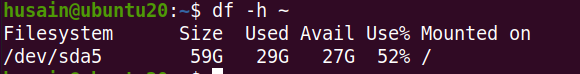
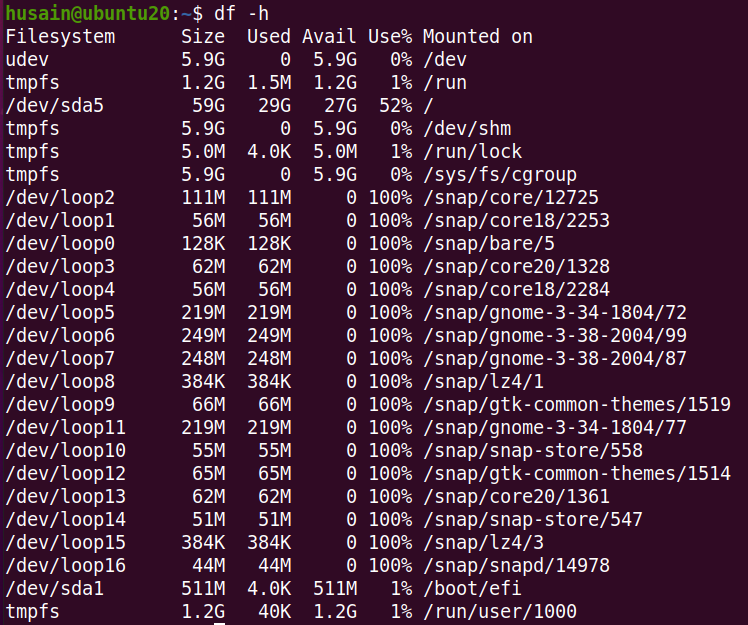
1. **Whoam**i: will display the user identity. // this command takes no arguments and has no options.
2. **ID**: returns the user ID. Or the user’s group ID
3. **Uname**: stands for unix name. returns OS information such as kernel name or version number.
4. **Uname -s -r**: both name and version
5. **Uname -v**: view all version information
6. **Df:** shows systems disk usages.
7. **Df -h ~** : home directory



1. **Df -h**: makes output human readable expressing disk usage in units like gb or tb.



1. **Ps**: the currently running process.
2. **Ps** -u root: to show only the processes the process id and the time and minutes and seconds that each process has been running.
3. **Top**: acts as a task manager and will show running processes and their resource usage.
4. **Top -n 3**: will show 3 running processes only.

(tasks are sorted by CPU usage. We simplified the output for the sake of the presentation, but top provides many other details, like memory usage and the executable file location. By default, tasks are sorted by CPU usage. We simplified the output for the sake of the presentation, but top provides many other details, like memory usage and the executable file location)

1. **Echo**: prints the given string, or the value of a given variable.

If you enter ”echo” you are basically saying “print nothing” so you just get a newline, If you would like to echo a single word without whitespace, such as “hello” you can just type “echo hello”.

1. **Date**: command displays the current system date and time. Entering “date” returns the default date format: day of week, day, month, year, time, and the time zone.
2. **date "+%j" day of %Y:**

The “percent letter” symbols in these arguments are used to extract properties of the current date. For example, “percent Y” is replaced by the current year, 2021

1. ls: command will list the files and directories within a directory. If you enter ls in your home directory, you will see all files and directories that your home directory contains.
2. Ls Downloads: this command will show you all the files inside Downloads directory.

1. ls -l: will show child files and directories in a longer, more detailed format. And you can see all child files and directories along with additional details, such as: permissions, last-modified date, and owner
2. cd: command is used to change the current working directory.

Expl: Now, suppose you are in your home directory, and you wish to move to your Documents folder, which happens to be contained in your home

directory You simply enter “**cd Documents**”. Now, if you enter “**pwd**”, you can see that you are indeed in your Documents folder. The “**cd**” command enables you to change directories with either an absolute path to the directory.

1. **Cd ..** : To get to your immediate parent directory.

**Example:** suppose you are in /home/noor/Downloads/movie/March

Now you want back to movie directory. Just type cd .. (cd space dot dot).

It will bring back to you in /home/noor/Downloads/movie

1. **Cd ~** : to get to your home folder directly without having to explore to get there, then you can simply use the “tilda” symbol

**Example**: directly back to /home/noor

From the above long path directories just type cd ~. This command directly brings back you to the home folder. Tilda represents your absolute path to home

1. **find**: powerful tool. that will find the path to every file that matches a given user-specified criterion.

**Example**: Now, suppose your Documents folder has a file structure as shown, with two subfolders, each containing a few files. And let’s say, you are currently in your Documents folder, and you want to find the path to all files called” a dot text” within your “Documents” tree. To do so, you can simply type “find dot minus name a dot txt”. The “dot” argument means “search within here”, that is, from your current working directory. To perform a case-insensitive version of your search, you can use the “minus i name” option instead of “ i name”. And, indeed, you find the same file, plus another file with the same name except that the letter “A” is uppercase. In this video you learned that: The “ls” command lists all files and directories contained within a specified directory tree And, you can navigate directories using the “cd” command.

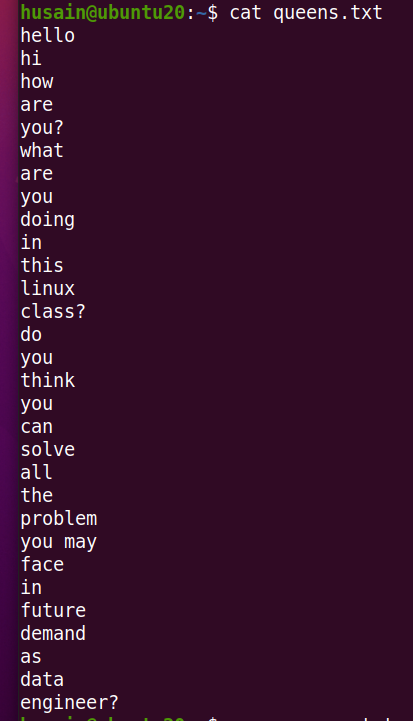
1. **Cat**: to print the entire file to standard output.

Suppose, when you use cat command, you might see 15/20 line in terminal, but your file contains more than 50 lines. How can you see all the lines? But thankfully there is another command name ‘more’.

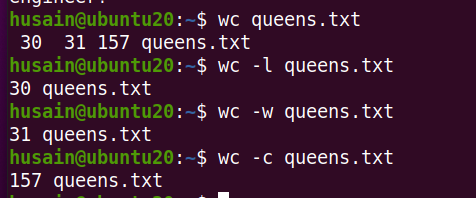
1. **more**: allows you to view a file’s content in a page-by-page format.

using space bar you can move and read rest of the lines which are not shown in the terminal

1. **head**: shows first 10 lines.
2. **Head** -n 3: show first 3 lines of the file. You can use random number instead of 3 as per your wish.
3. **tail**: shows last 10 lines.
4. **Tail** -n 4: shows last 4 lines. You can use any random number.
5. **wc**: to count the number of characters, words, or lines in your file.

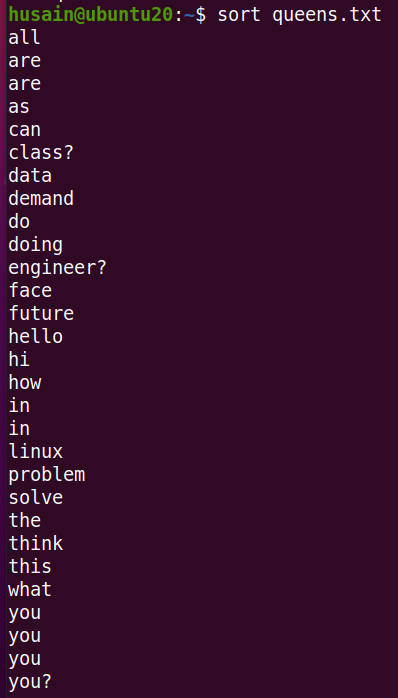


In this queens.txt file contails n numbers of lines. How can we determine the line? Wc is necessary here.

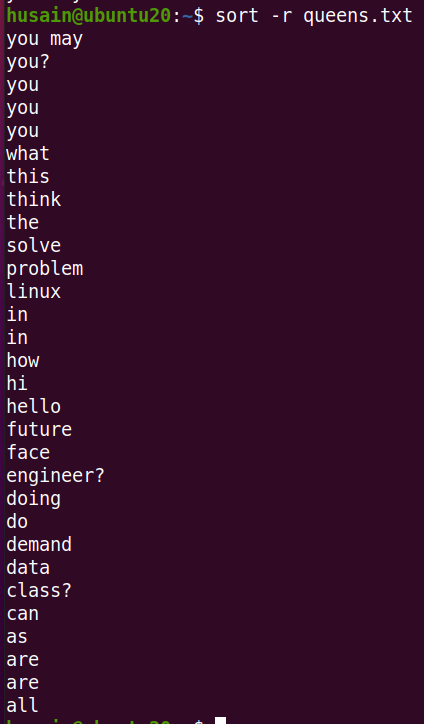


Here wc is returning that queesn.txt has 30 lines, 31 words and total word count(characters) is 157 including /n(entering new line character).

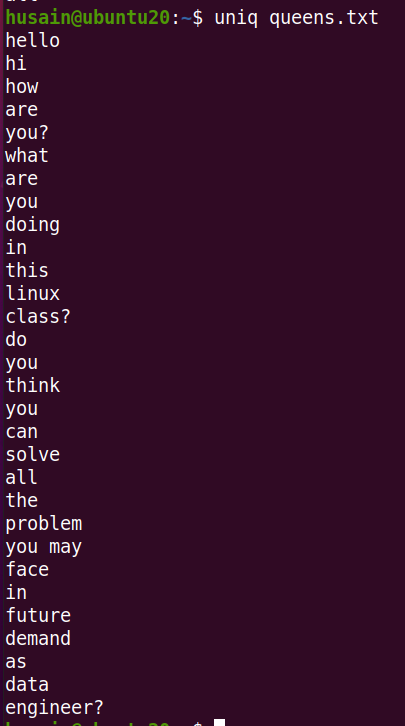
1. **wc -l:** total number of lines.
2. **wc -w**: total number of words.
3. **wc -c**: total number of characters.
4. **sort**: command sorts the lines of a file alpha-numerically and prints the sorted result to standard output.



1. **Sort** -r: get back the lines sorted in the reverse order.

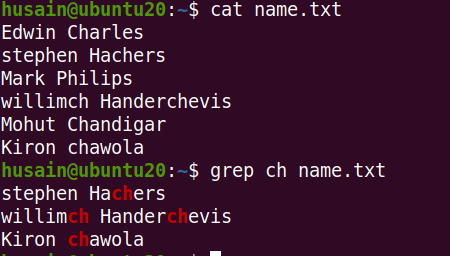


1. **Uniq**: to filter out the repeated lines and gives the unique value.

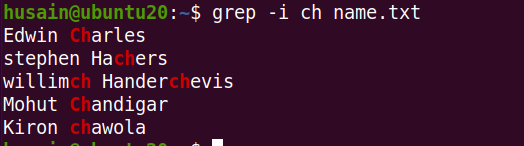


1. **grep: (global regular expression print)**

Suppose you have a list of famous people's names stored in a file, which you view by entering "**cat name.text**". You can use grep to find all lines in “**name.txt**” that contain the consecutive characters “c” and “h”. To do this you give **grep** the matching term, “ch”, followed by the filename. The output returned two results. Here ‘ch’ is lower case. And look at the result.



1. **Grep -i (ch)**: here ch is optional. It depends on your real case. with the – i option returns an extra result which has an upper case with case insensitive. It will bring out all capital or lower-case letter containing ch or Ch or CH. It will search only for ch. Nothing else.



Some of the frequently used options for grep are:

Option Description

-n Along with the matching lines, also print the line numbers

-c Get the count of matching lines

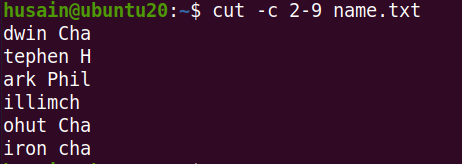
-i Ignore the case of the text while matching

-v Print all lines which do not contain the pattern

-w Match only if the pattern matches whole words

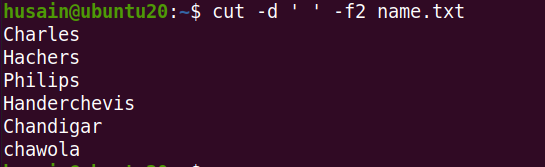
1. **Cut**: command to extract specific sections from each line in your file. Extract a section from each line from a file.

**Cut -c 2-9** name.txt meaning: for example, Edwin Charles, stephen Hachers, Mark Philips, willimch Handerchevis, Mohut Chandigar, Kiron chawola. . Names are cutted from ***dwin cha, tephen H, ark Phil, illimch, ohut Cha and iron cha.*** From character no 2 to 9 are kept and rest of the characters are cut.

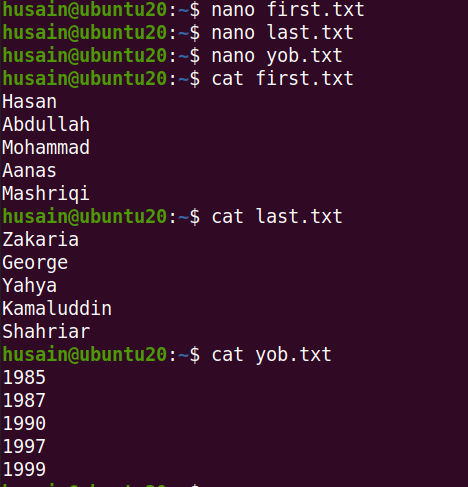


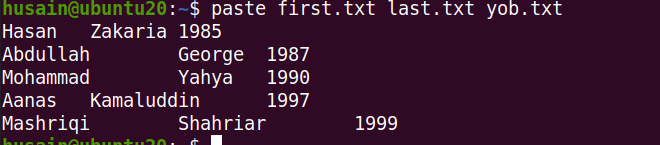
1. **Cut -d ‘ ’-f2<file name>:**

Explanation: Suppose you want to extract just the last names of each person in your list. You know that each line of your list is made up of two fields: one is the first name and the other is the second name. These fields are separated by a space. Here, you’re cutting to extract the second field from each line, using the "**-f2**" option.



Before discussing no 40, look at the below picture. I have created 3 separate files of first name, last name and year of birth of five people.



1. Paste <file name1 file name 2 file name 3…so on>: it combiles all the files and print on the screen. 

You can view these files as a table by entering **“paste first.txt last.txt yob.txt”** Notice the three columns are nicely aligned. This is because paste uses a "tab" as the default delimiter. And, for example, you can see that **Hasan Zakaria was born in 1985.**

1. **Paste -d “ , ” <file name1…file name N>:**

For example, you can use a comma as a delimiter by entering “**paste -d**" with a comma surrounded by double quotes, together with the "first", "last", and "YOB" text files.

