the following statement:

```
CREATE TABLE example_database.todo_list (  
    item_id INT AUTO_INCREMENT,  
    content VARCHAR(255),  
    PRIMARY KEY(item_id)  
)
```

Step 6 — Testing Database Connection from PHP

Step 6 — Testing Database Connection from PHP

- Insert a few rows of content in the test table.
- You might want to repeat the next command a few times, using different values:

```
INSERT INTO example_database.todo_list (content) VALUES ("My first important item");
```

- To confirm that the data was successfully saved to your table, run:

```
SELECT * FROM example_database.todo_list;
```

Step 6 — Testing Database Connection from PHP

- You'll see the following output:

Output

```
+-----+  
| item_id | content          |  
+-----+  
| 1 | My first important item |  
| 2 | My second important item |  
| 3 | My third important item |  
| 4 | and this one more thing |  
+-----+
```

4 rows in set (0.000 sec)

- After confirming that you have valid data in your test table, you can exit the MySQL console:

exit

Step 6 — Testing Database Connection from PHP

- Now you can create the PHP script that will connect to MySQL and query for your content.
- Create a new PHP file in your custom web root directory using your preferred editor. We'll use nano for that:

```
nano /var/www/your_domain/todo_list.php
```

Step 6 — Testing Database Connection from PHP

- Copy this content into your todo_list.php script:

/var/www/your_domain/todo_list.php

```
<?php
$user = "example_user";
$password = "password";
$database = "example_database";
$table = "todo_list";

try {
    $db = new PDO("mysql:host=localhost;dbname=$database", $user, $password);
    echo "<h2>TODO</h2><ol>";
    foreach($db->query("SELECT content FROM $table") as $row) {
        echo "<li>" . $row['content'] . "</li>";
    }
    echo "</ol>";
} catch (PDOException $e) {
    print "Error!: " . $e->getMessage() . "<br/>";
    die();
}
```

Step 6 — Testing Database Connection from PHP

- You can now access this page in your web browser by visiting the domain name or public IP address configured for your website, followed by /todo_list.php:

`http://your_domain_or_IP/todo_list.php`

- You should see a page like this, showing the content you've inserted in your test table:

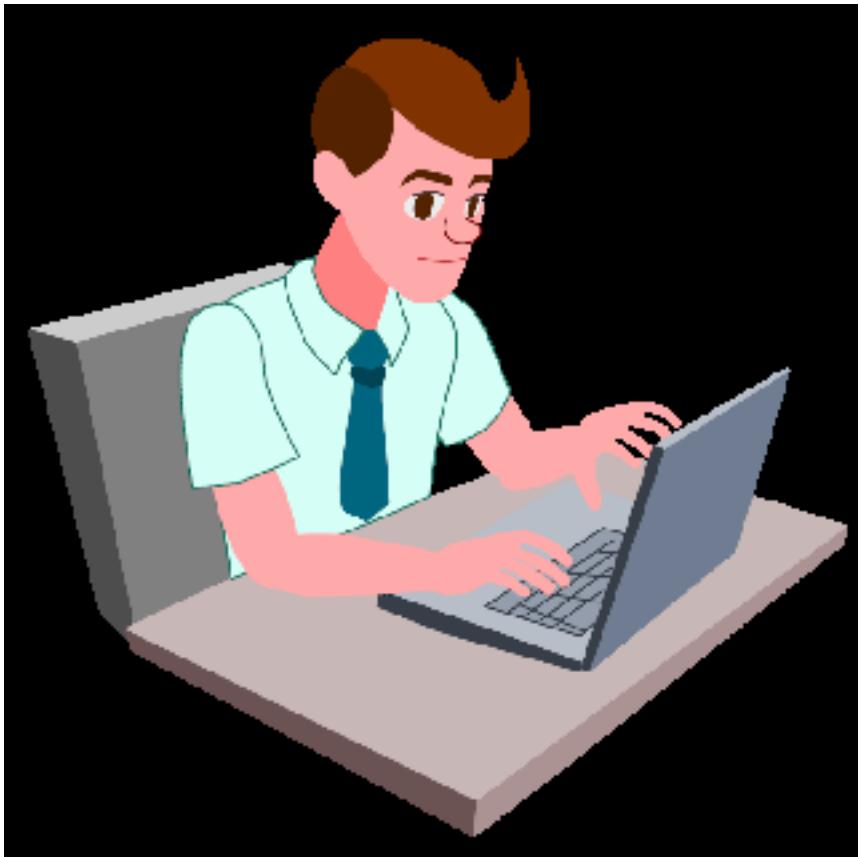
TODO

1. My first important item
2. My second important item
3. My third important item
4. and this one more thing

Conclusion

In this, we've built a flexible foundation for serving PHP websites and applications to your visitors, using Apache as web server and MySQL as database system.

- As an immediate next step, you should ensure that connections to your web server are secured, by serving them via HTTPS.
- In order to accomplish that, you can use Let's Encrypt to secure your site with a free TLS/SSL certificate.



"Complete Lab"

22. System Administration Overview



Introduction

- Samba is an open-source implementation of the SMB/CIFS networking protocol used in Windows environments for shared services such as file and printer access and Active Directory.
- Samba can also be used to create cross-platform file shares in a configuration called a standalone server.
- In this, you will install and configure a standalone Samba server to provide networked file stores or shares for a hypothetical small organization called Example.com.

Step 1 — Installing Samba

- Before installing new packages, let's update the local package index to include the most up-to-date versions from the Ubuntu repositories:

`sudo apt-get update`

- Next, install Samba:

`sudo apt-get install samba`

- This command will install and start both the Samba server smbd and the Samba NetBIOS server nmbd. nmbd is not required for this tutorial, so in the interests of security you can stop and disable it with systemctl:

`sudo systemctl stop nmbd.service`

`sudo systemctl disable nmbd.service`

Step 1 — Installing Samba

- The sudo systemctl disable nmbd.service command will produce the following output when run:

Output

```
nmbd.service is not a native service, redirecting to systemd-sysv-install
Executing /lib/systemd/systemd-sysv-install disable nmbd
insserv: warning: current start runlevel(s) (empty) of script `nmbd'
overrides LSB defaults (2 3 4 5).
insserv: warning: current stop runlevel(s) (0 1 2 3 4 5 6) of script `nmbd'
overrides LSB defaults (0 1 6).
```

Step 1 — Installing Samba

- To avoid security issues that can arise from running an unconfigured, network-enabled service
- let's stop the Samba server until configuration details are in place:

```
sudo systemctl stop smbd.service
```

Step 2 — Setting Samba's Global Options

- Rather than editing /etc/samba/smb.conf directly, rename it to smb.conf.original and create a new file with the name smb.conf:
- `sudo mv /etc/samba/smb.conf /etc/samba/smb.conf.orig`
- Before editing /etc/samba/smb.conf, let's check the available interfaces in order to tell Samba which it should recognize. Type the following:

`ip link`

Output

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    mode DEFAULT group default qlen 1
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    state UP mode DEFAULT group default qlen 1000
        link/ether 02:21:2c:03:ef:e2 brd ff:ff:ff:ff:ff:ff
```

- Let's begin editing this file with nano or your favorite editor:

`sudo nano /etc/samba/smb.conf`

- The [global] section of this file will define the server's name, role, and other details, including network interfaces:

`/etc/samba/smb.conf`

`[global]`

`server string = samba_server`

`server role = standalone server`

`interfaces = lo your_network_interface`

`bind interfaces only = yes`

`disable netbios = yes`

`smb ports = 445`

`log file = /var/log/samba/smb.log`

`max log size = 10000`

Step 2 — Setting Samba's Global Options

Step 2 — Setting Samba's Global Options

- If you want more detailed logging while you are setting up the server, append the following line to the [global] section:

/etc/samba/smb.conf

log level = 3 passdb:5 auth:5

- This sets the log level to 3 (info), increasing the granularity of log information from the default setting of 1.
- The higher setting of 5 for the passdb and auth debug classes provides more information related to user authentication.

Step 2 — Setting Samba's Global Options

- Running the testparm command on the smb.conf file produces the following output:

Output

Load smb config files from /etc/samba/smb.conf

Loaded services file OK.

Server role: ROLE_STANDALONE

Press enter to see a dump of your service definitions

- Pressing ENTER produces the following output:

Output

```
# Global parameters
[global]
    server string = samba_server
    interfaces = lo your_network_interface
    bind interfaces only = Yes
    server role = standalone server
    log file = /var/log/samba/smb.log
    max log size = 10000
    smb ports = 445
    disable netbios = Yes
    idmap config * : backend = tdb
```

Step 2 — Setting Samba's Global Options

Step 3 — Creating Users

- Execute the following commands to create the /samba/ directory and set the group ownership to sambashare:

```
sudo mkdir /samba/
```

```
sudo chown :sambashare /samba/
```

- Next, create david's home directory under the /samba/ directory:

```
sudo mkdir /samba/david
```

- Now, add david as a system user with the following command:

```
sudo adduser --home /samba/david --no-create-home --shell  
/usr/sbin/nologin --ingroup sambashare david
```

Step 3 — Creating Users

- Now that the system user david exists, you can set the ownership and permissions on his Samba home directory:

```
sudo chown david:sambashare /samba/david/  
sudo chmod 2770 /samba/david/
```

Step 3 — Creating Users

- Next, add david to the Samba server. Samba keeps its own database of users and passwords, which it uses to authenticate logins.
- In order to log in, all users must be added to the Samba server and enabled.
- Execute the following smbpasswd commands to accomplish both of these tasks:

```
sudo smbpasswd -a david  
sudo smbpasswd -e david
```

Step 3 — Creating Users

- To create the admin user, run through the following commands, changing the home directory to /samba/everyone/:

```
sudo mkdir /samba/everyone
sudo adduser --home /samba/everyone --no-create-home --shell
/usr/sbin/nologin --ingroup sambashare admin
sudo chown admin:sambashare /samba/everyone/
sudo chmod 2770 /samba/everyone/
sudo smbpasswd -a admin
sudo smbpasswd -e admin
```

Step 3 — Creating Users

- Execute the following commands to create a new group called admins and add the user admin to this group:

```
sudo groupadd admins
```

```
sudo usermod -G admins admin
```

Step 4 — Configuring the Samba Shares

- Use the nano text editor again to open and edit this file:
`sudo nano /etc/samba/smb.conf`
- The following configuration block will define each user's personal share:
`/etc/samba/smb.conf`

...

[share_name]

path =

browsable =

read only =

force create mode =

force directory mode =

valid users =

Step 4 — Configuring the Samba Shares

- Add the following share configuration block for david, defining his home directory, the permissions for this directory's group ownership, and the users that should have access to his share:

/etc/samba/smb.conf

[david]

path = /samba/david

browsable = no

read only = no

force create mode = 0660

force directory mode = 2770

valid users = david @admins

Step 4 — Configuring the Samba Shares

- The [everyone] share will differ from the others in both [name], path, valid users, and browsable options, and will look like this:

`/etc/samba/smb.conf`

...

`[everyone]`

`path = /samba/everyone`

`browsable = yes`

`read only = no`

`force create mode = 0660`

`force directory mode = 2770`

`valid users = @sambashare @admins`

Step 4 – Configuring the Samba Shares

- The complete smb.conf file will look like this:

/etc/samba/smb.conf

```
[global]
server string = samba_server
server role = standalone server
interfaces = lo your_network_interface
bind interfaces only = yes
disable netbios = yes
smb ports = 445
log file = /var/log/samba/smb.log
max log size = 10000

[david]
path = /samba/david
browseable = no
read only = no
force create mode = 0660
force directory mode = 2770
valid users = david @admins

[mike]
path = /samba/mike
browseable = no
read only = no
force create mode = 0660
force directory mode = 2770
valid users = mike @admins

[jane]
path = /samba/jane
browseable = no
read only = no
force create mode = 0660
force directory mode = 2770
valid users = jane @admins

[lucy]
path = /samba/lucy
browseable = no
read only = no
force create mode = 0660
force directory mode = 2770
valid users = lucy @admins

[everyone]
path = /samba/everyone
browseable = yes
read only = no
force create mode = 0660
force directory mode = 2770
valid users = @sambashare @admins
```

Step 4 — Configuring the Samba Shares

Test the configuration again:

```
$ testparm
```

This will produce output that looks like the following:

Output

```
Load smb config files from /etc/samba/smb.conf
rlimit_max: increasing rlimit_max (1024) to minimum Windows limit (16384)
Processing section "[david]"
Processing section "[jane]"
Processing section "[mike]"
Processing section "[lucy]"
Processing section "[everyone]"
Loaded services file OK.
Server role: ROLE_STANDALONE
```

Press enter to see a dump of your service definitions

With the configuration check complete, let's start the Samba server with `systemctl`:

```
$ sudo systemctl start smbd.service
```

Step 5 — Logging Into the Samba Server

Linux — The Command Line

- You can use a tool called smbclient to access Samba from the command line.
- This package is not included by default on most Linux distributions, so you will need to install it with your local package manager.
- On Debian and Ubuntu servers install smbclient with the following command:

```
sudo apt-get update
```

```
sudo apt-get install smbclient
```

- On Fedora systems, use the following:

```
sudo dnf update
```

```
sudo samba-client
```

Step 5 — Logging Into the Samba Server

- And on CentOS:

```
sudo yum update
```

```
sudo yum install samba-client
```

- smbclient uses the following format to access Samba shares:

```
smbclient //your_samba_hostname_or_server_ip/share -U username
```

Step 5 — Logging Into the Samba Server

- You can use either your server's IP or the hostname you defined in /etc/samba/smb.conf to access the share.
- This example uses the hostname samba.example.com to access david's share on the Samba server you created in the previous steps:

`smbclient //samba.example.com/david -U david`

- If david wants access to the common share (everyone), change the command to:

`smbclient //samba.example.com/everyone -U david`

- After running the smbclient command, you will be prompted for the Samba password and logged into a command line interface reminiscent of the FTP text interface:

`smb: \>`

Step 5 — Logging Into the Samba Server

- For example, you can create a directory and list its contents as follows:

```
mkdir test
```

```
ls
```

You should see the following output:

Output

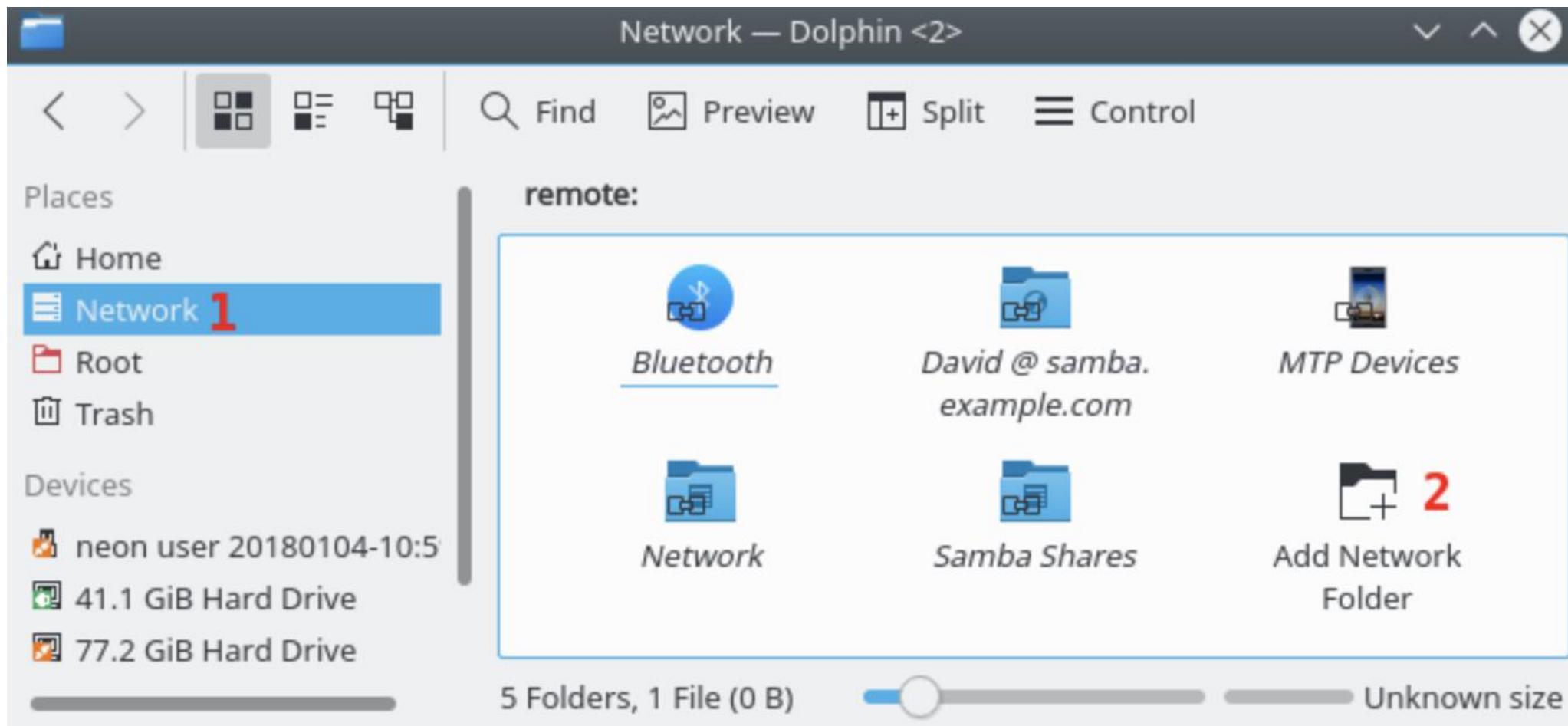
```
.          D  0 Fri Feb  2 14:49:01 2018
..
test       D  0 Fri Feb  2 14:49:01 2018
```

- Remove the directory by typing:

```
rmdir test
```

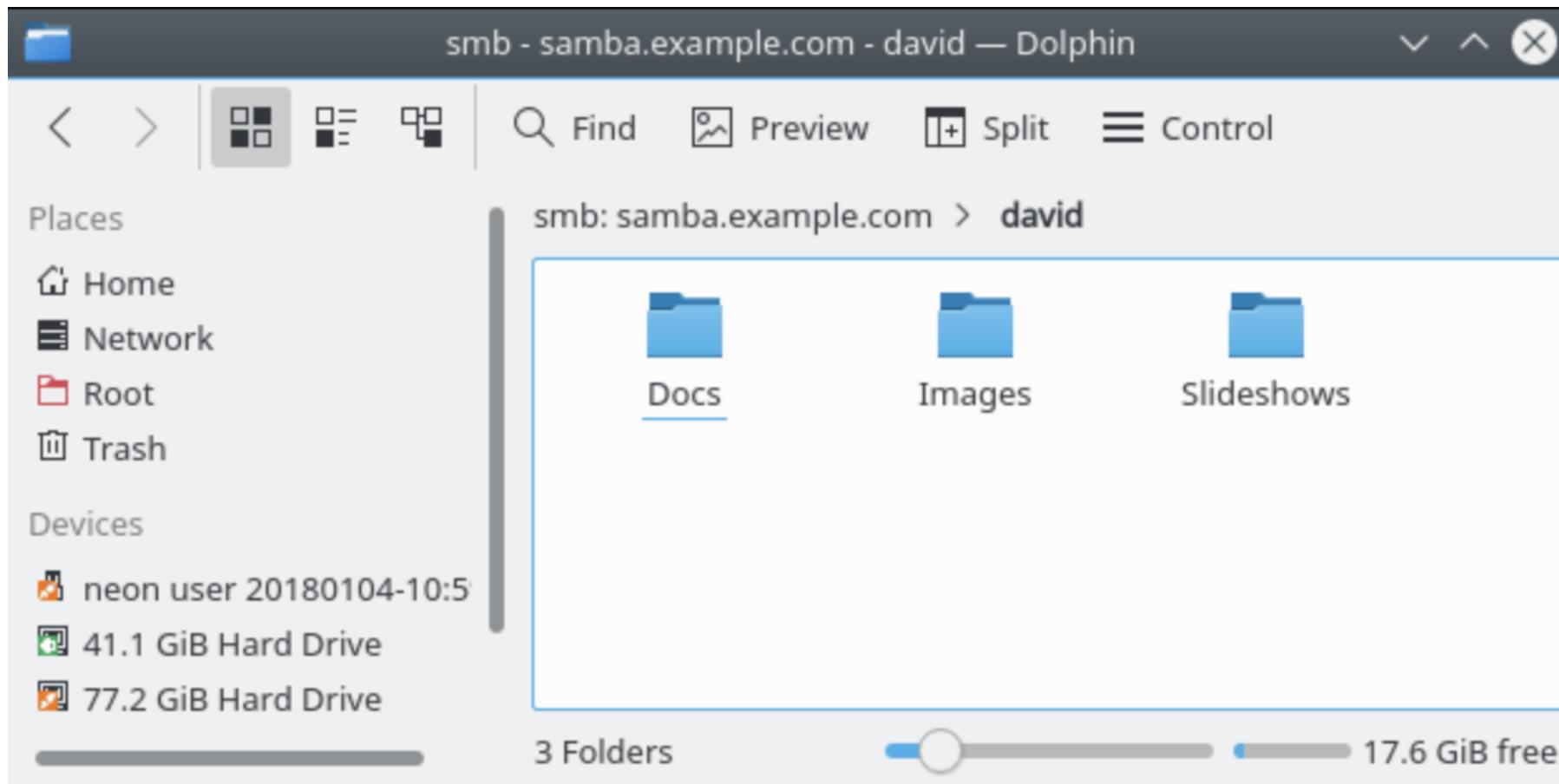
Step 5 — Logging Into the Samba Server

Linux — KDE with Dolphin



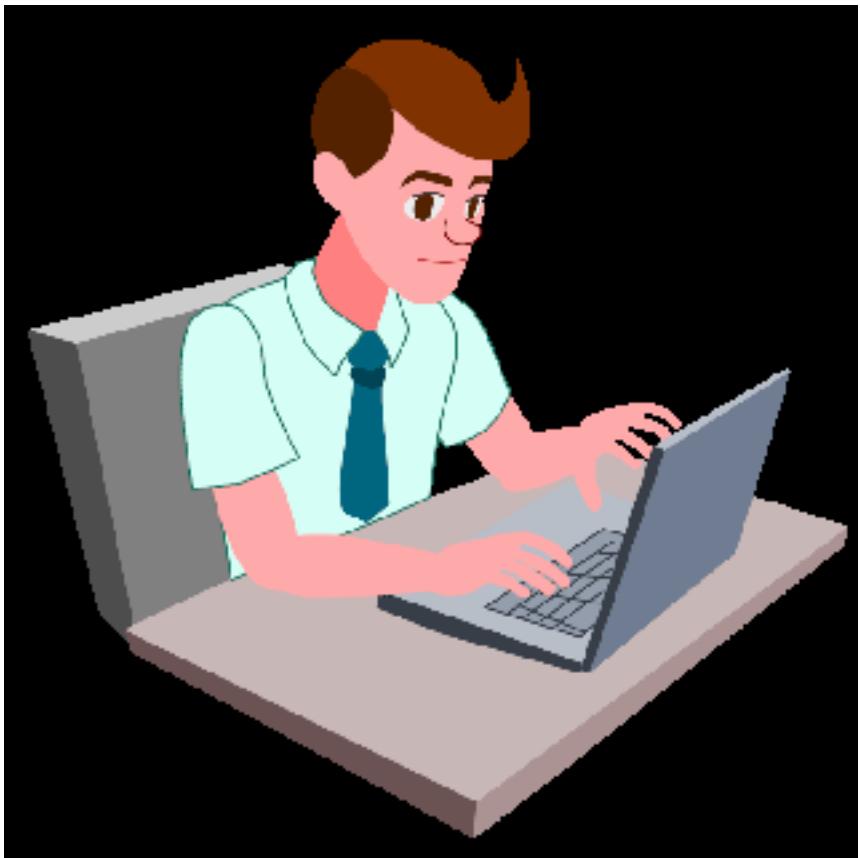
Step 5 – Logging Into the Samba Server

- Dolphin will now connect and open the Samba share which will look like this:



Conclusion

- In this, you have created cross-platform online file shares using the Samba server.
- You have also accessed these shares from Windows, Linux, and macOS.
- Samba shares have become so common that many applications are able to access the data stored in them.
- These applications can extend the functionality and usefulness of your Samba shares.



"Complete Lab"

23. Networked File Systems (NFS)

Introduction

- NFS, or Network File System, is a distributed file system protocol that allows you to mount remote directories on your server.
- This lets you manage storage space in a different location and write to that space from multiple clients.
- NFS provides a relatively standard and performant way to access remote systems over a network and works well in situations where the shared resources must be accessed regularly.

Step 1 — Downloading and Installing the Components

On the Host

- On the host server, install the nfs-kernel-server package, which will allow you to share your directories.
- Since this is the first operation that you’re performing with apt in this session, refresh your local package index before the installation:

`sudo apt update`

`sudo apt install nfs-kernel-server`

- Once these packages are installed, switch to the client server.

Step 1 — Downloading and Installing the Components

On the Client

- On the client server, we need to install a package called nfs-common, which provides NFS functionality without including any server components.
- Again, refresh the local package index prior to installation to ensure that you have up-to-date information:

```
sudo apt update
```

```
sudo apt install nfs-common
```

Step 2 — Creating the Share Directories on the Host

- First, make the share directory:

```
sudo mkdir /var/nfs/general -p
```

- Since we're creating it with sudo, the directory is owned by the host's root user:

```
ls -la /var/nfs/general
```

Output

```
drwxr-xr-x 2 root root 4096 May 14 18:36 .
```

- NFS will translate any root operations on the client to the nobody:nogroup credentials as a security measure.
- Therefore, we need to change the directory ownership to match those credentials.

```
sudo chown nobody:nogroup /var/nfs/general
```

Step 3 — Configuring the NFS Exports on the Host Server

- On the host machine, open the `/etc/exports` file in your text editor with root privileges:

```
sudo nano /etc/exports
```

- The file has comments showing the general structure of each configuration line.
- The syntax is as follows:

```
/etc/exports
```

```
directory_to_share client/share_option1,...,share_optionN)
```

Step 3 — Configuring the NFS Exports on the Host Server

- When you are finished making your changes, save and close the file.
- Then, to make the shares available to the clients that you configured, restart the NFS server with the following command:

`sudo systemctl restart nfs-kernel-server`

- Before you can actually use the new shares, however, you'll need to be sure that traffic to the shares is permitted by firewall rules.

Step 4 — Adjusting the Firewall on the Host

- First, let's check the firewall status to see if it's enabled and, if so, to see what's currently permitted:

```
sudo ufw status
```

Output

Status: active

To	Action	From
--	-----	-----
OpenSSH	ALLOW	Anywhere
OpenSSH (v6)	ALLOW	Anywhere (v6)

Step 4 — Adjusting the Firewall on the Host

- Use the following command to open port 2049 on the host, being sure to substitute your client IP address:

`sudo ufw allow from client_ip to any port nfs`

- You can verify the change by typing:

`sudo ufw status`

- You should see traffic allowed from port 2049 in the output:

`Output`

`Status: active`

To	Action	From
--	-----	-----
OpenSSH	ALLOW	Anywhere
2049	ALLOW	203.0.113.24
OpenSSH (v6)	ALLOW	Anywhere (v6)

Step 5 — Creating Mount Points and Mounting Directories on the Client

- We'll create two directories for our mounts:

```
sudo mkdir -p /nfs/general
```

```
sudo mkdir -p /nfs/home
```

- Now that we have a location to put the remote shares and we've opened the firewall, we can mount the shares using the IP address of our host server:

```
sudo mount host_ip:/var/nfs/general /nfs/general
```

```
sudo mount host_ip:/home /nfs/home
```

Step 5 – Creating Mount Points and Mounting Directories on the Client

These commands will mount the shares from the host computer onto the **client** machine. You can double-check that they mounted successfully in several ways. You can check this with a `mount` or `findmnt` command, but `df -h` provides a more readable output:

```
client:$ df -h
```

Output

Filesystem	Size	Used	Avail	Use%	Mounted on
udev	474M	0	474M	0%	/dev
tmpfs	99M	936K	98M	1%	/run
/dev/vda1	25G	1.8G	23G	8%	/
tmpfs	491M	0	491M	0%	/dev/shm
tmpfs	5.0M	0	5.0M	0%	/run/lock
tmpfs	491M	0	491M	0%	/sys/fs/cgroup
/dev/vda15	105M	3.9M	101M	4%	/boot/efi
tmpfs	99M	0	99M	0%	/run/user/1000
10.132.212.247:/var/nfs/general	25G	1.8G	23G	8%	/nfs/general
10.132.212.247:/home	25G	1.8G	23G	8%	/nfs/home

Step 5 — Creating Mount Points and Mounting Directories on the Client

- For example:

```
du -sh /nfs/home  
Output  
36K      /nfs/home
```

Step 6 — Testing NFS Access

Example 1: The General Purpose Share

First, write a test file to the `/var/nfs/general` share:

```
client:$ sudo touch /nfs/general/general.test
```

Then, check its ownership:

```
client:$ ls -l /nfs/general/general.test
```

Output

```
-rw-r--r-- 1 nobody nogroup 0 Aug 1 13:31 /nfs/general/general.test
```

Step 6 — Testing NFS Access

Example 2: The Home Directory Share

To compare the permissions of the General Purpose share with the Home Directory share, create a file in `/nfs/home` the same way:

```
client:$ sudo touch /nfs/home/home.test
```

Then look at the ownership of the file:

```
client:$ ls -l /nfs/home/home.test
```

Output

```
-rw-r--r-- 1 root root 0 Aug 1 13:32 /nfs/home/home.test
```

Step 7 — Mounting the Remote NFS Directories at Boot

- We can mount the remote NFS shares automatically at boot by adding them to /etc/fstab file on the client.
- Open this file with root privileges in your text editor:

`sudo nano /etc/fstab`

- At the bottom of the file, add a line for each of our shares. They will look like this:

`/etc/fstab`

`...`

```
host_ip:/var/nfs/general  /nfs/general  nfs
auto,nofail,noatime,nolock,intr,tcp,actimeo=1800 0 0
host_ip:/home          /nfs/home    nfs
auto,nofail,noatime,nolock,intr,tcp,actimeo=1800 0 0
```

Step 8 – Unmounting an NFS Remote Share

- If you no longer want the remote directory to be mounted on your system, you can unmount it by moving out of the share's directory structure and unmounting, like this:

```
cd ~  
sudo umount /nfs/home  
sudo umount /nfs/general
```

Step 8 — Unmounting an NFS Remote Share

- Will remove the remote shares, leaving only your local storage accessible:

`df -h`

Output

Filesystem

udev

tmpfs

/dev/vda1

tmpfs

tmpfs

tmpfs

/dev/vda15

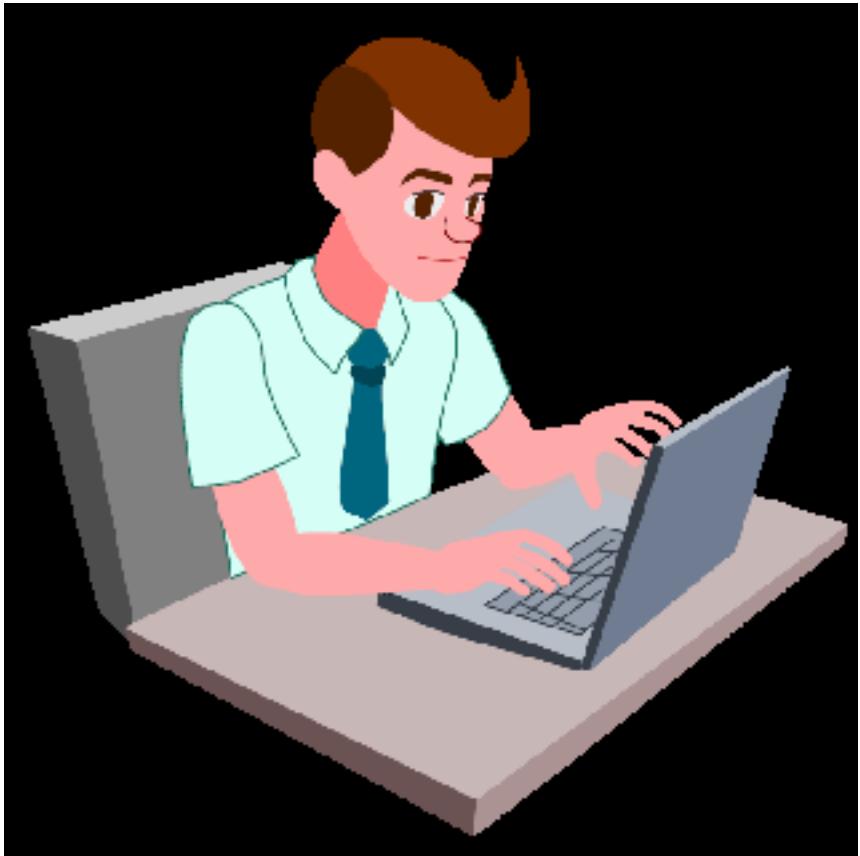
tmpfs

	Size	Used	Avail	Use%	Mounted on
udev	474M	0	474M	0%	/dev
tmpfs	99M	936K	98M	1%	/run
/dev/vda1	25G	1.8G	23G	8%	/
tmpfs	491M	0	491M	0%	/dev/shm
tmpfs	5.0M	0	5.0M	0%	/run/lock
tmpfs	491M	0	491M	0%	/sys/fs/cgroup
/dev/vda15	105M	3.9M	101M	4%	/boot/efi
tmpfs	99M	0	99M	0%	/run/user/1000

Conclusion

In this, we created an NFS host and illustrated some key NFS behaviours by creating two different NFS mounts, which we shared with a NFS client.

- If you're looking to implement NFS in production, it's important to note that the protocol itself is not encrypted.
- In cases where you're sharing over a private network, this may not be a problem.



"Complete Lab"