



# Lecture 5

LINUX essentials  
Dr .Sara Mohamed



# **Viewing Files**

# Viewing Files

- ▶ There are a few Linux commands available to view the content of files. The cat command, which stands for “**concatenate**”, is often used to quickly view the contents of small files.
- ▶ One of the most basic methods for opening text files is using the cat (short for concatenate) command. To open files using cat.
- ▶ The cat command will display the entire contents of the file, hence why it is mainly recommended for smaller files where the output is limited and does not require scrolling.

# Syntax for cat command

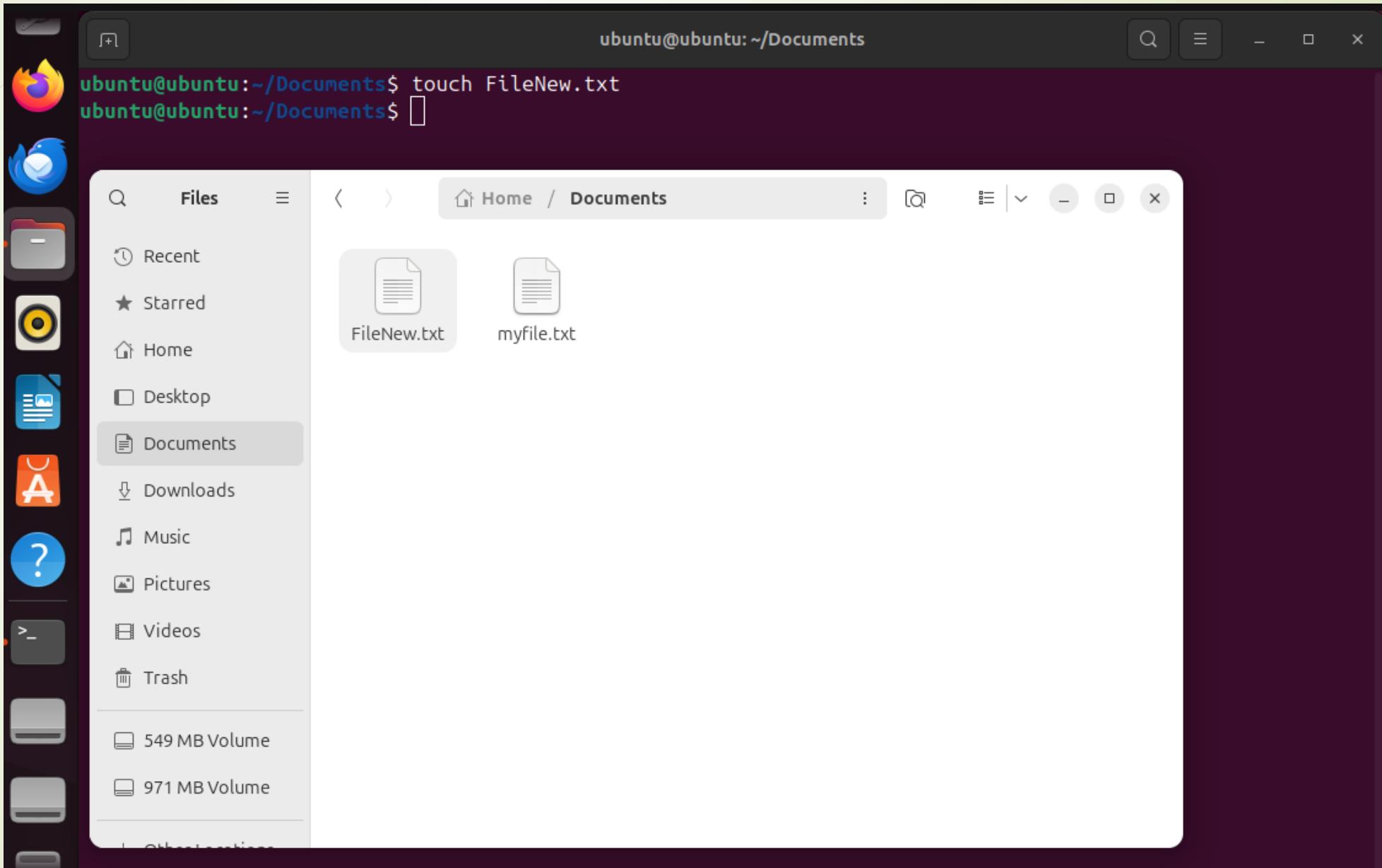
```
cat [OPTIONS] [FILE]
```

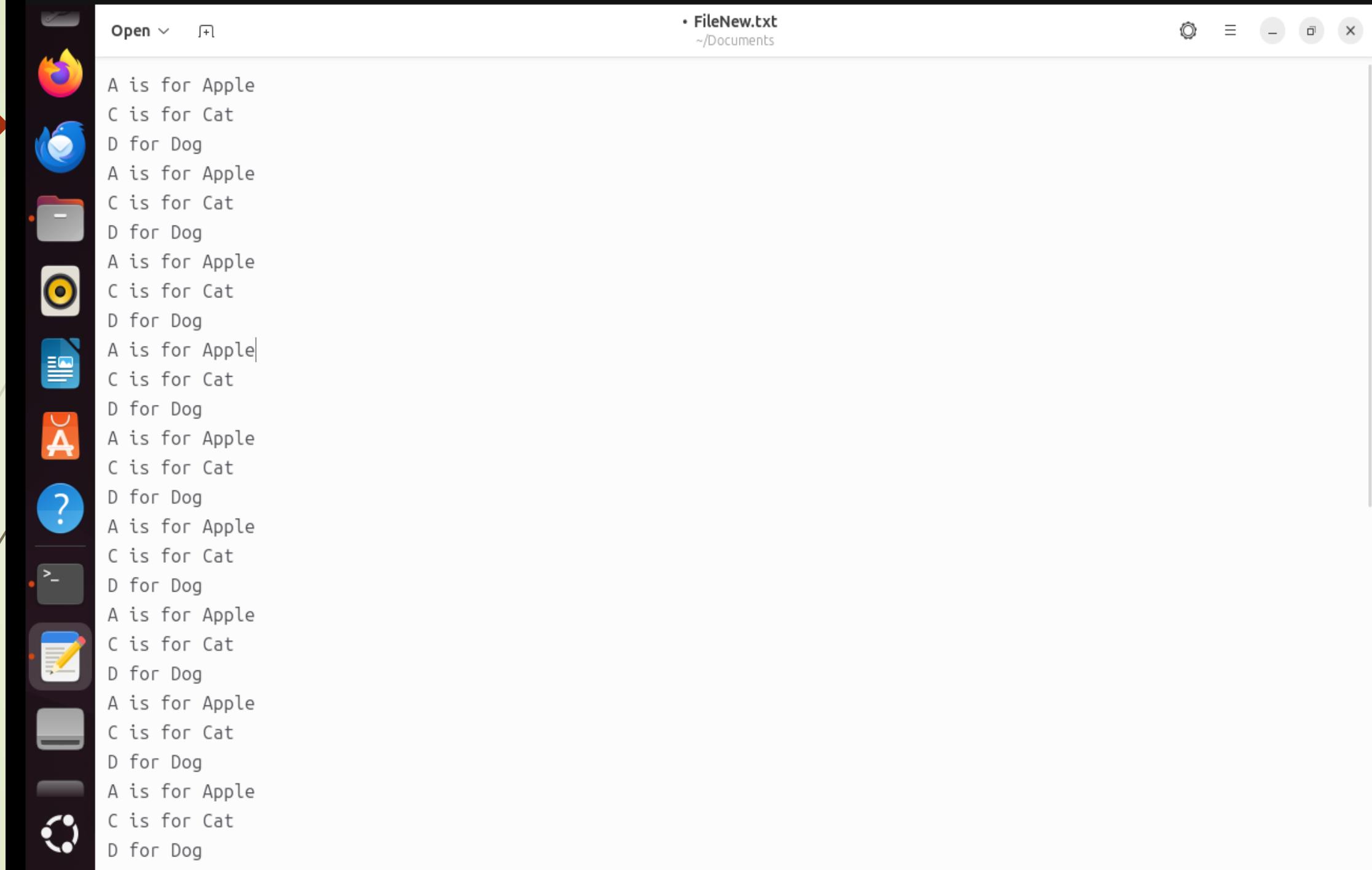
To view the contents of a file using the cat command, simply type the command and use the name of the file you wish to view as the argument.

The **cat** (concatenate) command is one of the simplest ways to view a file. It reads the entire contents of a file and prints them to the standard output.

# Example for **cat** command

Create new  
file called  
**FileNew.txt**







Display all the  
content of  
**FileNew.txt**

ubuntu@ubuntu:~/Documents\$ cat FileNew.txt

A is for Apple

C is for Cat

D for Dog

A is for Apple

C is for Cat

D for Dog

A is for Apple

C is for Cat

D for Dog

This a lesson for linux files

C is for Cat

D for Dog

A is for Apple

C is for Cat

D for Dog

A is for Apple

C is for Cat

D for degree

A is for Apple

C is for Cat

D for Dog

A is for Apple

C is for Cat

D for Dog

A is for Apple

C is for Cat

End of lines

# Using head and tail

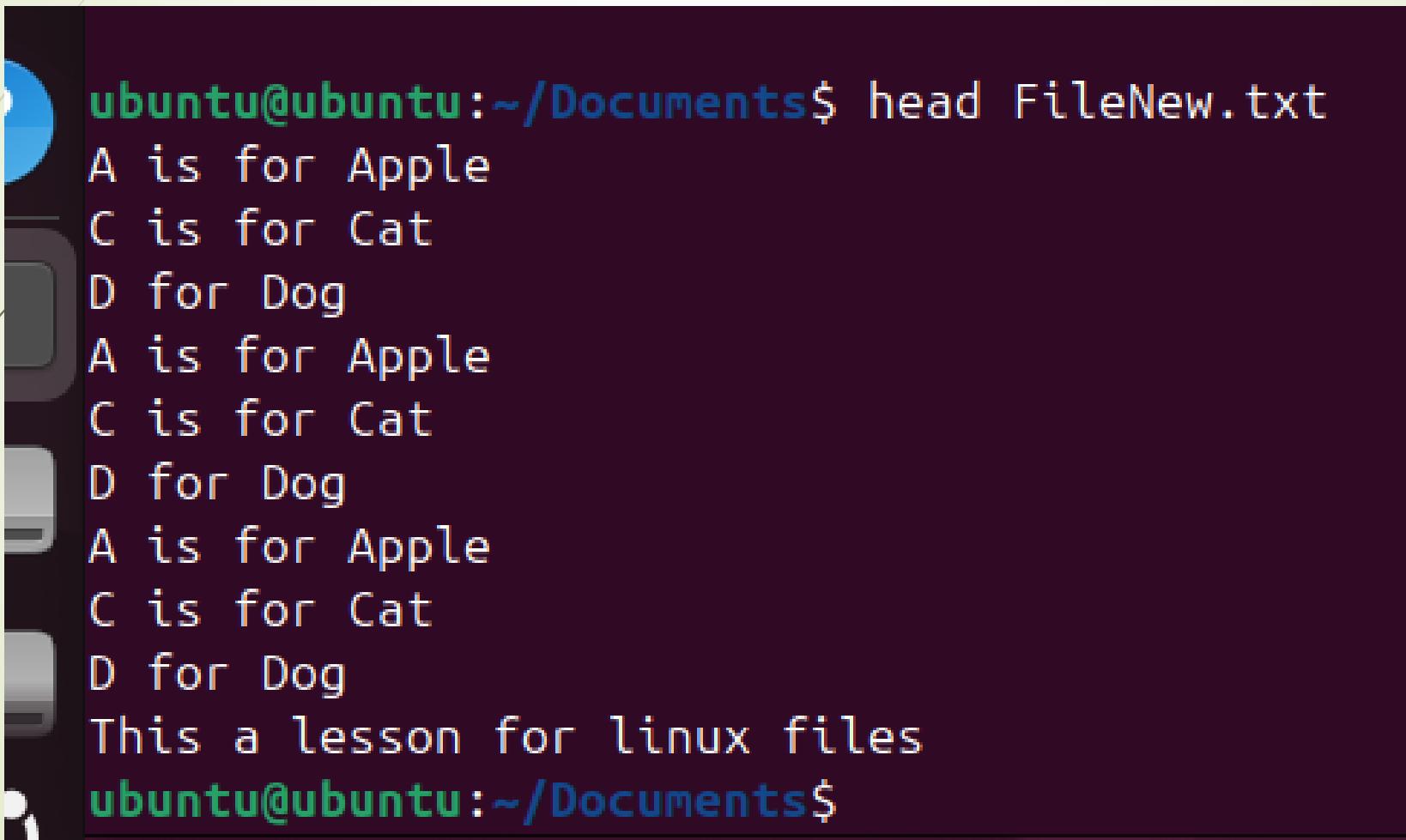
- ▶ Another way to view the content of files is by using the **head** and **tail** commands. These commands are used to view a select number of lines from the top or bottom of a file. The **head** command is used to display the first few lines of a file, while the **tail** command is used to display the last few lines.

```
head [OPTIONS] [FILE]
```

```
tail [OPTIONS] [FILE]
```



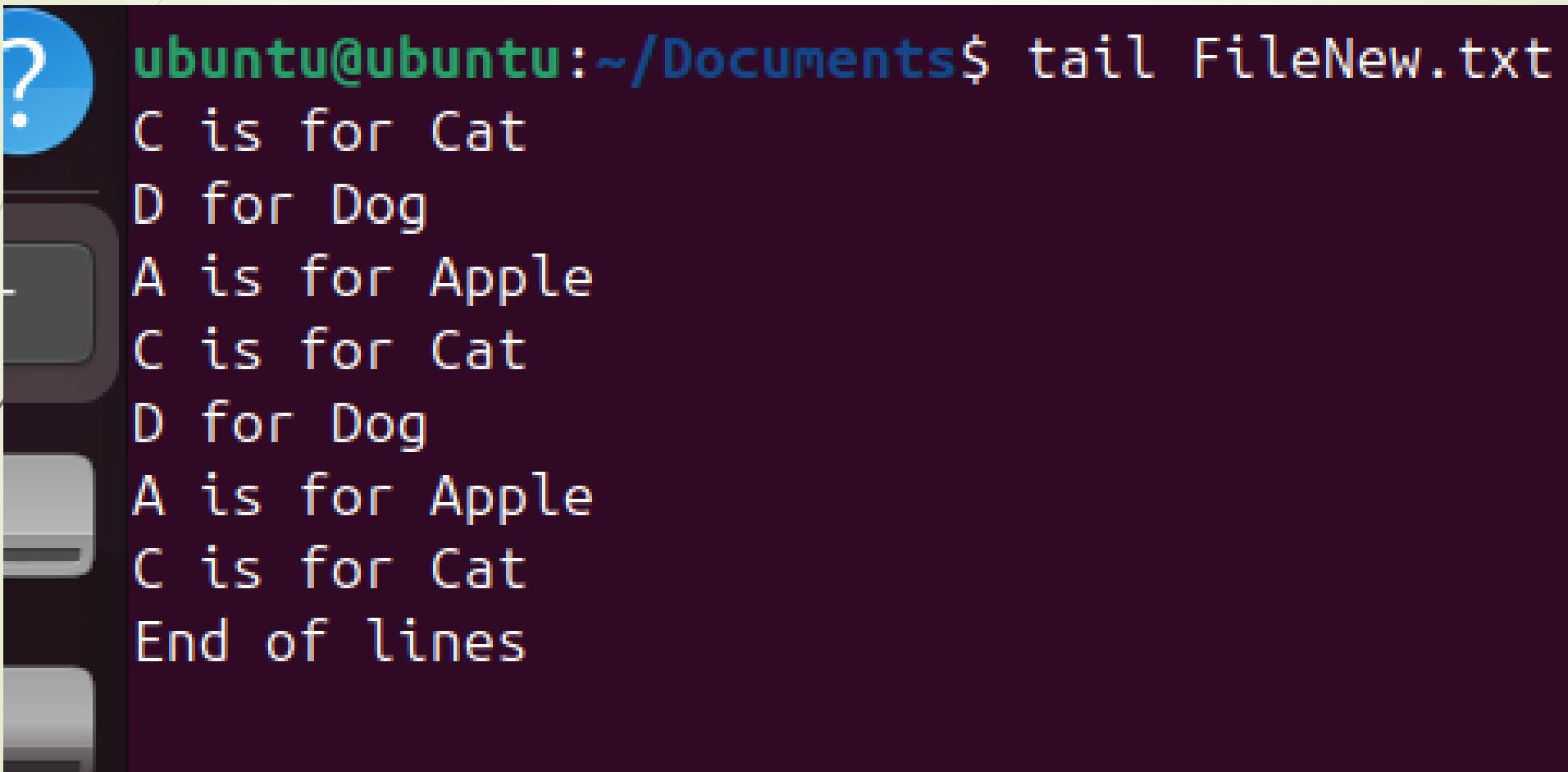
Show the beginning or top of a file. So, if you want to see the first few lines of a file, type.



```
ubuntu@ubuntu:~/Documents$ head FileNew.txt
A is for Apple
C is for Cat
D for Dog
A is for Apple
C is for Cat
D for Dog
A is for Apple
C is for Cat
D for Dog
This a lesson for linux files
ubuntu@ubuntu:~/Documents$
```



To see the ending lines or bottom of a text file,  
type.



```
ubuntu@ubuntu:~/Documents$ tail FileNew.txt
C is for Cat
D for Dog
A is for Apple
C is for Cat
D for Dog
A is for Apple
C is for Cat
End of lines
```

# Using more command

- ▶ The **more** command is useful for viewing large files. It displays the file one screen at a time. When you reach the end of the screen, you can press the **Space** key to view the next screen or **q** to quit.

```
ubuntu@ubuntu:~/Documents$ more FileNew.txt
A is for Apple
C is for Cat
D for Dog
A is for Apple
C is for Cat
D for Dog
A is for Apple
C is for Cat
D for Dog
This a lesson for linux files
C is for Cat
D for Dog
A is for Apple
C is for Cat
D for Dog
A is for Apple
C is for Cat
D for degree
A is for Apple
C is for Cat
D for Dog
A is for Apple
C is for Cat
D for Dog
A is for Apple
C is for Cat
End of lines
```

# Using less command

```
ubuntu@ubuntu:~/Documents$ less FileNew.txt  
ubuntu@ubuntu:~/Documents$
```

Some useful commands while using `less`:

- `j` and `k`: Move down and up one line respectively.
- `n`: Go to the next occurrence of the pattern.
- `N`: Go to the previous occurrence of the pattern.
- `q`: Quit `less`.

	A is for Apple
	C is for Cat
	D for Dog
	A is for Apple
	C is for Cat
	D for Dog
	A is for Apple
	C is for Cat
	D for Dog
	This a lesson for linux files
	C is for Cat
	D for Dog
	A is for Apple
	C is for Cat
	D for Dog
	A is for Apple
	C is for Cat
	D for Dog
	A is for Apple
	C is for Cat
	D for Dog
	A is for Apple
	C is for Cat
	D for Dog
	A is for Apple
	C is for Cat
	D for Dog
	A is for Apple
	C is for Cat
	D for Dog
	A is for Apple
	C is for Cat
	End of lines
	~
	(END)



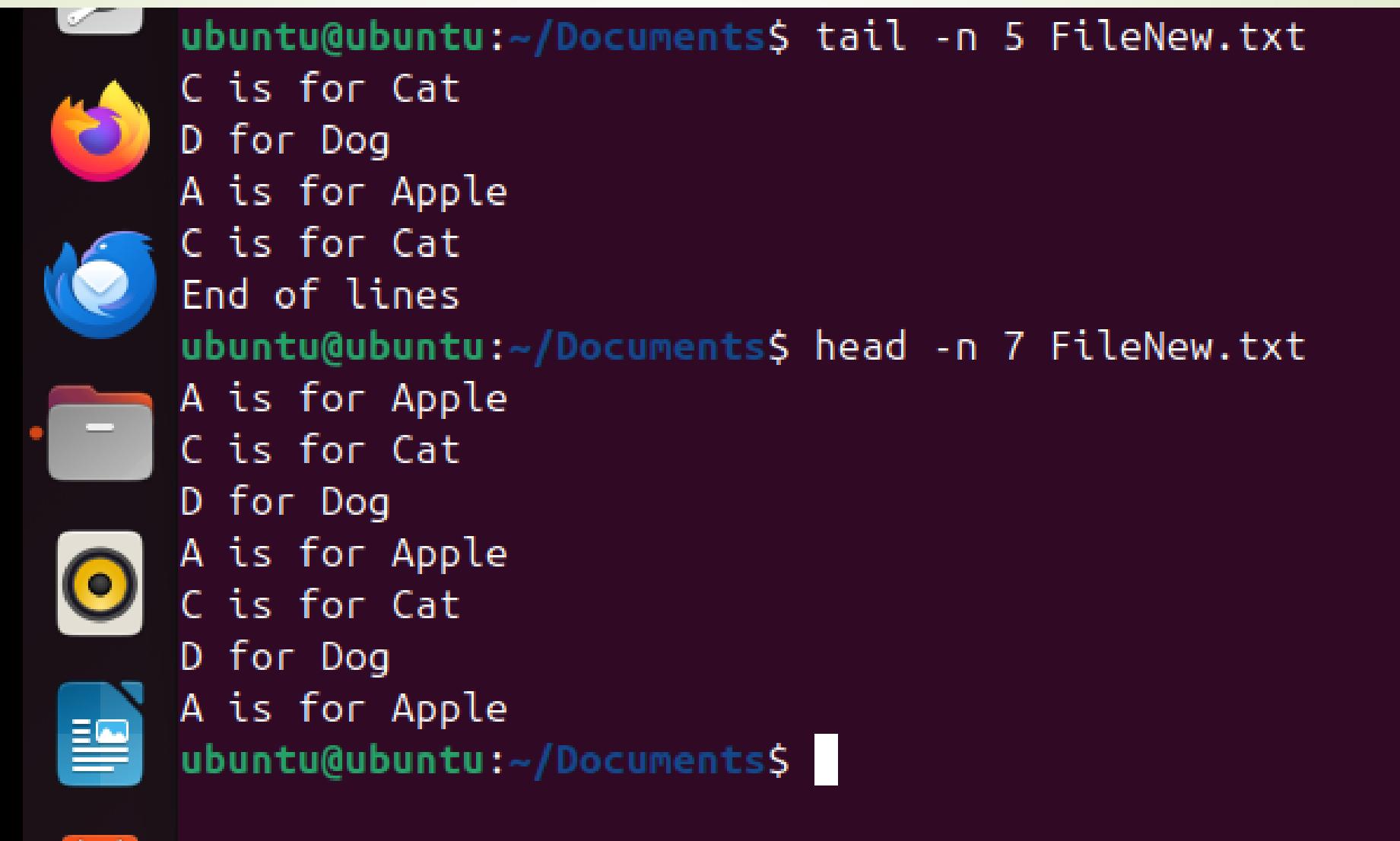
► The **-n** option with the head and tail commands can be used to specify the number of lines to display. To use the -n option, specify the number of lines from the file you want to display after the option and use the filename as an argument:

```
head -n number_of_lines filename
```

# Display the last 5 lines of a file

```
ubuntu@ubuntu:~/Documents$ tail -n 5 FileNew.txt
C is for Cat
D for Dog
A is for Apple
C is for Cat
End of lines
ubuntu@ubuntu:~/Documents$
```

# Display the first 7 lines of a file

A screenshot of an Ubuntu desktop environment. On the left, there's a dock with several icons: a white document, a red Firefox icon, a blue Mail icon, a grey folder icon with a red dot, a yellow speaker icon, and a blue file icon. In the center, a terminal window is open with the following command and output:

```
ubuntu@ubuntu:~/Documents$ tail -n 5 FileNew.txt
C is for Cat
D for Dog
A is for Apple
C is for Cat
End of lines
ubuntu@ubuntu:~/Documents$ head -n 7 FileNew.txt
A is for Apple
C is for Cat
D for Dog
A is for Apple
C is for Cat
D for Dog
A is for Apple
ubuntu@ubuntu:~/Documents$
```

The terminal window has a dark background with light-colored text. The cursor is visible at the end of the last line.

# **echo command in Linux**

- ▶ The **echo** command in Linux is a built-in command that allows users to display lines of text or strings that are passed as arguments. It is commonly used in shell scripts and batch files to output status text to the screen or a file.
- ▶ The most straightforward usage of the echo command is to display a text or string on the terminal. To do this, you simply provide the desired text or string as an argument to the echo command.

# Syntax of `echo` command in Linux

```
echo [option] [string]
```

Here,

[options] = The various options available for modifying the behavior of the 'echo' command

[string] = It is the string that we want to display.

```
ubuntu@ubuntu:~/Documents$ echo "Hello, this a lesson for Linux"  
Hello, this a lesson for Linux  
ubuntu@ubuntu:~/Documents$
```



# **Options Available in `echo` command in Linux**

- ▶ -e here enables the interpretation of backslash escapes.
- ▶ \b : it removes all the spaces in between the text



A screenshot of a terminal window titled "raghvendra@raghvendra-Inspiron-15-3567: ~". The window has standard Linux-style window controls (minimize, maximize, close) in the top right corner. The menu bar at the top includes "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal itself shows the command "echo -e "Geeks \bfor \bGeeks"" being run, followed by its output "GeeksforGeeks". The terminal has a dark background with light-colored text and a dark purple bottom bar.

```
raghvendra@raghvendra-Inspiron-15-3567:~$ echo -e "Geeks \bfor \bGeeks"
GeeksforGeeks
raghvendra@raghvendra-Inspiron-15-3567:~$
```



**\c** : suppress trailing new line with backspace  
interpreter '-e' to continue without emitting new line.



```
ubuntu@ubuntu:~/Documents$ echo -e "This Lesson \c for Linux"  
This Lesson ubuntu@ubuntu:~/Documents$ █
```

In the above example, text after **\c** is not printed and omitted trailing new line.

\n : this option creates a new line from where it is used.

```
ubuntu@ubuntu:~/Documents$ echo -e "Hello\n This Lesson for Linux "
Hello
 This Lesson for Linux
ubuntu@ubuntu:~/Documents$
```

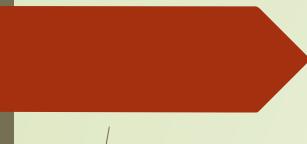
```
raghvendra@raghvendra-Inspiron-15-3567: ~
File Edit View Search Terminal Help
raghvendra@raghvendra-Inspiron-15-3567:~$ echo -e "Geeks \nfor \nGeeks"
Geeks
for
Geeks
raghvendra@raghvendra-Inspiron-15-3567:~$
```

→ \t : this option is used to create horizontal tab spaces.

```
ubuntu@ubuntu:~/Documents$ echo -e "hi\tBoys\tHow are you?"  
hi      Boys      How are you?  
ubuntu@ubuntu:~/Documents$
```

text before \r is not printed.

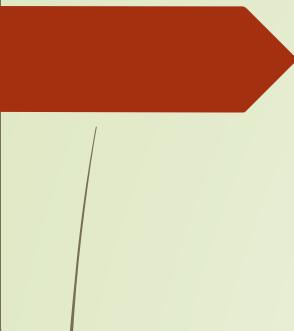
```
ubuntu@ubuntu:~/Documents$ echo -e "Hello\rGood Morning"  
Good Morning  
ubuntu@ubuntu:~/Documents$
```



\v : this option is used to create vertical tab spaces.

```
raghvendra@raghvendra-Inspiron-15-3567: ~
File Edit View Search Terminal Help
raghvendra@raghvendra-Inspiron-15-3567:~$ echo -e "Geeks \vfor \vGeeks"
Geeks
    for
        Geeks
raghvendra@raghvendra-Inspiron-15-3567:~$
```

\a : alert return with backspace  
interpreter ‘-e’ to have sound alert.



```
raghvendra@raghvendra-Inspiron-15-3567: ~
File Edit View Search Terminal Help
raghvendra@raghvendra-Inspiron-15-3567:~$ echo -e "\aGeeks for Geeks"
Geeks for Geeks
raghvendra@raghvendra-Inspiron-15-3567:~$
```

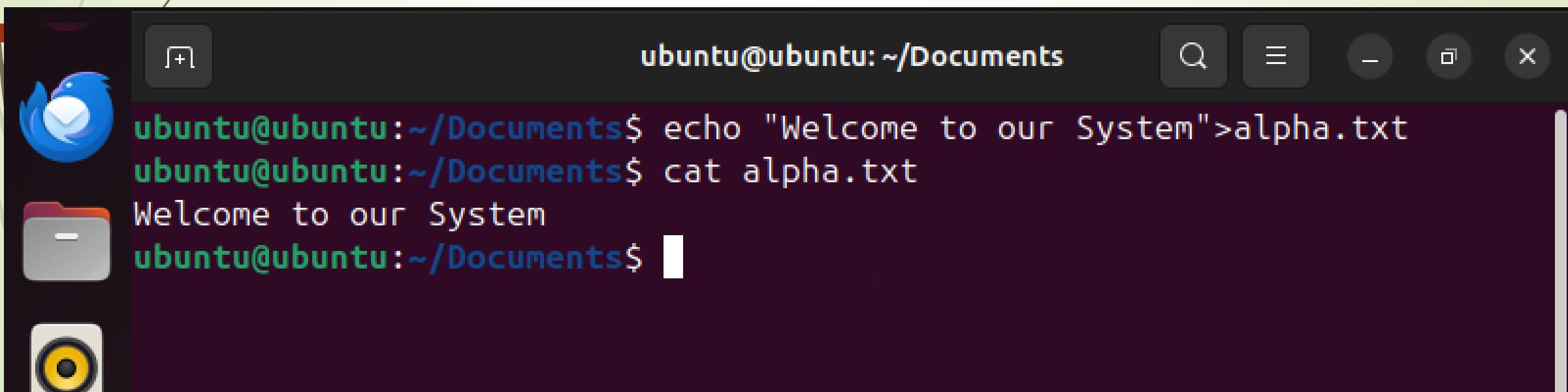


The output of the `echo` can be redirected to a file instead of displaying it on the terminal. We can achieve this by using the `>` or `>>` operators for output redirection. “>” and “>>” both of these operators represent **output redirection** in Linux.

Operators are characters that offer various functionalities. And these redirection operators **redirect** the result or the output.

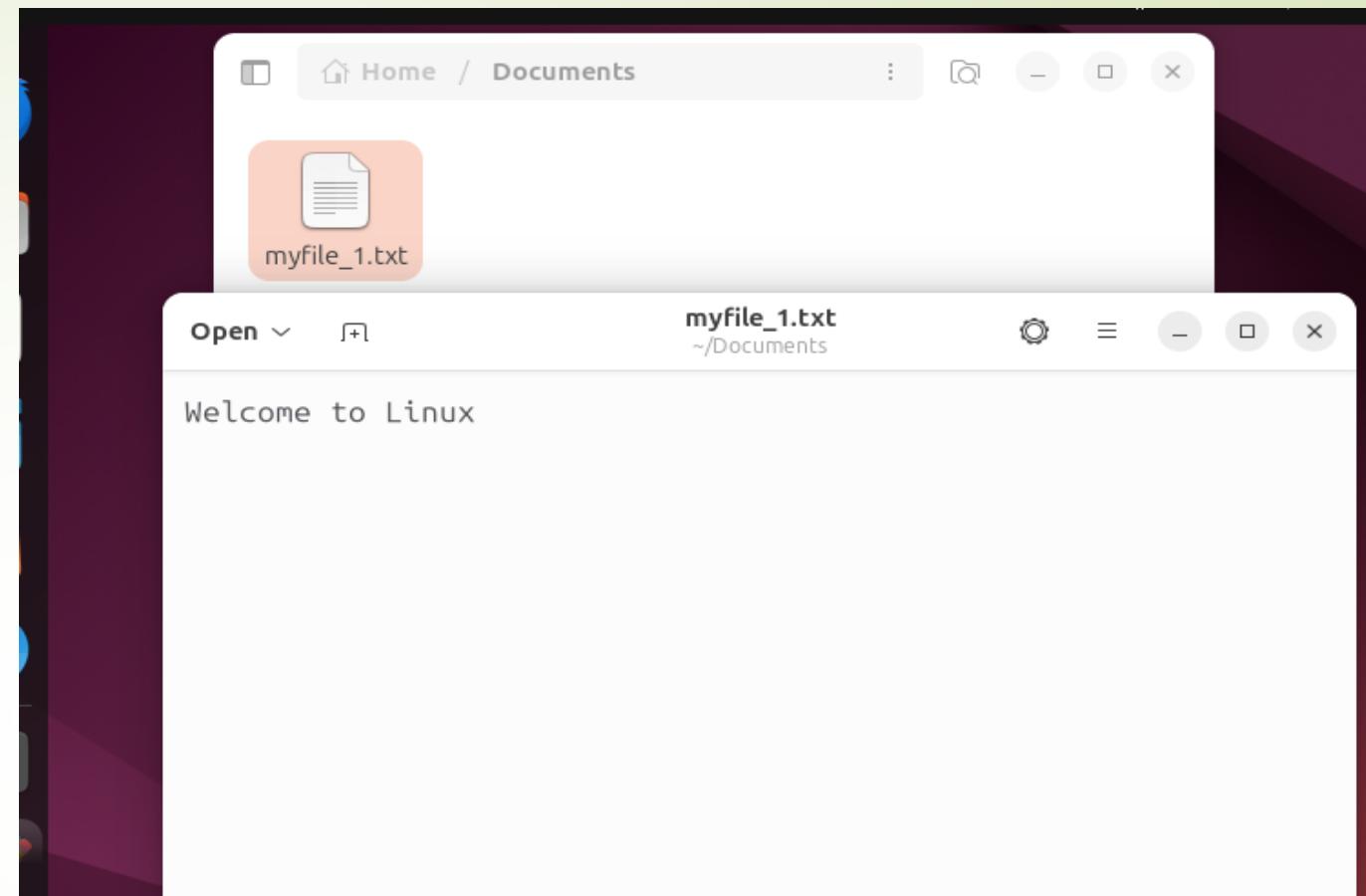
# The ">" Operator

- ">" overwrites an already **existing file** or a **new file** is created providing the mentioned file name isn't there in the directory. This means that while making changes in a file you need to **overwrite** certain any existing data, use the ">" operator.

A screenshot of a Linux desktop environment, specifically Ubuntu, showing a terminal window. The terminal window has a dark theme with a title bar containing the session name "ubuntu@ubuntu: ~/Documents" and window control buttons. The main area of the terminal shows command-line input and output. The user runs the command "echo "Welcome to our System">alpha.txt" followed by "cat alpha.txt" to verify the contents of the newly created file.

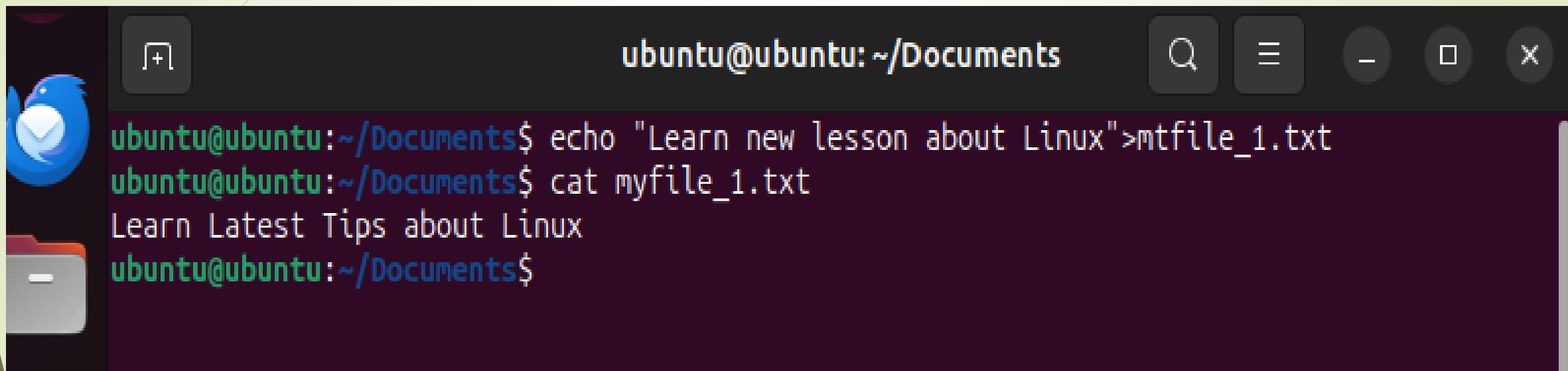
```
ubuntu@ubuntu:~/Documents$ echo "Welcome to our System">alpha.txt
ubuntu@ubuntu:~/Documents$ cat alpha.txt
Welcome to our System
ubuntu@ubuntu:~/Documents$
```

After executing the above command, you'll find that a text file “myfile\_1.txt” is created in the directory. It'll contain the text “Welcome to Linux” in it.

A screenshot of a terminal window titled "ubuntu@ubuntu: ~/Documents". The terminal shows the following commands and output:

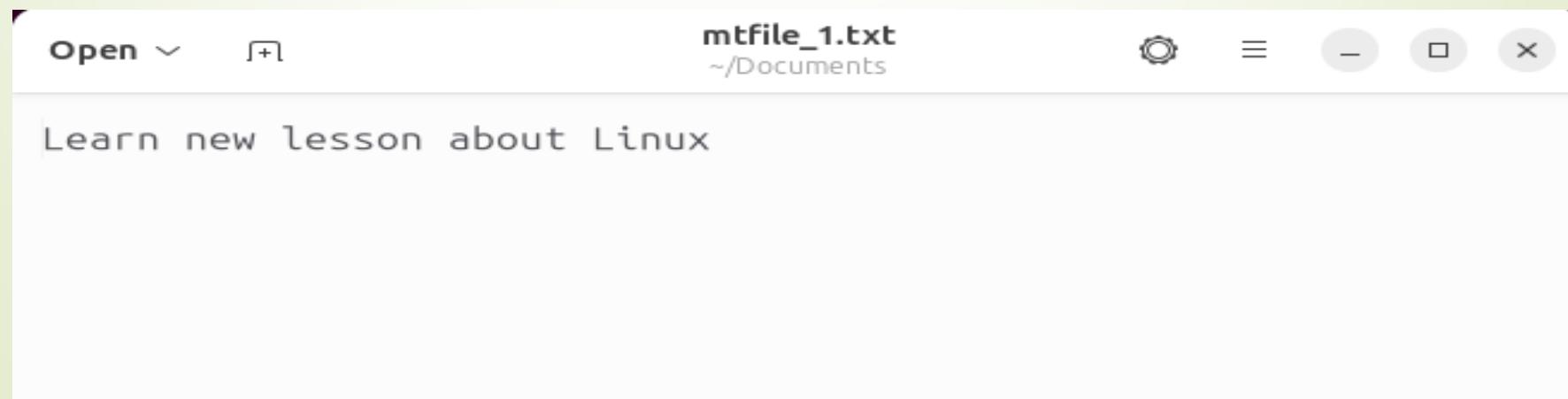
```
ubuntu@ubuntu:~/Documents$ echo "Welcome to Linux">>myfile_1.txt
ubuntu@ubuntu:~/Documents$ cat myfile_1.txt
Welcome to Linux
ubuntu@ubuntu:~/Documents$
```

Now, let's execute the same command with a different text. You'll see that the new text has successfully **overwritten** the earlier text.



A screenshot of a Linux desktop environment, likely Ubuntu. On the left, there's a dock with icons for the Dash (blue bird), Home (orange folder), and Dash (grey folder). In the center, a terminal window is open with the title bar showing "ubuntu@ubuntu: ~/Documents". The terminal displays the following commands and output:

```
ubuntu@ubuntu:~/Documents$ echo "Learn new lesson about Linux">myfile_1.txt
ubuntu@ubuntu:~/Documents$ cat myfile_1.txt
Learn Latest Tips about Linux
ubuntu@ubuntu:~/Documents$
```



A screenshot of a text editor window titled "myfile\_1.txt" located in the "/Documents" directory. The window shows the single line of text "Learn new lesson about Linux". The interface includes standard window controls (minimize, maximize, close) and a toolbar with an "Open" button and a folder icon.

# The “>>” Operator

- ▶ “>>” operator appends an already present file or creates a new file if that file name doesn’t exist in the directory.
- ▶ The above command will create a file by the name “**my\_file\_2.txt**” in your current directory.



```
ubuntu@ubuntu:~/Documents$ echo "Linux Essentials ">> myfile_2.txt
ubuntu@ubuntu:~/Documents$ cat myfile_2.txt
Linux Essentials
ubuntu@ubuntu:~/Documents$ ls -l myfile_2.txt
-rw-rw-r-- 1 ubuntu ubuntu 18 Oct 18 14:54 myfile_2.txt
ubuntu@ubuntu:~/Documents$
```

Open  

myfile\_2.txt  
~/Documents



Linux Essentials



Let's alter the text, now, into, And you'll see instead of **overwriting** the previously entered text, the "**>>**" operator has **appended** the text.

```
ubuntu@ubuntu:~/Documents$ echo "Linux Essentials ">> myfile_2.txt
ubuntu@ubuntu:~/Documents$ cat myfile_2.txt
Linux Essentials
ubuntu@ubuntu:~/Documents$ ls -l myfile_2.txt
-rw-rw-r-- 1 ubuntu ubuntu 18 Oct 18 14:54 myfile_2.txt
ubuntu@ubuntu:~/Documents$ echo "Good Lesson ">>myfile_2.txt
ubuntu@ubuntu:~/Documents$ cat myfile_2.txt
Linux Essentials
Good Lesson
ubuntu@ubuntu:~/Documents$ █
```

Open ▾



myfile\_2.txt  
~/Documents



Linux Essentials

Good Lesson |

So, what we learned is, the “>” is the output redirection operator used for overwriting files that already exist in the directory. While the “>>” is an output operator as well, but it appends the data of an existing file. Often, both of these operators are used together to **modify** files in Linux.

# uname command in Linux

- Let's start with the basics. The term "uname" stands for "Unix Name," and the command itself is designed to provide you with key details about your Linux system. It's like asking your computer, "Hey, who are you, and what are you made of?" The answers you get can help you understand your system's kernel version, operating system, hardware architecture, and more. This command '**uname**' displays the information about the system.

## Syntax:

The basic syntax of 'uname' command is as follows:

```
uname [OPTIONS]
```

# Options and Examples of 'uname' Command in Linux

Options	Description	'-m' or '--machine'	Show the machine hardware name.
'-a' or '--all'	Displays all available information.		
'-s' or '--kernel-name'	Shows the kernel name.	'-p' or '--processor'	Displays the processor type or "unknown."
'-n' or '--nodename'	Displays the network (domain) name of the machine.	'-i' or '--hardware-platform'	Shows the hardware platform or "unknown."
'-r' or '--kernel-release'	Shows the kernel release.	'-o' or '--operating-system'	Displays the operating system.
'-v' or '--kernel-version'	Displays the kernel version.		

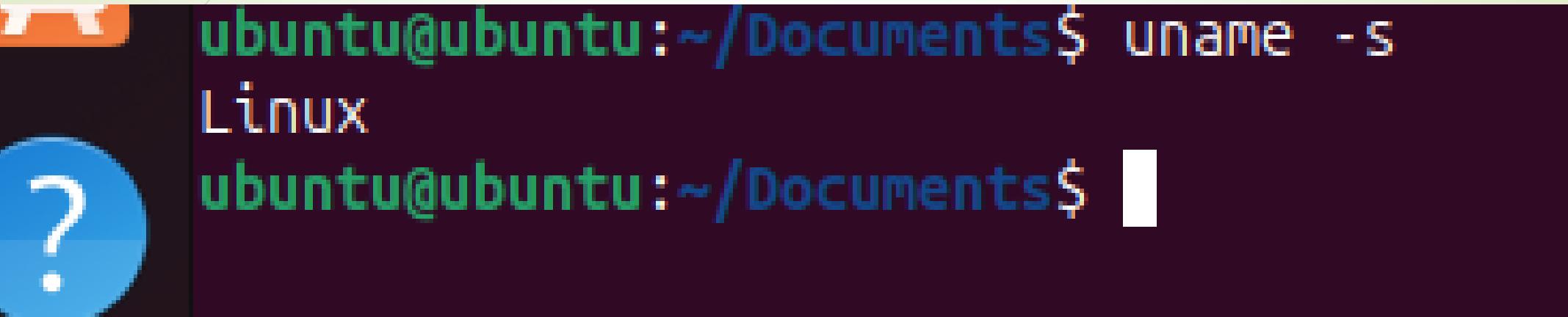
## '-a' option in uname Command in Linux

```
ubuntu@ubuntu:~/Documents$ uname -a
Linux ubuntu 6.14.0-27-generic #27~24.04.1-Ubuntu SMP PREEMPT_DYNAMIC Tue Jul 22 17:38:4
9 UTC 2 x86_64 x86_64 x86_64 GNU/Linux
ubuntu@ubuntu:~/Documents$
```

It prints all the system information in the following order:

Kernel name, network node hostname, kernel release date, kernel version, machine hardware name, hardware platform, operating system.

# '-s' option in uname Command in Linux



```
ubuntu@ubuntu:~/Documents$ uname -s  
Linux  
ubuntu@ubuntu:~/Documents$
```

It prints the kernel name.

# '-n' option in uname Command in Linux

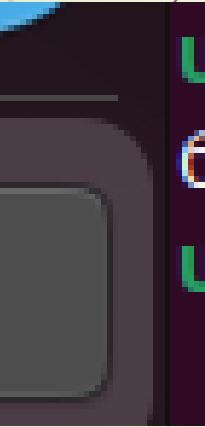
- It prints the hostname of the network node(current computer).
- 



```
ubuntu@ubuntu:~/Documents$ uname -n  
ubuntu  
ubuntu@ubuntu:~/Documents$ █
```

# '-r' option in uname Command in Linux

- ▶ It prints the kernel release date. This is useful for debugging or understanding the exact environment you're working in.



```
ubuntu@ubuntu:~/Documents$ uname -r
6.14.0-27-generic
ubuntu@ubuntu:~/Documents$
```

# '-v' option in uname Command in Linux

- It prints the version of the current kernel, including the build date and time.

```
ubuntu@ubuntu:~/Documents$ uname -v  
#27~24.04.1-Ubuntu SMP PREEMPT_DYNAMIC Tue Jul 22 17:38:49 UTC 2  
ubuntu@ubuntu:~/Documents$
```

## '-m' option in uname Command in Linux

- It prints the machine hardware name of the machine, such as '**x86\_64**' for a 64-bit system.

```
ubuntu@ubuntu:~/Documents$ uname -m  
x86_64  
ubuntu@ubuntu:~/Documents$
```

# '-o' option in uname Command in Linux

- It prints the name of the operating system running on the machine.



```
ubuntu@ubuntu:~/Documents$ uname -o
GNU/Linux
ubuntu@ubuntu:~/Documents$
```

# References

- ▶ <https://www.howtogeek.com/ways-to-view-or-open-a-file-in-the-linux-terminal/>
- ▶ <https://linuxvox.com/blog/view-files-in-linux/#fundamental-concepts>
- ▶ <https://www.geeksforgeeks.org/linux-unix/echo-command-in-linux-with-examples/>
- ▶ <https://www.shells.com/l/en-US/tutorial/Difference-between-%E2%80%9C%E2%80%9D-and-%E2%80%9C%E2%80%9D-in-Linux>
- ▶ Ramses van Zon," Securing File Access Permissions on Linux ", SciNet HPC, University of Toronto ,27 October 2022.
- ▶ <https://www.geeksforgeeks.org/linux-unix/uname-command-in-linux-with-examples/>