

Lab #1 - Linux Commands

You may wonder what Linux is and why should we learn about it. Yes, we understand your concerns and are ready to help! Just follow this introductory tutorial, you will have a better understanding of the following questions:

1. What is Linux?
2. Why learn it?
3. How to use it?

What are Linux and Linux Commands?

Linux is an operating system like Windows and Mac OS, and it is a special kind of program which controls the computer's processor, hard drive, and network connection, but its most important job is to run different programs. All the Linux/Unix commands are run in **Terminal (Mac)/PowerShell (Windows)**. Commands themselves are **case-sensitive**. The terminal can be used to accomplish all administrative tasks including package installation, file manipulation, and user management.

Why learn it?

As a human, how can we interact with the operating system and tell the computer to do things we want it to do? The most common way is using a graphical file explorer which translates clicks and double-clicks into commands to open files and run programs.

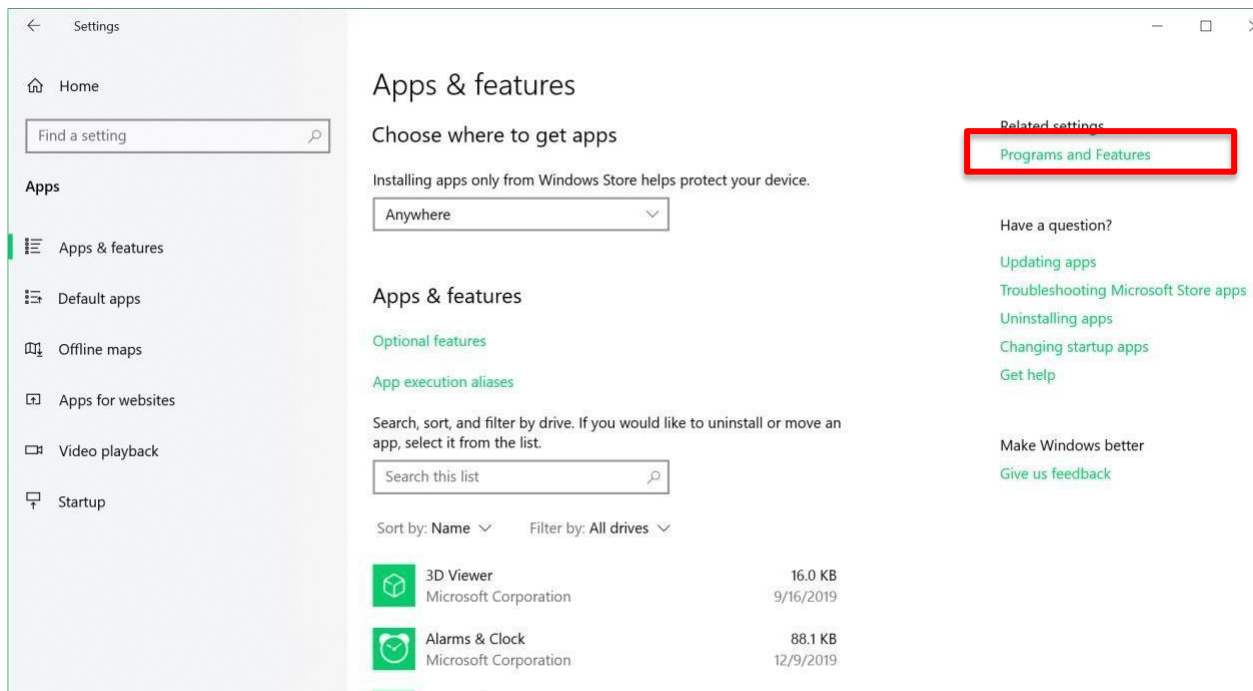
As future programmers, it is a good idea for us to learn and to understand how to interact with our computers, here are some reasons why:

- There are commands that are not doable using GUI (Graphical User Interfaces).
- GUIs are not as stable as commands. Sometimes after installing or updating a GUI application, it might have trouble starting and leaves you without any error messages. In this case, it is better to start the application from the command line which shows you what is wrong.
- Linux provides a safer environment for users, very few viruses are for Linux and they are not of that high quality.

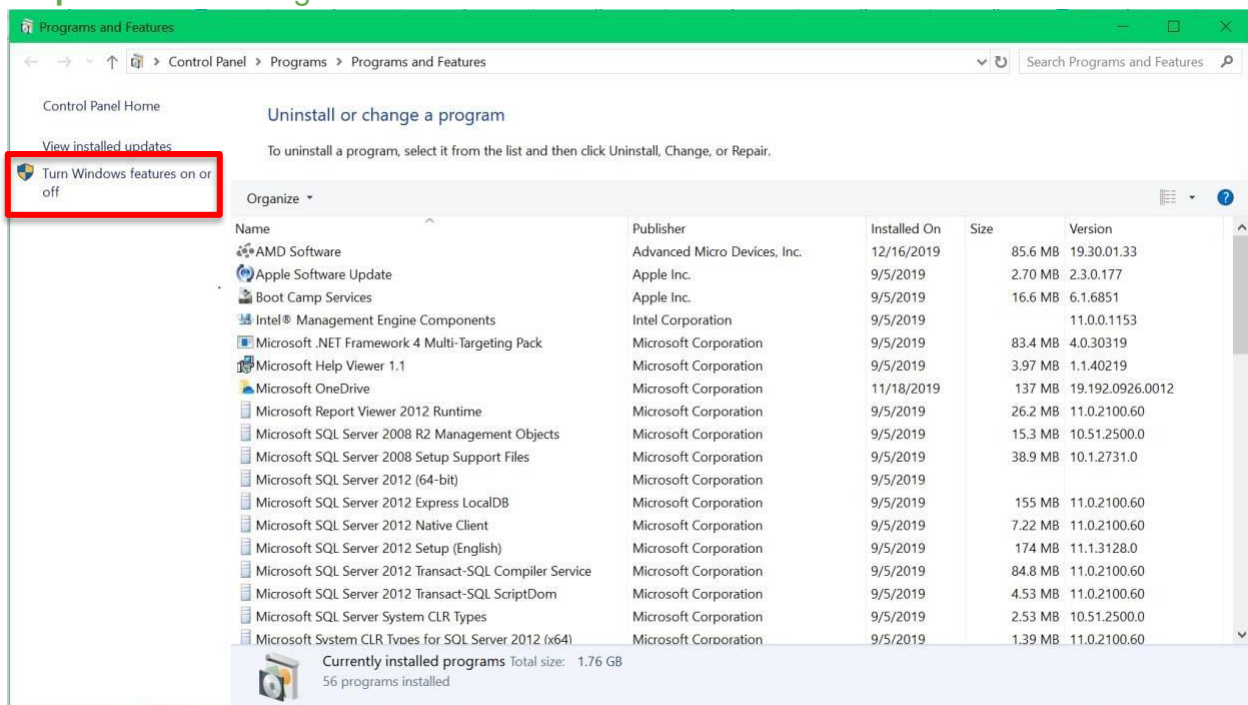
How to use it?

For Windows User ONLY (Mac users skip this part): To run Linux commands on Windows, you will need to enable the Windows Linux subsystem.

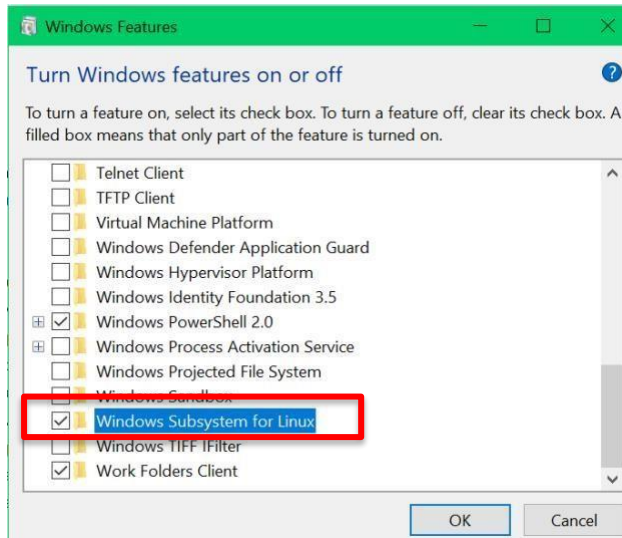
Step 1: Open Setting and go to App & features.



Step 2: Click on Programs and Features.



Step 3: Click on Turn Windows features on or off.



Find and check Windows Subsystem for Linux. Press OK and follow the pop-up window to restart your computer.

Topic 1: Navigate and check directories

- **pwd** (print working directory) - prints out the absolute path of your current working directory.
- **ls** (listing) - lists the contents of the current directory.
- **cd** <...> (changes directory)
 - **cd /path/to/filename** - changes the directory to any directory on the system by typing its full path after the **cd** command.
 - **cd <directory>** - changes the directory to <directory>
 - **cd** - changes to the home directory.
 - **cd ..** - moves to the parent directory of the current directory. **Demo**

1:

1. Open Terminal on Mac/PowerShell on Windows. **ATTENTION:** For Windows users **ONLY**, after opening PowerShell, run **bash** command to enable bash. Once bash is enabled, everything will be the same for Mac and Windows. If you are using a **Mac**, the initial Terminal should look like this:

```
mandili — -bash — 80x24
Last login: Tue Jan 14 12:56:21 on ttys000
(base) Mandis-MacBook-Pro:~ mandili$
```

If you are using a **Windows**, the initial PowerShell should look like this:

2. The `pwd` command will print the path of your current working directory. (Your output should be slightly different than mine as `/Users/yourUserName`. **Mac:**

```
(base) Mandis-MacBook-Pro:~ mandili$ pwd
/Users/mandili
```

Windows:

```
mandili@DESKTOP-55BNPEK: /mnt/c/Users/Mandi Li $ pwd
/mnt/c/Users/Mandi Li
```

3. Now, let's run `ls` command. All contents (files) in my current directory are printed to the screen. Again, your results will be different than mine.

```
(base) Mandis-MacBook-Pro:~ mandili$ ls
Applications      Music              jelloWorld
Desktop           Pictures          ngtutorial
Documents         Public            node_modules
Downloads         VirtualBox VMs    opt
Library           anaconda3         package-lock.json
Movies            eclipse-workspace test1
```

4. What if I want to go to `Desktop` and work on files in that directory? Use `cd` command followed by the name of the content: `cd Desktop`.

ATTENTION: Linux commands are case-sensitive, so make sure the name is exactly the same as displayed.

```
(base) Mandis-MacBook-Pro:~ mandili$ cd Desktop
```

Use `pwd` command to check if you are in the `Desktop` directory. **Mac:**

```
(base) Mandis-MacBook-Pro:Desktop mandili$ pwd
/Users/mandili/Desktop
```

Windows:

```
mandili@DESKTOP-55BNPEK: /mnt/c/Users/Mandi Li $ cd Desktop
mandili@DESKTOP-55BNPEK: /mnt/c/Users/Mandi Li/Desktop $ pwd
/mnt/c/Users/Mandi Li/Desktop
```

Run `ls` command to list all contents there. All names of files on your desktop should be printed here.

```
(base) Mandis-MacBook-Pro:Desktop mandili$ ls
Academic Transcript.pdf
Case_Competition
Cover Letter-ML-AG.pdf
Cover Letter-ML.docx
Cover Letter-ML.pdf
Data Mining Project
Data Mining for Business
```

Topic 2: Create folders and files

- **touch** - creates/modifies new files or directories.
- **mkdir** (make directory) - creates a new directory.
- **rmdir** (remove directory) - deletes a directory.
- **head** - prints top n number of data of the given input.
- **tail** - prints bottom n number of data of the given input.
- **cat** - reads data from a file and gives their contents as output. **Demo 2:**

1. In this Demo, we will practice creating and deleting directories/files. First, choose a directory you like on your laptop, and create a folder **tmp** using **mkdir tmp** command. (I recommend creating the folder on your home screen - Desktop)
Check your desktop screen, a folder named **tmp** should be created.
2. Let's change the directory into our newly created **tmp** folder using **cd tmp** command.
3. Run **ls** command to see what is in **tmp**. Obviously, it prints nothing, since **tmp** is empty.
4. Let's create three additional folders in **tmp** using **mkdir dir1 dir2 dir3**. Use **ls** to check if they are created successfully.

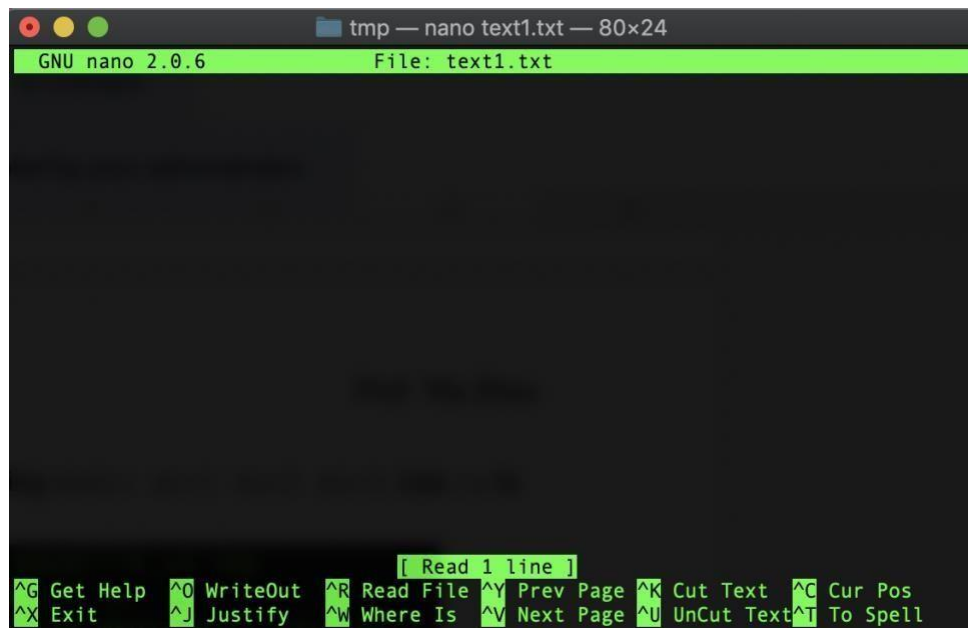
```
(base) Mandis-MacBook-Pro:Desktop mandili$ cd tmp
(base) Mandis-MacBook-Pro:tmp mandili$ ls
(base) Mandis-MacBook-Pro:tmp mandili$ mkdir dir1 dir2 dir3
(base) Mandis-MacBook-Pro:tmp mandili$ ls
dir1  dir2  dir3
```

5. Create an empty file named **text1.txt** in **tmp** using **touch text1.txt**.
6. Run **nano text1.txt** to open the editor to write something to the empty file.

Note: *nano* is one of the text editors we can use to edit files in Terminal. Alternatively, you can

use **vi/vim**. To use **vi/vim**, run **vi <fileName>**, and press the **"I"** key to insert.

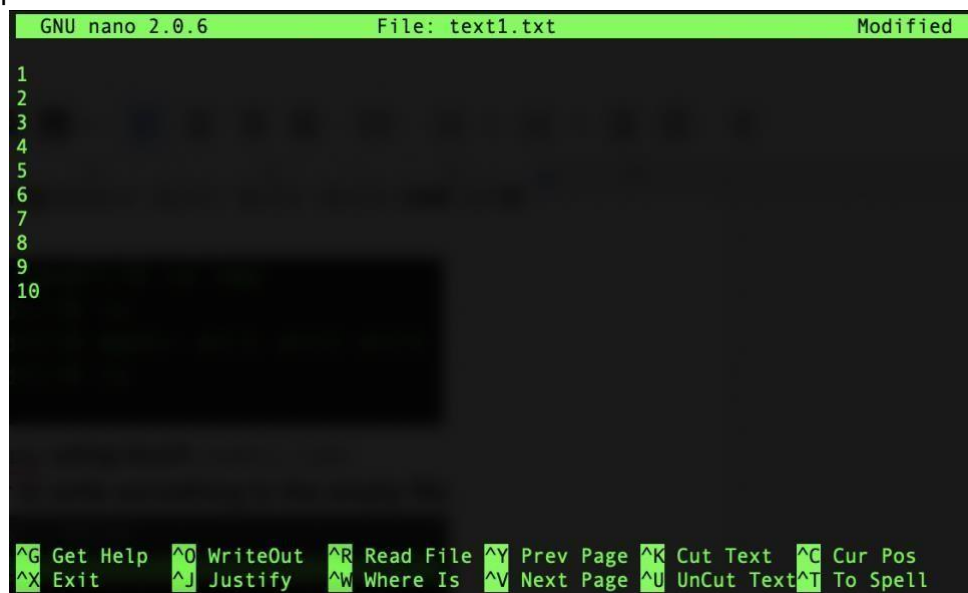
To exit the editor, type **":q!"**.



```
tmp — nano text1.txt — 80x24
GNU nano 2.0.6      File: text1.txt

[ Read 1 line ]
^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```

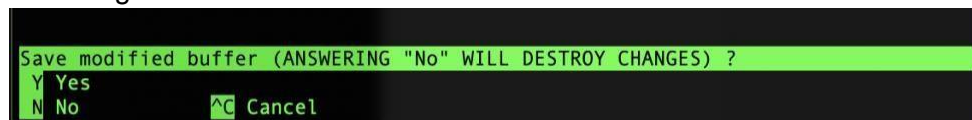
Type in 1 to 10 to the editor.



```
GNU nano 2.0.6      File: text1.txt      Modified
1
2
3
4
5
6
7
8
9
10

^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```

Press the “Control” key + X to exit. The editor will ask you whether to save or discard your changes.



```
Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ?
Y Yes
N No      ^C Cancel
```

Press the ‘Y’ key to save the changes for this time and then press the “Return” key to exit the editor.

7. If we have done everything correctly, the `text1.txt` file should contain the data we have typed in. Use `cat text1.txt` to check the content.

```
(base) Mandis-MacBook-Pro:tmp mandili$ cat text1.txt
1
2
3
4
5
6
7
8
9
10
```

Cool!

8. What if we only want to check the first 3 lines of data. Use `head -n 3 text1.txt`.

```
(base) Mandis-MacBook-Pro:tmp mandili$ head -n 3 text1.txt
1
2
3
```

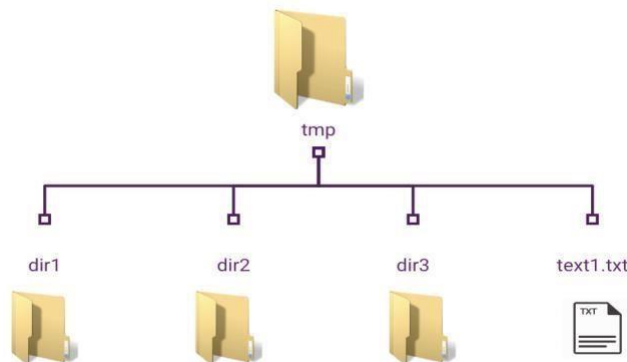
What if we want to see the end of the content? It is similar to what we did using `head` command, just replace `head` with `tail` and replace the integer after `-n` to the lines of data you would like to see (do not exceed the total number of lines in the document). Give a try and observe what is returned.

Topic 3: Move and manipulate files

- `mv` (move) - move one or more files/directories from one place to another.
- `cp` (copy) - copies the contents of one file to another file.
- `rm` `<...>` (remove file/directory)
 - `rm filename` - deletes the file.
 - `rm -r` - deletes the entire directory. Demo

3:

1. Here is the current structure of our `tmp` folder.



What if we require to move `text1.txt` file to `dir1`? We can unquestionably do that by running `mv text1.txt dir1`.

```
(base) Mandis-MacBook-Pro:tmp mandili$ mv text1.txt dir1
(base) Mandis-MacBook-Pro:tmp mandili$ ls
dir1  dir2  dir3
(base) Mandis-MacBook-Pro:tmp mandili$ cd dir1
(base) Mandis-MacBook-Pro:dir1 mandili$ ls
text1.txt
```

Again, use combinations of `ls` and `cd` commands to check if we did the right thing.

2. Run `cp text1.txt text2.txt` to create a copy of the `text1.txt` file and name it `text2.txt`.

```
(base) Mandis-MacBook-Pro:dir1 mandili$ cp text1.txt text2.txt
(base) Mandis-MacBook-Pro:dir1 mandili$ ls
text1.txt  text2.txt
(base) Mandis-MacBook-Pro:dir1 mandili$ cat text2.txt
1
2
3
4
5
6
7
8
9
10
```

3. In this step, I will demonstrate how to use the `mv` command. First, create an empty file by yourself and remember to give it a cool name. Then, run `rm <filename>` to practice deleting it.

Topic 4: Additional commands

- `clear` - clears the terminal screen.
- `sudo` - runs commands with elevated privileges.
- `man <command>` (manual) - displays the manual page where you can learn more about the command in detail.
- `cmp` (compare) - compares two files byte by byte and helps to find out whether two files are identical.

Demo 4:

1. Run `clear` to see what happens.
2. We use `sudo` to gain privileges when downloading/updating some packages/software or running some programs. Here is a circumstance where we may use `sudo`: You are working on installing new software on your computer. You might receive an error message informing you that the user does not have proper permissions to execute the command. It is because, by default, a standard user is not allowed to install software on the Linux machine. To successfully install the software, you should run `sudo <command>` to gain superuser privilege.

3. If you have any concerns or simply want to know more about the usages of each command. Run `man <commandName>` to read more.
4. The `cmp` is used to compare two files. Run `cmp text1.txt text2.txt` to see. Nothing is shown, why?

```
(base) Mandis-MacBook-Pro:dir1 mandili$ cmp text1.txt text2.txt
(base) Mandis-MacBook-Pro:dir1 mandili$
```

Let's do something to figure out what is happening.

```
(base) Mandis-MacBook-Pro:dir1 mandili$ nano text1.txt
(base) Mandis-MacBook-Pro:dir1 mandili$ cmp text1.txt text2.txt
text1.txt text2.txt differ: char 1, line 1
(base) Mandis-MacBook-Pro:dir1 mandili$ cat text1.txt

2
3
4
5
6
7
8
9
10
(base) Mandis-MacBook-Pro:dir1 mandili$ cat text2.txt
1
2
3
4
5
6
7
8
9
10
```

Here, I used the `nano` editor to delete the number from line 1 in the `text1.txt` file. Then, I ran `cmp text1.txt text2.txt` again. Now, you can see the result indicating the difference between `text1.txt` and `text2.txt`.

5. You can go ahead and play around with the commands you just learned. After you have done with playing around with our directory `tmp`, we can delete it from our desktop using `rm -r tmp` command. Check your desktop again, the folder `tmp` should be gone!

Note: To delete a file, we can simply run `rm filename`. The shell will prompt an error message if you try to delete a directory without the `-r` command option. The added `-r` notation is designed to stop you from accidentally deleting an entire directory full of work. Alternatively, `rmdir directoryName` can do the same.

Topic 5: Monitor and manage processes (Optional)

- `top` - shows all dynamic real-time views of the running system.

- **ps** (process status) - lists a snapshot of the currently running processes and their PIDs along with some other information depending on different options.
 - **kill** - terminates the process manually. **Demo 5: (Optional)**
5. Let's run the **top** command to see what will show on the screen. **Mac:**

```
Processes: 388 total, 2 running, 386 sleeping, 2006 threads      13:07:37
Load Avg: 1.89, 2.04, 2.21  CPU usage: 2.68% user, 3.43% sys, 93.87% idle
SharedLibs: 219M resident, 56M data, 31M linkedit.
MemRegions: 221834 total, 4183M resident, 147M private, 1943M shared.
PhysMem: 14G used (3149M wired), 2506M unused.
VM: 2202G vsize, 1374M framework vsize, 50755271(0) swapins, 56276547(0) swapouts
Networks: packets: 35255527/35G in, 22787488/5871M out.
Disks: 70239175/899G read, 33283797/584G written.

PID   COMMAND      %CPU   TIME    #TH   #WQ   #PORTS  MEM    PURG    CMPS    PGRP
39548  screencaptur 0.5    00:00.41 2     1     54      3120K  620K   0B     300
39547  helpd        0.0    00:00.02 2     1     42      1056K  0B     0B     39547
39546  top          14.1   00:02.19 1/1    0     27      8416K+ 0B     0B     39546
39539  mdworker_sha 0.0    00:00.06 3     1     56      3180K  0B     0B     39539
39538  mdworker_sha 0.0    00:00.07 3     1     56      3256K  0B     0B     39538
39536  com.apple.iC 0.0    00:00.38 2     1     55      4532K  0B     0B     39536
39502  bash         0.0    00:00.02 1     0     21      948K   0B     0B     39502
39501  login        0.0    00:00.01 2     1     31      1304K  0B     0B     39501
39500  Google Chrom 0.0    00:00.12 14     1     104     14M    4096B  0B     286
39499  Google Chrom 0.0    00:02.24 16     2     146     130M   4096B  0B     286
39485  Google Chrom 1.7    01:33.16 21     1     269     331M- 4096B  0B     286
39484  mdworker_sha 0.0    00:00.08 3     1     61      3368K  0B     0B     39484
39483  mdworker_sha 0.0    00:00.17 3     1     57      4852K  0B     0B     39483
39482  mdworker_sha 0.0    00:00.05 3     1     56      2880K  0B     0B     39482
```

Windows:

```
top - 09:02:42 up 12 min, 0 users, load average: 0.52, 0.58, 0.59
Tasks: 4 total, 1 running, 3 sleeping, 0 stopped, 0 zombie
%Cpu(s): 3.1 us, 3.4 sy, 0.0 ni, 93.4 id, 0.0 wa, 0.1 hi, 0.0 si, 0.0 st
KiB Mem : 16671964 total, 12716356 free, 3726256 used, 229352 buff/cache
KiB Swap : 5942908 total, 5942908 free, 0 used, 12811976 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM    TIME+  COMMAND
    1 root        20   0   8892    308    272  S   0.0   0.0   0:00.07 init
   23 root        20   0   8896    212    172  S   0.0   0.0   0:00.01 init
   24 mandili    20   0  16796   3428   3324  S   0.0   0.0   0:00.04 bash
   43 mandili    20   0  17620   2076   1504  R   0.0   0.0   0:00.01 top
```

The status of your processes managed by the kernel is changing constantly; therefore, you are seeing a dynamic view here. There are also a lot of options when running the **top** command depending on different administrative tasks. Here is a [link](#) for you to learn more! To exit from the view, simply press the 'Q' key on your keyboard.

6. Run **ps** which is another commonly used command when performing administrative tasks. You are able to view processes running by users, groups, process numbers, etc. If we want to view all processes that user **mandili** is running, just run **ps -u mandili**.

(Change **mandili** to the username you want to run.) **Mac:**

```
(base) Mandis-MacBook-Pro:~ mandili$ ps -u mandili
  UID  PID TTY          TIME CMD
  501   266 ??        0:10.73 /System/Library/Frameworks/LocalAuthentication.f
  501   267 ??        2:30.25 /usr/sbin/cfprefsd agent
  501   279 ??        4:56.42 /usr/libexec/UserEventAgent (Aqua)
  501   281 ??        5:28.44 /usr/sbin/distnoted agent
  501   283 ??        1:06.95 /System/Library/Frameworks/CoreTelephony.framewo
  501   284 ??        1:09.93 /usr/libexec/lsd
  501   285 ??       13:51.30 /usr/libexec/trustd --agent
  501   286 ??      443:37.90 /Applications/Google Chrome.app/Contents/MacOS/G
  501   288 ??        0:09.52 /System/Library/CoreServices/sharedfilelistd
  501   291 ??        4:35.84 /Applications/Stickies.app/Contents/MacOS/Sticki
  501   293 ??        4:51.83 /usr/libexec/secd
  501   295 ??        1:30.67 /System/Library/PrivateFrameworks/CloudKitDaemon
  501   297 ??        0:51.76 /System/Library/CoreServices/talagent
  501   298 ??       11:16.48 /System/Library/CoreServices/Dock.app/Contents/M
  501   299 ??        3:14.54 /System/Library/PrivateFrameworks/TelephonyUtili
  501   300 ??        4:58.17 /System/Library/CoreServices/SystemUIServer.app/
  501   302 ??       14:12.96 /System/Library/CoreServices/Finder.app/Contents
  501   303 ??       23:42.36 /System/Library/Frameworks/Accounts.framework/Ve
  501   304 ??        0:23.04 /System/Library/PrivateFrameworks/IDS.framework/
  501   307 ??        0:11.07 /usr/libexec/pboard
  501   309 ??        0:28.16 /System/Library/PrivateFrameworks/IMCore.framewo
```

Windows:

```
mandili@DESKTOP-55BNPEK: /mnt/c/Users/Mandili$ ps -u mandili
  PID TTY          TIME CMD
   24 tty1        00:00:00 bash
   46 tty1        00:00:00 ps
```

A list containing all processes that belong to user 'mandili' is shown. [Learn More](#)

7. **kill** command is used to stop an unresponsive process. To kill a process, you will need to locate the process name and process id (PID) using **top** or **ps**. [Learn More](#)

Additional resources for more commands

Sweet and short videos explain in detail about Linux commands. Some advanced commands we may not need right now. You can check them out if interested.

https://www.youtube.com/playlist?list=PLS1QuIWo1RIb9WVQGJ_vh-RQusbZgO_As