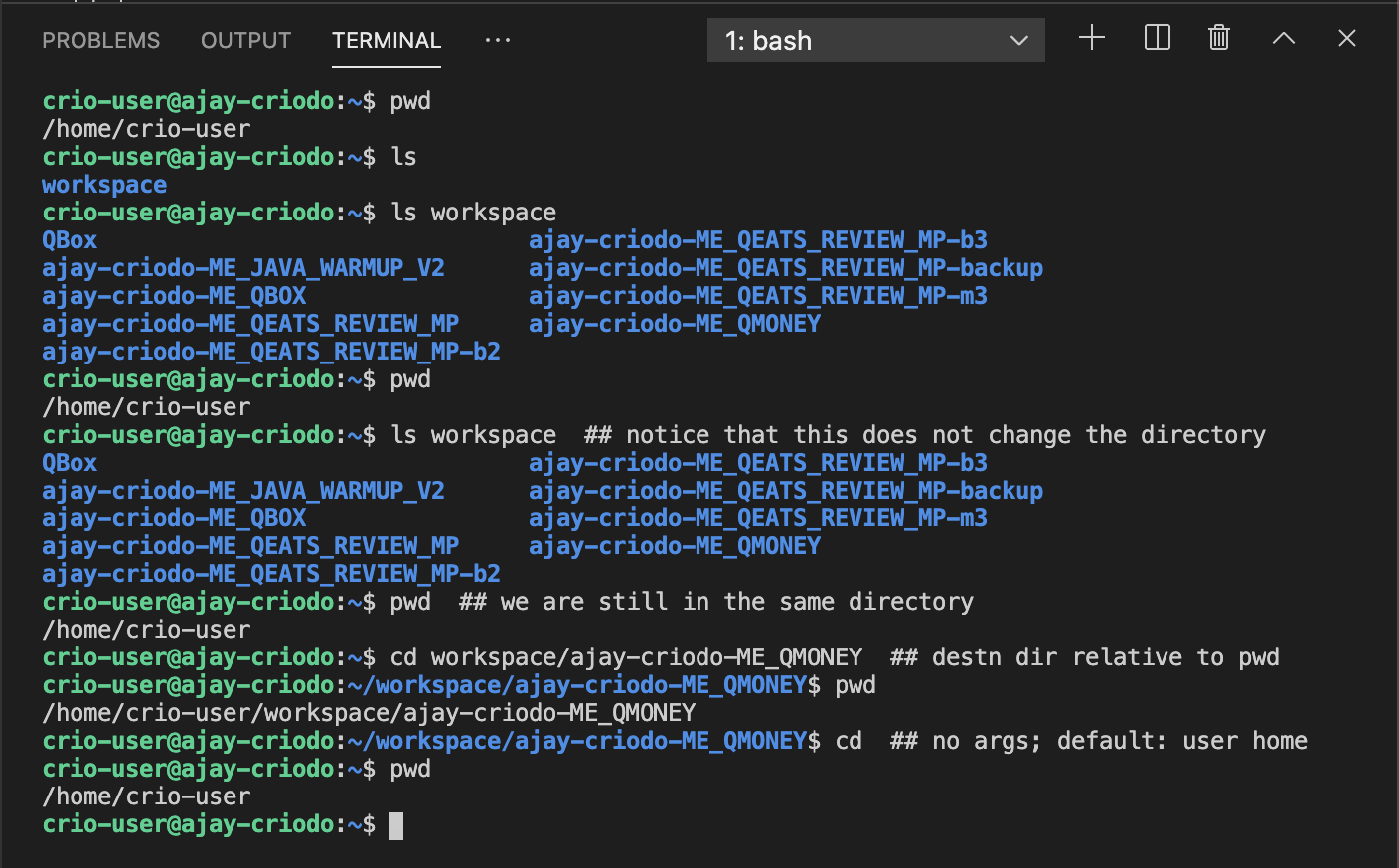
### GETTING STARTED WITH LINUX

90% of public cloud computing services run Linux. A competent software developer must know how to use a Linux system comfortably.

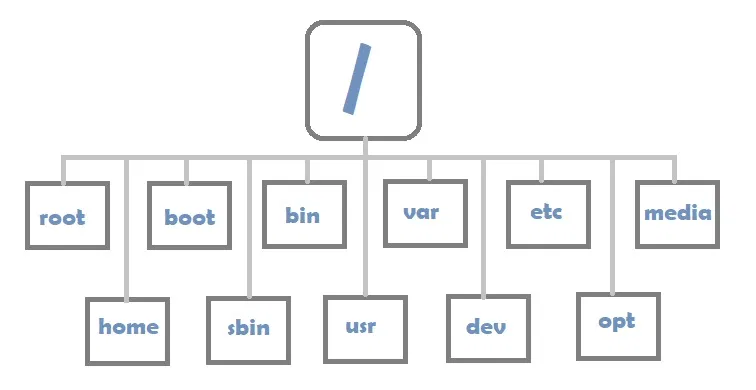
When you create your own Linux virtual machine (VM) from services like GCloud, AWS or Microsoft Azure, you don’t usually get access to a Graphical User Interface (GUI). You have to use the Linux terminal to operate and manage your VM.



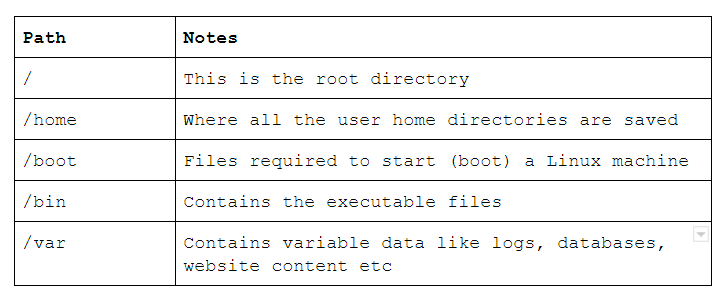
1. Directories in blue
2. Files are in white
3. Executables are in green (we will learn more about executable files later)

**MILESTONE 2:**

**LINUX DIRECTORY STRUCTURE**

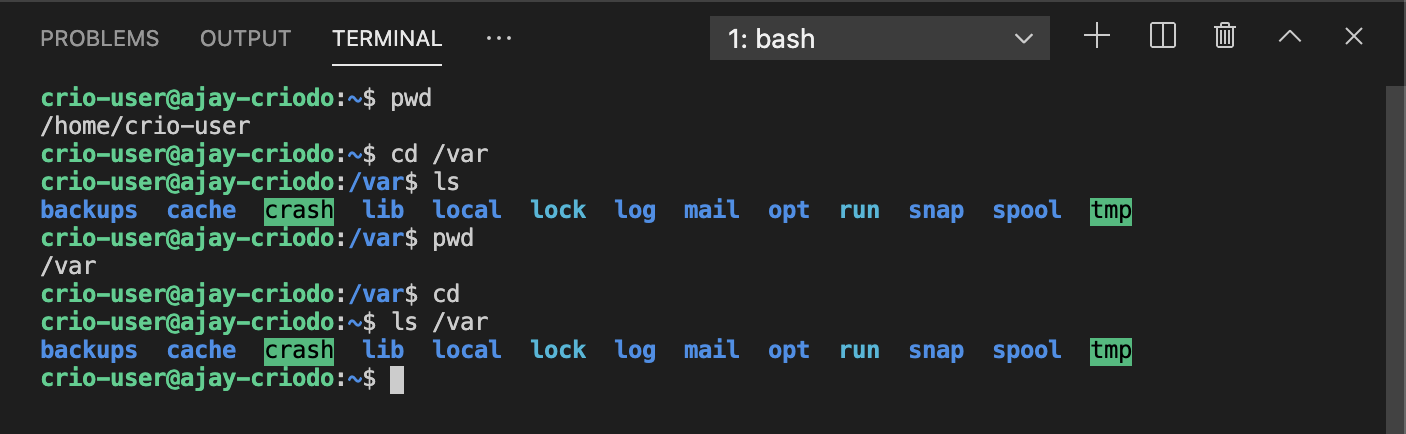


NAMING CONVENTION



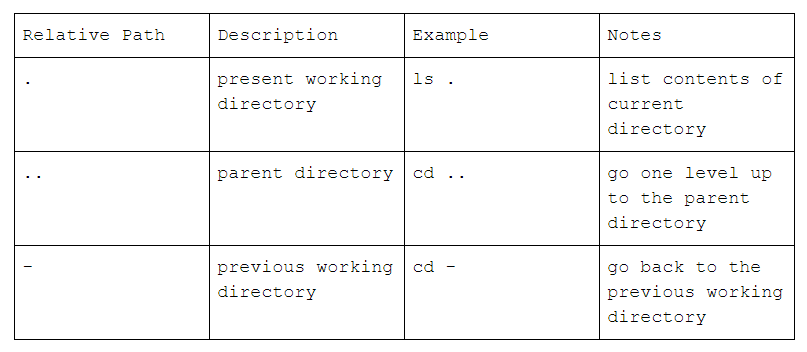
#### **Absolute path**

In the Linux terminal, there is always more than one way to do the same thing. Below are some options to display the contents of /var.



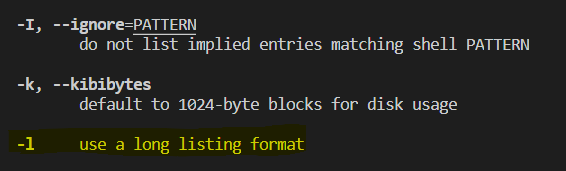
#### **Relative Path**

Relative paths are relative to the present working directory. A list of special relative paths are listed in the table below and additional examples are in the code block that follow.



Commands come with a "Manual" as well. We can access it using the man command followed by the name of the command we need to see the manual of. For ls, we do man ls.





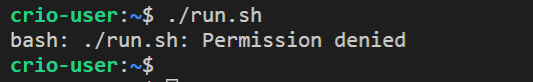
HIT q TO EXIT man ls

#### **Manipulating file permissions**

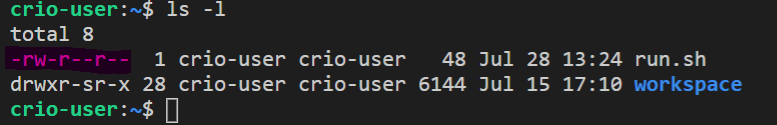
Executables are programs that can be run to perform some instructions.

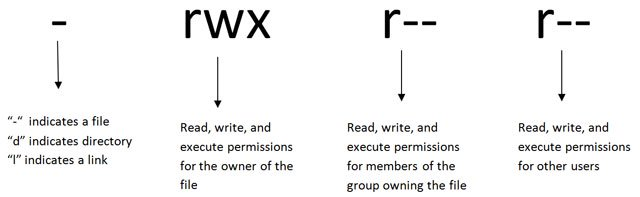
echo "echo 'Congratulations on running a bash script'" > run.sh

Now, how do we run an executable? We do ./<name-of-executable>. That’d be ./run.sh for us.



CHECK PERMISSIONS USING : ls -l





As the image tells us, **r** means permission to read data from a file, **w** means permission to write/edit a file & **x** means permission to execute a file.

Aha! If we see the permissions for our **run.sh** file, it doesn’t have any **x** in it & thus was throwing an error when we tried to run it. Just have to give e**x**ecutable permission & we’ll be good.

chmod command lets us **ch**ange the access **mod**e of a file. To add executable permission, we use chmod +x <filename>

chmod +x run.sh

CHECK FILE PERMISSIONS AGAIN, THEY HAVE CHANGED

-rwxr-xr-x 1 gitpod gitpod 48 Jan 15 10:28 run.sh

References

[Intro to Linux file systems](https://opensource.com/life/16/10/introduction-linux-filesystems" \t "https://learn.crio.do/home/me/_blank)

[ls command usage](https://www.rapidtables.com/code/linux/ls.html" \t "https://learn.crio.do/home/me/_blank)

[Understanding ls command output](https://detailed.wordpress.com/2017/10/28/understanding-ls-command-output/" \t "https://learn.crio.do/home/me/_blank)

[Creating symbolic links](https://www.cyberciti.biz/faq/creating-soft-link-or-symbolic-link/" \t "https://learn.crio.do/home/me/_blank)

[Absolute and relative paths](https://www.youtube.com/watch?v=ephId3mYu9o" \t "https://learn.crio.do/home/me/_blank)

[Linux file permissions](https://www.pluralsight.com/blog/it-ops/linux-file-permissions" \t "https://learn.crio.do/home/me/_blank)

### **MILESTONE : 3 DEALING WITH FILES AND DIRECTORIES**

#### **CRUD - Create, Read, Update, Delete**

I watched **K.G.F: Chapter 1** last night and I’m planning to store all movies I watch and some info about those, like synopsis, review etc. To start with, let’s create a directory structure like this in the home directory

mkdir kgf

mkdir kgf/part1

mkdir kgf/part2

touch part1/synopsis.txt

touch part2/synopsis.txt

Edit in file synopsis

nano part1/synopsis.txt   ---> will open the file for us to edit. After adding the required text

Hit ctrl + X and press Y to Save the changesand press enter

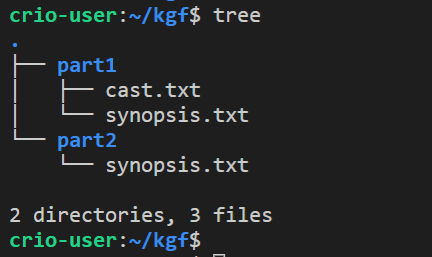
To verify whether changes have been saved or not use :

cat part1/synopsis.txt ------>print out contents of a file

Create one more file cast.txt in part 1

touch part1/cast.txt

To view the tree of directories



TO Remove cast.txt from part 1 an add it to kgf directory

1. Copy it to kgf dir.

cp part1/cast.txt . to **c**o**p**y the **part1/cast.txt** file to directory denoted by "**.**"  which is the current dir. Kgf.

1. Remove from the part 1

rm part1/cast.txt - to **r**e**m**ove the **part1/cast.txt** file

**To remove a dir** use rmdir dir\_name or rm -rf dir\_name

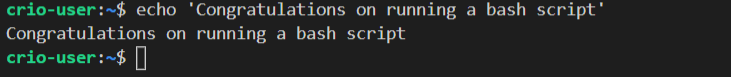
To rename a file use mv oldile newfile

#### MILESTONE 4 : MANIPULATING FILES AND OUTPUT REDIRECTION

#### echo "echo 'Congratulations on running a bash script'" > run.sh

ECO COMMAND

echo command is used to print out the value provided to it. By default, it prints the value to the terminal which is the "standard output".



Now, we want this command to be run when the **run.sh** file is executed (or run). For this, we need to add the whole command echo 'Congratulations on running a bash script' itself to the file. As we didn’t know about using the nano editor earlier (or echo for that matter :) ), we used the echo command itself to write the whole command to the file.

echo "echo 'Congratulations on running a bash script'" prints **echo 'Congratulations on running a bash script'** to the terminal. We can redirect the output instead to a file using the redirection operator **>**. You can verify this indeed was happening earlier by cating the content of the run.sh file from earlier to the terminal.

#### **Feeding Output of a command to the Input of another**

The **/proc/meminfo** file has info related to the RAM installed in our system. Try using cat to print its content. The first three lines are the attributes **MemTotal**, **MemFree** & **MemAvailable**. What if we needed to print out just the value of the **MemFree** attribute which denotes the amount of free memory?

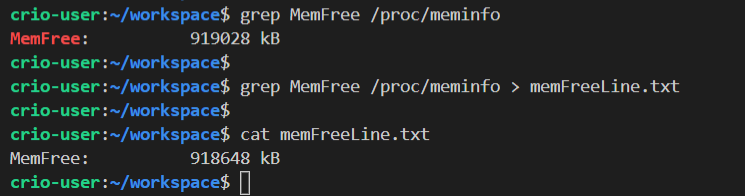
We’ll be required to perform a couple of actions.

Filter for the line in **/proc/meminfo** containing the attribute **MemFree**

Fetch the numerical value in that line by separating that column

grep command is used to filter/search for text using strings or patterns. It’s usage is grep <pattern> <file>. For us, it’d be grep MemFree /proc/meminfo. You’ll see only the line containing **MemFree** from **/proc/meminfo** filtered out. Let’s write it to a file using the redirection operator (**>**) we learnt in the previous section.

(The MemFree value is dynamic & hence can be different every time we print it out)

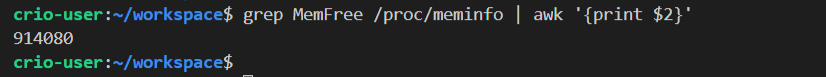


Now, how will we fetch only the numerical value?

awk can do that for us. It uses space as a field separator (by default space is used as field separator but other field separators can be specified) and separates text into columns. That’d mean, we have 3 words in our line - **MemFree:**, **918648** & **kB**. To print out the 2nd word, the usage is awk '{print $2}' memFreeLine.txt where $2 denotes the second word in the line.

That did the work for us and we have just the free memory in kB printed out to the terminal. Looking back, the **memFreeLine.txt** file was a by-product which we didn’t actually need. Is there some way to do this without having to create this file?

**Piping** is a type of redirection in Linux used to send output of one command to input of another command. We had earlier written the output of grep to a file & then used it as input to awk. Instead of this, we can use the Linux pipe operator (**|**) to redirect output of grep directly to input of awk.



As you can see, awk command isn’t given a file to read from but rather get its input from out of grep because of the use of the pipe operator (**|**).

You can continue linking together more commands like this, using multiple pipes, to achieve any of your goals!

Like we saw here, there can be multiple ways we can do a task in Linux

### References

1. [Piping and Redirection](https://ryanstutorials.net/linuxtutorial/piping.php" \t "https://learn.crio.do/home/me/_blank)

**MILESTONE 5 HADOOP ANALYSIS**

**MILESTONE 6 :**

#### **Useful shortcuts**

up arrow key → will bring up the last command that was executed. Each press will bring up the previous command executed.

history → will print out a list of the previous commands executed.

tab key → pressing the tab key can be used to auto-complete the directory and file names while typing paths or filenames.

#### **References**

Find pointers to Curious Cats questions [here](https://docs.google.com/document/d/1Yodppa7kld9O_KQqLoonbdyEaFfHdqIHUG_Xmwb8UhU/edit?usp=sharing" \t "https://learn.crio.do/home/me/_blank)

Handy Linux cheat sheet - [Cheat Sheet 1](https://www.linuxtrainingacademy.com/linux-commands-cheat-sheet/" \t "https://learn.crio.do/home/me/_blank) and [Cheat Sheet 2](https://learncodethehardway.org/unix/bash_cheat_sheet.pdf" \t "https://learn.crio.do/home/me/_blank)

Examples for Linux usage - [Examples](https://www.tldp.org/HOWTO/VMS-to-Linux-HOWTO/examples.html" \t "https://learn.crio.do/home/me/_blank)

Detailed Linux command list - [Linux Programming Commands](https://goalkicker.com/LinuxBook/" \t "https://learn.crio.do/home/me/_blank)

#### **Why Linux?**

A majority of systems around the world run some form of Linux. These range from enterprise and desktop servers to smartphones.

60 to 70% of all Web Servers in the world run some form of Linux/Unix and ~90% of all cloud computing happens on Linux based servers.

Most of the smartphones in the world run on Linux.

Mac is based on Unix and supports a terminal where these commands can be run.

#### **In the Real World**

What does Linux power - [25 Awesome unexpected things powered by Linux](https://www.omgubuntu.co.uk/2016/08/25-awesome-unexpected-things-powered-linux" \t "https://learn.crio.do/home/me/_blank)

What you could use Linux additionally for - [Alternative examples of Linux usage](https://lifehacker.com/top-10-uses-for-linux-even-if-your-main-pc-runs-window-1513172815" \t "https://learn.crio.do/home/me/_blank)

#### **Special update**

Windows users don’t have to fret with having to have a virtual machine to run Linux anymore - we have a treat by Microsoft, called the Windows Subsystem for Linux, which lets us Windows folks use the CLI without any need to waste precious memory installing the gigantic VM and having your computer go into coma every time you boot. Check out more details - [https://docs.microsoft.com/en-us/windows/wsl/about](https://docs.microsoft.com/en-us/windows/wsl/about" \t "https://learn.crio.do/home/me/_blank)