

SUMMER SCHOOL 2021

Introduction to the Shell

Dr Krishna K. Govender

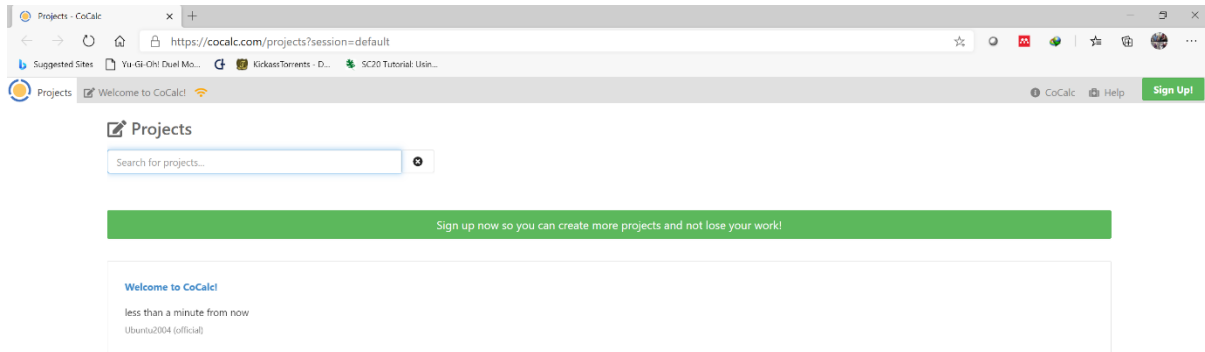
Summary of commands used in this lecture

Command	Meaning
cal	Gives you a terminal view of the calendar
date	Provides the current date and time
clear	Clears the terminal screen
man	Display the manual page for a specific Linux command
pwd	Present working directory
ls	List items in current directory
echo	Print text to screen
df	Checks amount of free disk space
free	Checks the memory on a machine

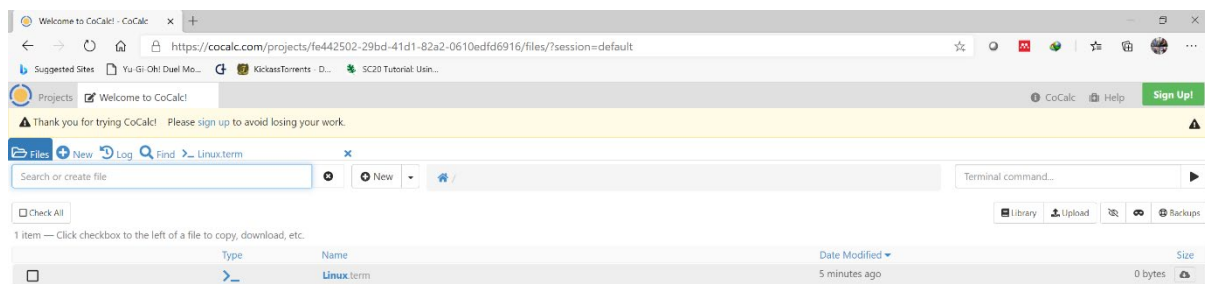
- If you followed the steps outlined in the previous lecture (**Getting Set Up**) and you closed your browser, you will need to first navigate to:

[CoCalc – Collaborative Calculation and Data Science](https://cocalc.com)

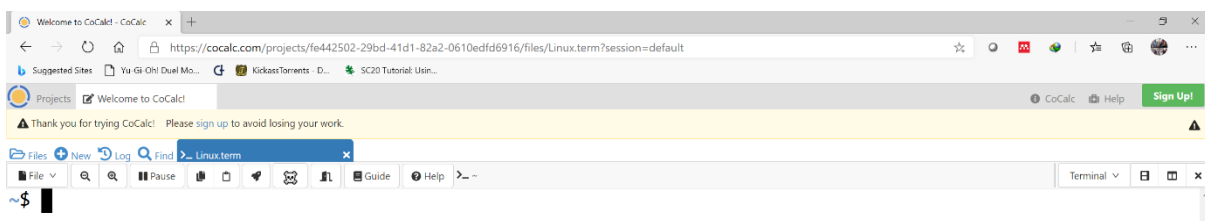
- You should see a page that looks like:



- Clicking on Welcome to CoCalc! will take you here:



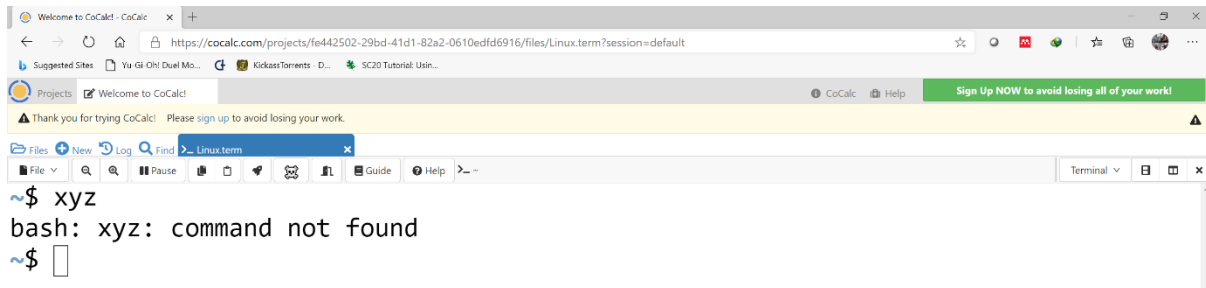
- Now click on Linux.term and you will end up on a terminal, where we will be executing our commands:



- If you are working offline, then you should have your **MobaXterm terminal** (Windows users) or a **local terminal** (Linux and MacOS users) open. Please refer to the **Getting Set Up** lecture if you are not sure about how to do this.
- All commands that will be provided in **BOLD** and center aligned from here onwards are commands that can be typed/executed directly in a terminal.

- Let's begin with some random command (any command typed in a terminal should always be followed by the **Enter** key for the command to take effect):

xyz



The screenshot shows a web browser window with the CoCalc interface. The terminal pane at the bottom displays the following text:

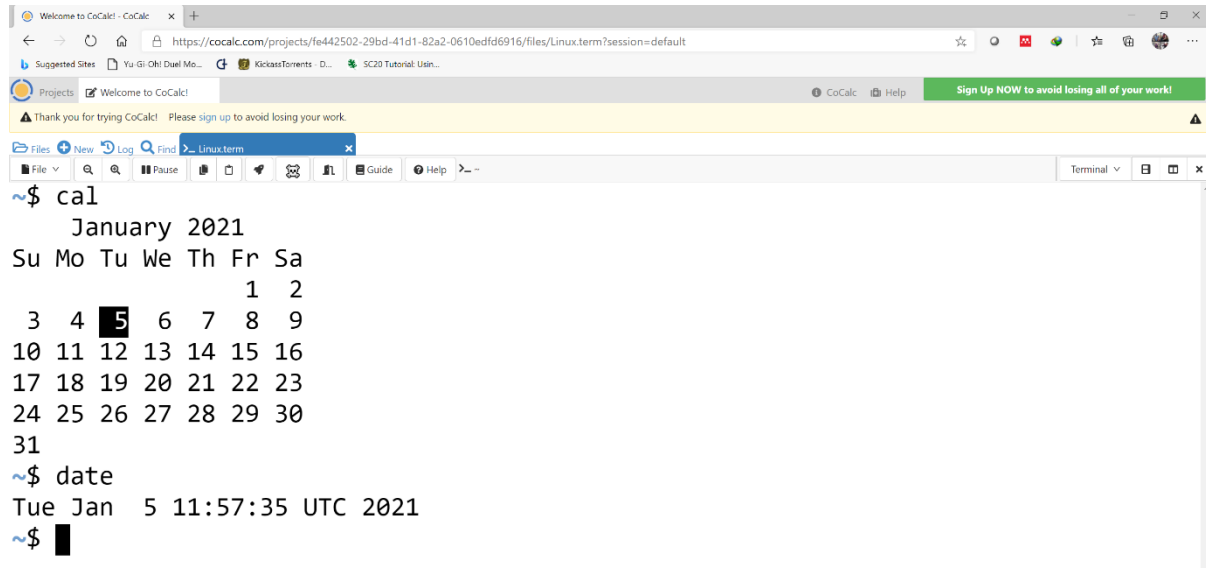
```
~$ xyz
bash: xyz: command not found
~$
```

- As you can see the terminal will accept anything you type, however in the current case the command **xyz** is one that is not recognized and produces a **command not found** error.

- Now let's display the calendar and the current date and time:

cal

date

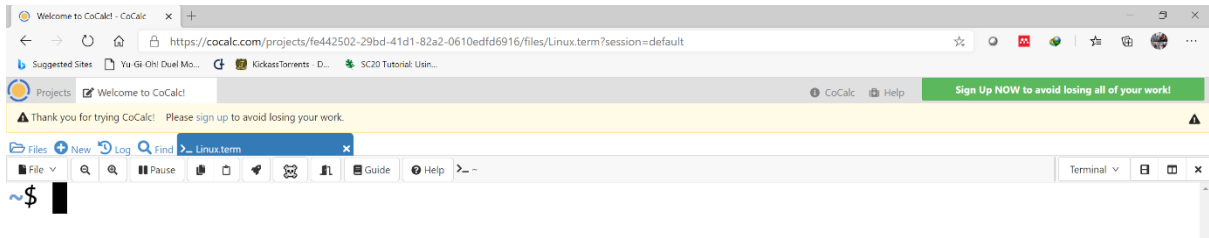


The screenshot shows the same CoCalc terminal window. The terminal pane displays the following text:

```
~$ cal
      January 2021
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
~$ date
Tue Jan  5 11:57:35 UTC 2021
~$
```

- Should you wish to remove the contents displayed on the current screen you can use:

clear



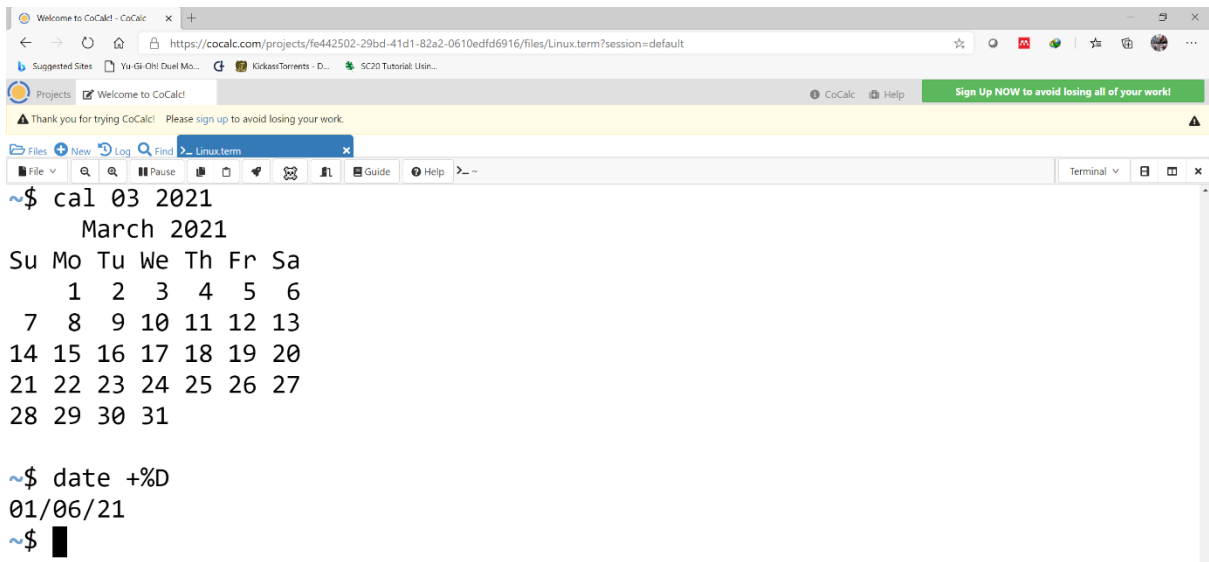
- **clear** merely clears the screen so that we can work on a fresh new screen. Don't worry the commands you typed up to this point are not lost. I will show you, later in the course, how you can retrieve all the commands that have been executed on the terminal.

- Most Linux commands have additional flags that you can specify after the command to provide additional/less information.

Let's use the calendar and date commands again, but this time we will add in some extra flags:

cal 03 2021

date +%D



- The calendar command now provides information for the 3rd month of 2021 (March) and date provides the date in the format mm/dd/yy.

- There are various other flags that can be used for **cal** and **date**.

These can be viewed by making use of the manual pages (man pages) for these commands:

man cal

q

```

CAL(1)                                BSD General Commands Manual                                CAL(1)

NAME
    cal, ncal – displays a calendar and the date of Easter

SYNOPSIS
    cal [-31jy] [-A number] [-B number] [-d yyyy-mm] [[month] year]
    cal [-31j] [-A number] [-B number] [-d yyyy-mm] -m month [year]
    ncal [-C] [-31jy] [-A number] [-B number] [-d yyyy-mm] [[month] year]
    ncal [-C] [-31j] [-A number] [-B number] [-d yyyy-mm] -m month [year]
    ncal [-31bhjJpwySM] [-A number] [-B number] [-H yyyy-mm-dd] [-d yyyy-mm]
        [-s country_code] [[month] year]
    ncal [-31bhJeoSM] [-A number] [-B number] [-d yyyy-mm] [year]

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

```

~$ man cal
~$
  
```

- The man page for **cal** provides you with a list of additional flags that can be used. You should be able to scroll through this document using the arrow keys on your keyboard. One can exit a man page by using **q** and you will end up back on the terminal where you can type commands again.

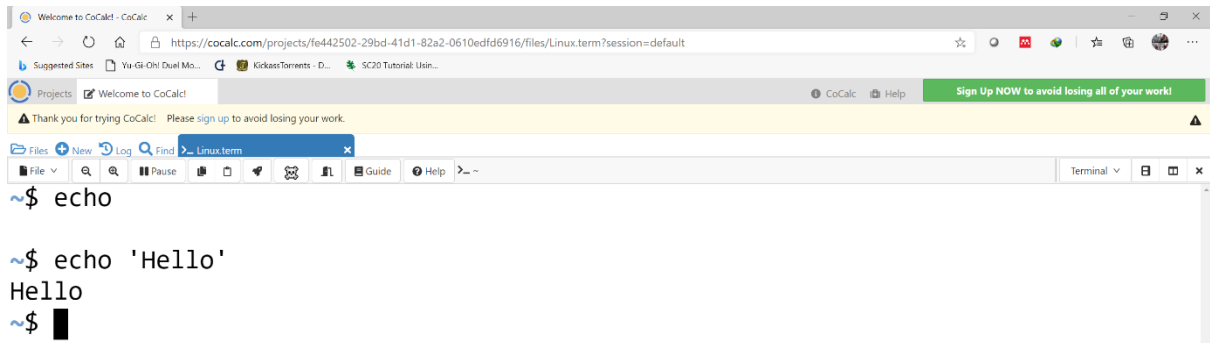
For those using **MobaXterm**, the man pages look very different to that displayed above. It is for this reason that it would be best for **Windows** users to [Google](#) “man cal linux” and obtain the relevant information from the first link provided by Google.

A similar approach should be followed for the man pages of any other Linux command used in this course.

- I will leave it up to you to try out some flags for **cal** to see what they do.
You should also have a look at the man page for **date** and test a few of those flags as well.

- To print something to the screen you can **echo** the statement:

echo
echo 'Hello'



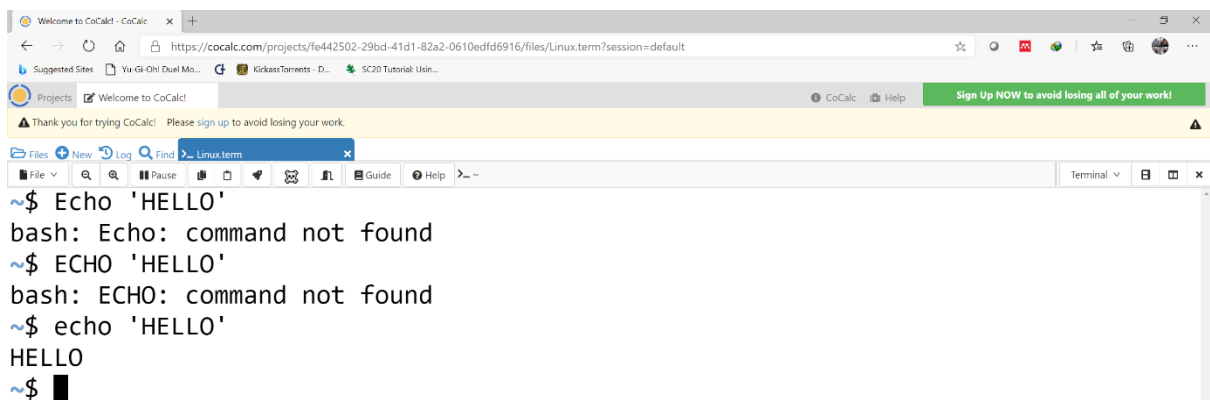
A screenshot of a web browser window displaying a CoCalc terminal. The terminal interface includes a top bar with navigation icons and a status bar with a 'Sign Up NOW' button. The terminal window shows a prompt '~\$' followed by the command 'echo', which results in no output. The next prompt '~\$' is followed by the command 'echo 'Hello'', which results in the output 'Hello'. The final prompt '~\$' is followed by a cursor, indicating the command has not yet been executed.

```
~$ echo

~$ echo 'Hello'
Hello
~$
```

- If you use **echo** on its own then nothing will appear on the screen, whereas if you put in a statement like Hello after the **echo** then the word Hello will display on screen.
- At this point it is important to take note that Linux commands are case sensitive.
To test this, we will run a few commands:

Echo 'HELLO'
ECHO 'HELLO'
echo 'HELLO'



A screenshot of a web browser window displaying a CoCalc terminal. The terminal window shows a prompt '~\$' followed by the command 'Echo 'HELLO'', which results in the error message 'bash: Echo: command not found'. The next prompt '~\$' is followed by the command 'ECHO 'HELLO'', which also results in the error message 'bash: ECHO: command not found'. The final prompt '~\$' is followed by the command 'echo 'HELLO'', which results in the output 'HELLO'. The final prompt '~\$' is followed by a cursor, indicating the command has not yet been executed.

```
~$ Echo 'HELLO'
bash: Echo: command not found
~$ ECHO 'HELLO'
bash: ECHO: command not found
~$ echo 'HELLO'
HELLO
~$
```

- The first two commands produce **command not found** errors, while the last prints out the text as expected.

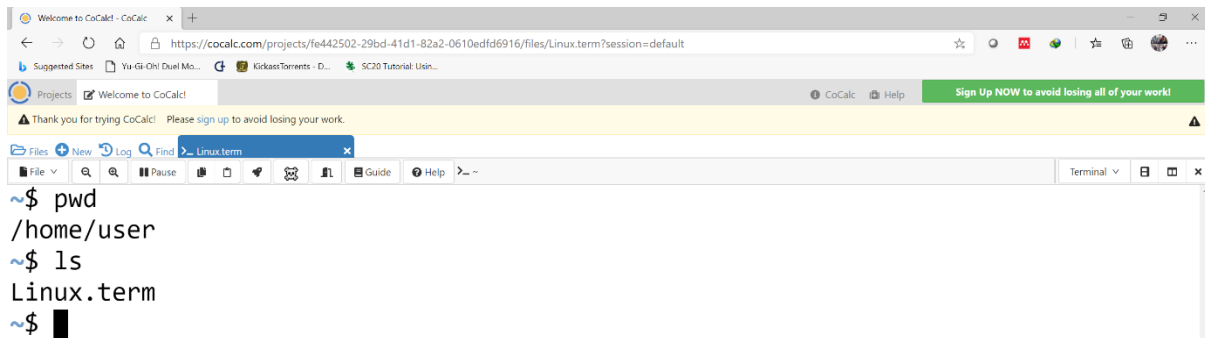
This means that Linux commands should always be in lowercase, while things that appear after the Linux command (HELLO, in the example above) need not be only in lowercase.

- Often you will want to determine your **present working directory** and list the items in the current directory.

This can be done by using:

pwd

ls



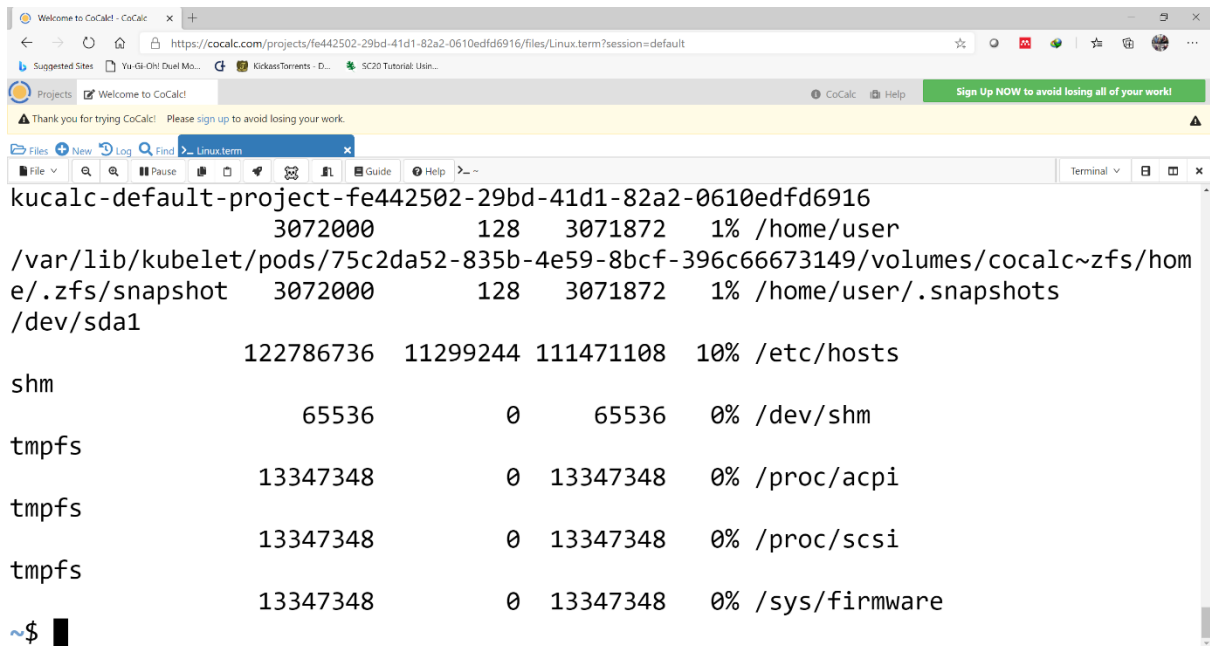
```
~$ pwd
/home/user
~$ ls
Linux.term
~$
```

- **pwd** shows that I am working in the folder **/home/user** and **ls** shows that I have a file in my directory called Linux.term.

If you chose to work in CoCalc as I am doing you should see something similar however, if you are working offline in either Windows, Linux or MacOS then your output might be slightly different.

- When you wish to determine the amount of space you have on your system you can make use of:

df



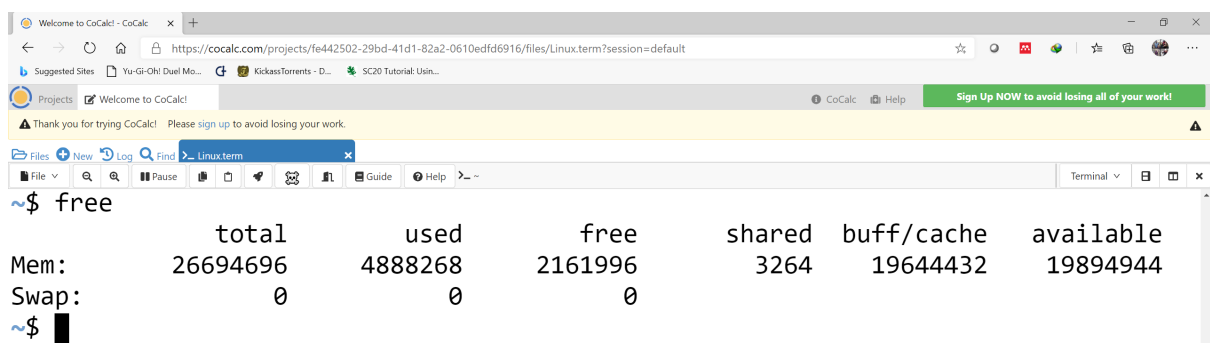
```
kucalc-default-project-fe442502-29bd-41d1-82a2-0610edfd6916
3072000      128    3071872    1% /home/user
/var/lib/kubelet/pods/75c2da52-835b-4e59-8bcf-396c66673149/volumes/cocalc~zfs/home/.zfs/snapshot    3072000      128    3071872    1% /home/user/.snapshots
/dev/sda1
122786736 11299244 111471108 10% /etc/hosts
shm
65536      0      65536    0% /dev/shm
tmpfs
13347348      0 13347348    0% /proc/acpi
tmpfs
13347348      0 13347348    0% /proc/scsi
tmpfs
13347348      0 13347348    0% /sys/firmware
~$
```

- There are several entries above the ones that I display here, but I want us to focus on the **/home/user** located at the top of the image shown.

What is shown is that the size of the drive is 3072000 KB, 128 KB of space is being used and 3071872 KB are still available, equating to 1% of space currently being used. You can scroll to the top of the output to see the labels for the different columns.

- There is also a command that can be used to check the available memory on a system:

free



```
~$ free
              total        used        free      shared  buff/cache   available
Mem:      26694696    4888268    2161996        3264    19644432    19894944
Swap:              0              0              0
~$
```

- The command above shows us that there is 26694696 KB of memory in total, 4888268 KB being used and 2161996 KB of free memory.

- If you are running these commands locally using Windows, Linux or MacOS then the numbers you see will differ from those that I provided above, but the primary column headings should be the same.
- This brings us to the end of **Introduction to the Shell**.
- At this point you should give **Exercise 1** a try and see if you can find the appropriate flags needed to provide the desired outcome.