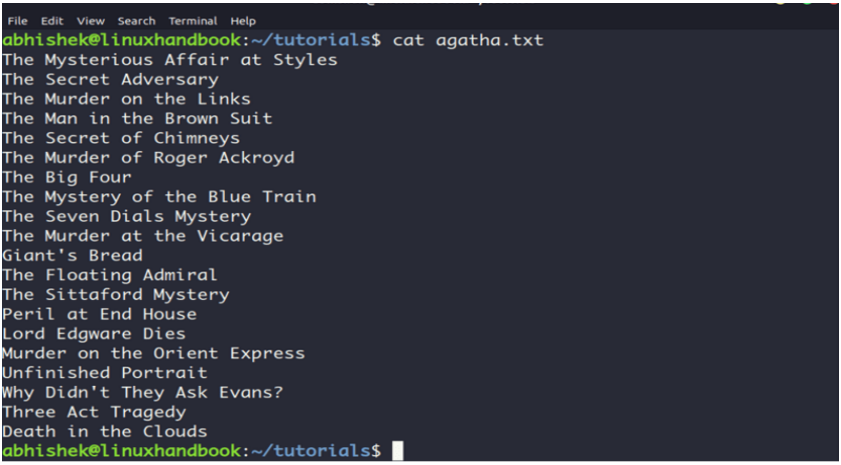
hitbullseye

File Display Commands: (cat, less, more, head, tail) Redirection

A file opened in Notepad and read in a Linux terminal are not the same thing. You should use commands to read files in Linux as you are in command-line mode. A file can be displayed quite easily in Linux. It is simple and crucial that you understand how to read line files. Following commands will enable you to view a file's content in a Linux terminal.

1. Cat

This is the simplest and perhaps the most popular command to view a file in Linux. Cat simply prints the content of the file to standard display i.e. your screen. It cannot be simpler than this, can it?

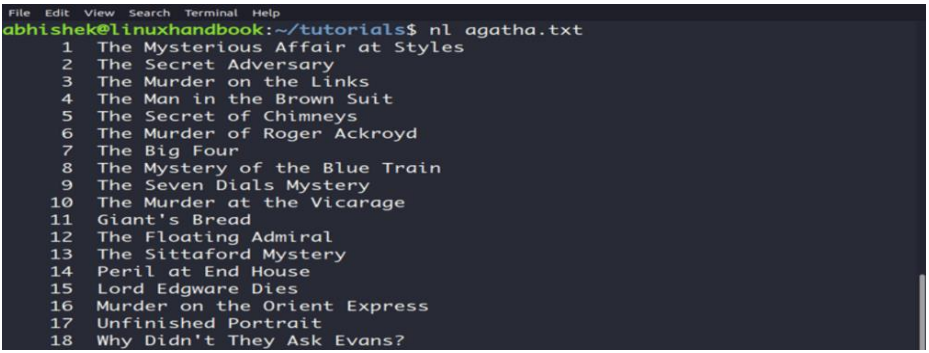


cat displays the content of the file on the screen

Cat becomes a powerful command when used with its options.

2. nl

The nl command is almost like the cat command. The only difference is that it prepends line numbers while displaying the text in the terminal.



nl command displays text with line numbers

There are a few options with nl command that allows you to control the numbering.

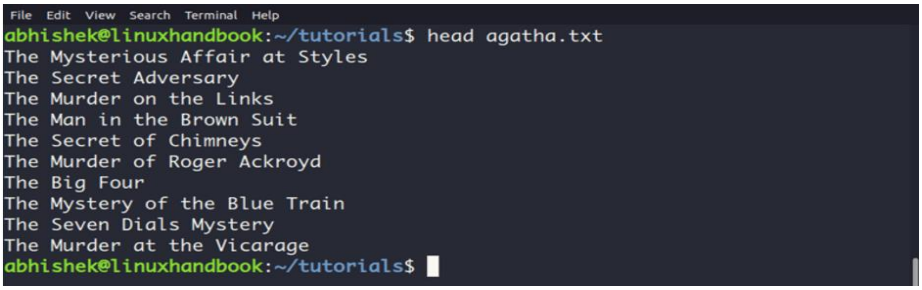
3. Less

Less command views the file one page at a time. The best thing is that you exit less (by pressing q), there are no lines displayed on the screen. Your terminal remains clean and pristine.

I strongly recommend learning a few options of the Less command so that you can use it more effectively. There is also more command which was used in olden days but less command has more friendly features. This is why you might come across the humorous term 'less is more'.

4. Head

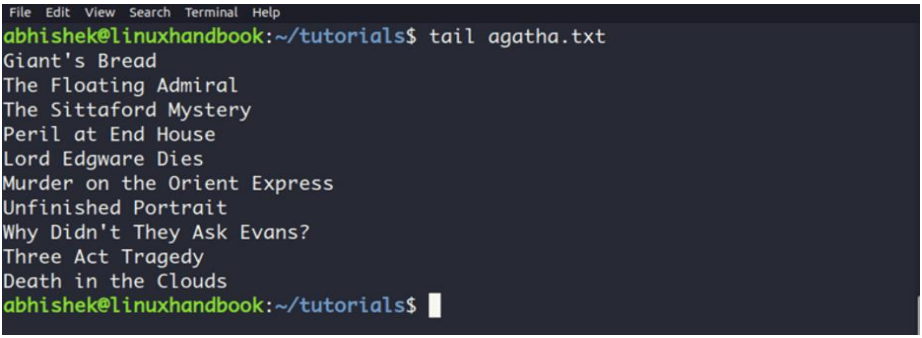
Head command is another way of viewing text file but with a slight difference. The head command displays the first 10 lines of a text file by default.



You can change this behavior by using options with head command but the fundamental principle remains the same: head command starts operating from the head (beginning) of the file.

5. Tail

Tail command in Linux is similar and yet opposite to the head command. While head command displays file from the beginning, the tail command displays file from the end. By default, tail command displays the last 10 lines of a file.



Head and Tail commands can be combined to display selected lines from a file. You can also use tail command to see the changes made to a file in real time.

6. More Command:

The **more** command displays the file called *name* in the screen. As 'cat' command displays the file content. Same way 'more' command also displays the content of a file.

Only difference is that, in case of larger files, 'cat' command output will scroll off your screen while 'more' command displays output one screenful at a time. Following keys are used in 'more' command to scroll the page:

- ✓ Enter key: To scroll down page line by line.
- ✓ Space bar: To go to next page.
- ✓ b key: To go to the backward page.
- ✓ / key: Lets you search the string.

Syntax: more **<file** name**>**

Example: more /var/log/udev

```
sssit@JavaTpoint:~$ more /var/log/udev
monitor will print the received events for:
UDEV - the event which udev sends out after rule processing
KERNEL - the kernel uevent

KERNEL[8.308288] add      /devices/LNXSYSTM:00 (acpi)
ACTION=add
DEVPATH=/devices/LNXSYSTM:00
MODALIAS=acpi:LNXSYSTM:
SEQNUM=1373
SUBSYSTEM=acpi
UDEV_LOG=3

KERNEL[8.308302] add      /devices/LNXSYSTM:00/LNXCPU:00 (acpi)
ACTION=add
DEVPATH=/devices/LNXSYSTM:00/LNXCPU:00
DRIVER=processor
MODALIAS=acpi:LNXCPU:
--More--(0%)
```

Look at the above snapshot, in the left corner it shows 0%, which indicates that 0% page is displayed. To scroll down use **'space'** button, next page will be displayed. If you want

to scroll down the page line by line use **'enter'** key. If you want to go to the last or backward page use **'b'** key.

Linux more options

Options	Function
more -num	Limits the line displayed per page.
more -d	Displays user message at right corner.
more -s	Squeeze blank lines.
more +/string name	It helps to find the string.
more +num	Used to display the content from a specific line.

Note: The 'more' command can't be used to display binary files.

Input/Output Redirections:

We will go into great detail regarding the input/output redirections in Shell. The vast majority of Linux system commands accept input from your terminal and output it back to your terminal. In most cases, a command reads its input from the standard input, which is by default your terminal. In a similar vein, a command typically outputs to standard output, which is again by default your terminal.

Output Redirection

The output from a command normally intended for standard output can be easily diverted to a file instead. This capability is known as output redirection.

If the notation > file is appended to any command that normally writes its output to standard output, the output of that command will be written to file instead of your terminal. Check the following **who** command which redirects the complete output of the command in the users file.

```
$ who > users
```

Notice that no output appears at the terminal. This is because the output has been redirected from the default standard output device (the terminal) into the specified file. You can check the users file for the complete content−

```
$ cat users
oko     tty01     Sep 16     07:30
ai      tty15     Sep 16     13:32
ruth    tty21     Sep 16     10:10
pat     tty24     Sep 16     13:07
steve   tty25     Sep 16     13:03
$
```

If a command has its output redirected to a file and the file already contains some data, that data will be lost. Consider the following example −

```
$ echo line 1 > users
$ cat users
line 1
$
```

You can use >> operator to append the output in an existing file as follows −

```
$ echo line 2 >> users
$ cat users
line 1
line 2
$
```

Input Redirection

Just as the output of a command can be redirected to a file, so can the input of a command be redirected from a file. As the **greater-than character >** is used for output redirection, the **less-than character <** is used to redirect the input of a command.

The commands that normally take their input from the standard input can have their input redirected from a file in this manner. For example, to count the number of lines

in the file *users* generated above, you can execute the command as follows −

```
$ wc -l users
2 users
$
```

Upon execution, you will receive the following output. You can count the number of lines in the file by redirecting the standard input of the **wc** command from the file *users* −

```
$ wc -l < users
2
$
```

Note that there is a difference in the output produced by the two forms of the wc command. In the first case, the name of the file users is listed with the line count; in the second case, it is not.

In the first case, wc knows that it is reading its input from the file users. In the second case, it only knows that it is reading its input from standard input so it does not display file name.