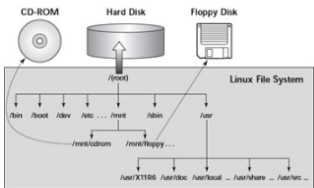


Files

A simple description of the UNIX system, also applicable to Linux, is this: "On a UNIX system, everything is a file; if something is not a file, it is a process."



A Linux system, just like UNIX, makes no difference between a file and a directory, since a directory is just a file containing names of other files. Programs, services, texts, images, and so forth, are all files. Input and output devices, and generally all devices, are considered to be files, according to the system.

A directory is a special kind of file, but it is still a (case sensitive!) file. Each terminal window (for example /dev/pts/4), any hard disk or partition (for example /dev/sdb1) and any process are all represented somewhere in the file system as a file. It will become clear throughout this course that everything on Linux is a file.

The tree of the file system starts at the trunk or slash, indicated by a forward slash (/). This directory, containing all underlying directories and files, is also called the root directory or "the root" of the file system.

- This table gives an overview of the characters determining the file type:


Symbol	Meaning
-	Regular file
d	Directory
l	Link
c	Special file
s	socket
p	Named pipe
b	Block device

Viewing file properties

ls command Displays a list of files in the current working directory. Besides the name of the file, **ls** can give a lot of other information, such as the file type. It can also show *permissions* on a file, *file size*, *inode number*, *creation date and time*, *owners* and *amount of links* to the file. With the **-la** option to **ls**, files that are normally hidden from view can be displayed as well. These are files that have a name starting with a *dot*.



- Directories are denoted in blue color.
- Files are denoted in white
- You will find similar color schemes in different flavors of Linux

 **Tips:**

On most Linux versions **ls** is aliased to **color-ls** by default. This feature allows to see the file type without using any options to **ls**. To achieve this, every file type has its own color. The standard scheme is in **/etc/DIR_COLORS**:

Color_File Type

- blue_directories
- red_compressed archives
- white_text files
- pink_images
- cyan_links
- yellow_devices
- green_executables
- flashing red_broken links

Working with File Contents-1

- head

You can use **head** to display the first ten lines of a file.

```
1 username@clarusway:~$ head /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
```

The **head** command can also display the first *n* lines of a file.

```
1 username@clarusway:~$ head -4 /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
```

- tail

Similar to **head**, the **tail** command will display the last ten lines of a file.

```
1 username@clarusway:~$ tail /etc/services
vboxd 20012/udp
binkp 24554/tcp # binkp fidonet protocol
asp 27374/tcp # Address Search Protocol
asp 27374/udp
csync2 30865/tcp # cluster synchronization tool
dirproxy 57000/tcp # Detachable IRC Proxy
tfido 60177/tcp # fidonet EMSI over telnet
fido 60179/tcp # fidonet EMSI over TCP
# Local services
```


You can give **tail** the number of lines you want to see.

```
1 username@clarusway:~$ tail -3 /etc/services
fido 60179/tcp # fidonet EMSI
over TCP
# Local services
```

- cat


You can use **cat** to display a file on the screen.

```
1 username@clarusway:~$ cat /etc/resolv.conf
domain linux-training.be
search linux-training.be
```

 Q: How do you display the contents of a file in Linux terminal?

A: **Cat** is most commonly used to display the contents of one or multiple text files, combine files by appending the contents of one file to the end of another file, and create new files.

- Interview Q&A



Working with File Contents-2

- concatenate

cat is short for concatenate. One of the basic uses of **cat** is to concatenate files into a bigger (or complete) file.

```
1 username@clarusway:~$ echo one > part1
2 username@clarusway:~$ echo two > part2
3 username@clarusway:~$ echo three > part3
4 username@clarusway:~$ cat part1
one
5 username@clarusway:~$ cat part2
two
6 username@clarusway:~$ cat part3
three
7 username@clarusway:~$ cat part1 part2 part3
one
two
three
8 username@clarusway:~$ cat part1 part2 part3 > all
9 username@clarusway:~$ cat all
one
two
three
```

- create files

You can use **cat** to create flat text files. Type the **cat > winter.txt** command as shown in the screenshot below. Then type one or more lines, finishing each line with the enter key. After the last line, type and hold the Control (Ctrl) key and press **d**.

You can use cat to create flat text files. Type the cat > winter.txt command as shown in the screenshot below. Then type one or more lines, finishing each line with the enter key. After the last line, type and hold the Control (Ctrl) key and press d.

```
1 username@clarusway:~$ cat > winter.txt
2 It is very cold today!
3 username@clarusway:~$ cat winter.txt
4 It is very cold today!
```

Tips:

The Ctrl d key combination will send an EOF (End of File) to the running process ending the cat command.

- copy files

You will see that cat can be used to copy files.

```
1 username@clarusway:~$ cat winter.txt
2 It is very cold today!
3 username@clarusway:~$ cat winter.txt > cold.txt
4 username@clarusway:~$ cat cold.txt
5 It is very cold today!
```

Working with File Contents-3

- tac

Just one example will show you the purpose of tac (cat backward).

```
1 username@clarusway:~$ cat count
2 one
3 two
4 three
5 four
6 username@clarusway:~$ tac count
7 four
8 three
9 two
10 one
```

- more

more command is Linux utility which can be used to view (but not modify) the contents of a text file one screen at a time.

```
1 username@clarusway:~$ more myFile
```

- less

Similar to more, less command allows you to view the contents of a file and navigate through the file. The main difference between more and less is that less command is faster because it does not load the entire file at once and allows navigation through file using page up/down keys.

```
1 username@clarusway:~$ less myFile
```

Searching Files (find Command)

find Command

The “find” command allows you to search for files for which you know the approximate filenames. The simplest form of the command searches for files in the current directory and recursively through its subdirectories that match the supplied search criteria.

Tips:

FIND is an command for searching file(s) and folder(s) using filters such as size , access time , modification time.

Typing the following command at the prompt lists all files found in the current directory.

```
1 username@clarusway:~$ find .
```

The dot after “find” means the current directory.

Examples:

```
1 username@clarusway:~$ cd documents
2 username@clarusway:~/documents$ find .
3 .
4 ./file1
5 ./file2
6 ./file3
7 username@clarusway:~/documents$
```

Use the -name argument to find files that fit a specific pattern.

For example, if we want to find all the files that start with “fil” in the /home directory, we type the following command.

```
1 yourname@yourcomp:~$ find /home -name fil\*
```

⚠️ Avoid:

- The find command is case sensitive. If you want to search for a word to be case insensitive, use the -iname option with the find command.

Examples:

```
1 yourname@yourcomp:~$ find . -iname fil\*
2 ./FILE4
3 ./FILE5
4 ./file1
5 ./file2
6 ./file3
7 yourname@yourcomp:~/documents$
```

Searching Files (grep Command)

grep Command

The grep, which stands for “global regular expression print,” is used to search text. It searches the given file for lines containing a match to the given strings or words.

Tips:

GREP :(Globally search a Regular Expression and Print).

The syntax:

```
1 grep 'word' filename
```

This means search any line that contains the word in filename on Linux.

Examples:

```
1 yourname@yourcomp:~$ grep 'understand' quotes.txt
2 I hear and I forget. I see and I remember. I do and I understand.
```

Tips:

The Major difference is FIND is for searching files and directories using filters while GREP is for searching a pattern inside a file or searching process(es)

Command	Description
grep -i	Returns the results for case insensitive strings
grep -n	Returns the matching strings along with their line number
grep -v	Returns the result of lines not matching the search string
grep -c	Returns the number of lines in which the results matched the search string

“ Q: Why the grep command used for?

A: The grep command is used to search text. It searches the given file for lines containing a match to the given strings or words. It is one of the most useful commands on Linux and Unix-like system.

— Interview Q&A

”