4 Files and directories

At the end of this chapter you:

- can copy, move and delete files
- how to get meta information on files
- know safe names for files
- can take hexdumps of files
- can display the content of text files
- can create, move and delete directories
- know the difference between absolute and relative paths
- can navigate a file system
- know the shortcuts to some directories

In this chapter we go through the day-to-day use of a computer. During the process you will come across a number of new commands. Since you learned how to get help from the system in chapter 3 the commands will simply be used without much explanation. It is highly recommended that you repeat those examples and refer the help as we progress. Some options we discuss are subtle, taking notes will help to remember them.

4.1 Handling files

4.1.1 What is a file

Like any other operating system, Linux stores information in chunks called files due to their similarity to ordinary office files. Each file has a name, content and some administrative information like file size, the physical place in the storage medium, various timestamps, the owner, etc. sometimes called the meta information, because they are about the content, not the content itself.

The file system in Linux is organized in such a way that you can maintain your own personal files without interfering with files belonging to others. In Linux jargon it is called your `home`. Every user has a '`home`' of his own. (Remember the HOME environment variable from the assignment last week?)

For the examples below, we have copied two files to your home. The `ls` (list) command lists the names of files:

```
$ ls
knave  queen
```

So, the files are named knave (In cards; a Jack) and queen. Notice that the list is sorted alphabetically.

Hint: We will be manipulating those files. Don't be afraid you might break something. You can always get a fresh copy from *user e00000*. The command for this is '`cp /home/e00000/knave knave`'. The meaning will be discussed below.

It is important to note that you cannot ‘open’ a file without knowing the kind of information in it and how exactly they are coded. For example, if you somehow open a HTML file with a photo viewer, you won’t see anything useful. People use filename extensions like .txt, .c, .html, ... as a convenience, but there is no mechanism to guarantee that the content matches the extension! The file (file type) command will tell you the kind of information in a file. Let’s try it:

```
$ file knave
knave: ASCII text
$ file queen
queen: ASCII text
```

So, both files are simple text files.

To get the time of last modification run ls with the option -l (long). If the time resolution is not good enough use the ls --full-time option:

```
$ ls -l knave
-rw-r--r-- 1 user group 91 Dez 30 22:15 knave
$ ls --full-time knave
-rw-r--r-- 1 user group 111 2014-12-30 22:15:47.787031906 +0100 knav
e
```

You can update the time stamp of a file by “touching” it:

```
$ ls -l knave
-rw-r--r-- 1 user group  91 Dez 30 22:15 knave
$ touch knave
$ ls -l knave
-rw-r--r-- 1 user group  91 Feb 11 22:15 knave
```

Hint: The touch command can be abused to create empty files, because touch creates an empty file if the given file does not exist. Find out the difference between `$ touch Hello Linux` and `$ touch "Hello Linux"`

```
$ touch king
$ ls -l king
-rw-r--r-- 1 user group  0 Feb 11 22:30 king
```

In fact, `ls -l` brings a whole list of meta information. In this chapter we are going to discuss only a few of them. For the sake of completeness, you will find the full list in the next table:

Column no.	Description
1	File type and file access permissions
2	Number of links
3	File owner
4	Group owner
5	File size (in bytes)

6, 7 and 8	Month, day and time of last modification to the file
9	Name of file

The final meta information we are going to look at is where a file is kept in the file system. When the file system was created it organizes the space in blocks and gives an index to each block called **inode**. The inode of the starting block of a file is all what the file system needs to find a file. The `ls -i` lists the inodes of files:

```
$ ls –i
3156545 knave  3156550 queen
```

4.1.2 What's in a file name

So far, we have used filenames without saying what a legal name is. First, in its initial design Linux filenames were limited to 14 characters. In today's Linux systems a file name could be up to 254 characters long, which should be ample, and should be utilized. Second, although you can use almost any character in a filename, common sense says you should avoid non-printable (invisible) characters and characters that have other meanings. We have already seen that the hyphen and double hyphens are used by Linux commands to denote their options. So, if you had a file whose name was `-t`, you would have a tough time listing it with `ls`.

Besides the hyphen as a first character, there are other characters with special meaning. To avoid pitfalls, you would do well to use only the latin characters a-z and A-Z, the digits 0-9, the period '.', the underscore '_' and the hyphen '-'. The hyphen should not be used as the first character. The period, the underscore and the hyphen are conveniently used to divide filenames into chunks, as in *assignment1_perera_sunil.txt* or *draft-2015-02-13.odt*. Finally, don't forget that the case distinctions matter: *Assignment1_Perera_Sunil.txt* is not the same as *assignment1_perera_sunil.txt*! The command line user tends to prefer lowercase - for easy typing.

4.1.3 Copying, moving and deleting files

The `cp` (copy) command duplicates files. Its syntax is `cp source target`. Example:

```
$ cp knave knave2
$ ls
knave  knave2  queen
```

Keep in mind that if the target exists, it will simply be overwritten. If you want to be informed when that happens, use the `-i` (interactive) option.

To move files use the `mv` (move) command:

```
$ mv knave knave2
$ ls
knave2  queen
```

The rm (remove) command deletes files:

```
$ ls
knave knave2  queen
$ rm knave2
$ ls
knave queen
```

Note that the file will be silently removed. If you want add confirmation to it, use the -i option. Also, find out what rm –r, rm –rf and rm * commands are used for.

```
$ ls
knave  knave2  queen
$ cp –i knave knave2
cp: overwrite 'knave2'? y
```

```
If you are interested:

Find out why $ cp -rfva ../foldersource/. ./ is different from
$ cp -rfva ../foldersource/* ./
Note: The former could be used to copy folder+their hidden files.
Tip: $ ls –la does the same job as $ ls –l -a (a for all)
For Enthusiasts: Find out what are dot files in Linux/Unix
```

4.1.4 Content of a file

To look at the content of a file byte-by-byte you take a 'dump'. The original Linux program for this is od (octal dump). It was common those days to work in the octal (base 8) system!

```
$ od knave
0000000 062163 005146 066040 066154 000012
0000011
$ od –h knave
0000000 6473 0a66 6c20 6c6c 000a
0000011
$ od –c knave
0000000   s   d   f  \n       l   l   l  \n
0000011
```

The example above demonstrates that the default behaviour of od could be changed by using options: -h (hexadecimal), -c (ASCII character).

There is another command which is often abused to display the content of text files. It is the cat (concatenate, to join) command, which was originally meant to join two or more files in to one. But you can just make it print the content of files by giving their names as arguments:

```
$ cat queen
The Queen of Hearts,
she made some tarts,
```

Print *knave* too. How do you print the full poem out of them? Hint: `cat file1 file2`

4.2 Handling directories

4.2.1 Absolute paths

When you are logged in to a shell, you are always “in” some directory called current directory or working directory. The pwd (print the name of working directory) command tells you where you are. Immediately after log in, you begin the session in your home directory:

```
$ pwd
/home/e00000
$ echo $HOME
/home/e00000
```

The cd (change directory) command changes the current directory to the directory specified:

```
$ pwd
/home/e00000
$ cd /usr/bin
$ pwd
/usr/bin
```

Paths to directories in the examples above always started with a /, the *‘root’*, which is the beginning of a Linux file system. Such paths are therefore called absolute paths.

Now to come back to your home, you can of course enter `‘cd /home/e00000’`. But that kind of typing is prohibitive. Linux allows many shortcuts. For example, just cd without any arguments takes you to your home.

Another short cut is the hyphen: `‘cd -’` takes you to the previous directory you were in. Try them out!

4.2.2 Relative paths

The cd command also allows relative paths. For example, let’s assume that you are in */usr*. Then change to */usr/bin* it is enough to type cd bin because the directory bin is where you “stand”.

```
$ cd /usr
$ pwd
/usr
$ cd bin
$ pwd
/usr/bin
```

In every directory, except in the root, there is a special directory with the symbol '..' (two dots) which points to the directory above it, the so-called parent directory. For example to go back to */usr* from */usr/bin*, you could do `cd ..` :

```
$ pwd
/usr/bin
$ cd ..
$ pwd
/usr
```

Another shortcut to remember is the '~' (tilde), which denotes your home directory. More interestingly, ~loginname stands for the home directory of the user loginname. Try changing to the home directories of others!

Note for the experts: Don't close your home directories for others yet. That'll take the fun out of the game!

4.2.3 Creating, moving and deleting directories

The `mkdir` (make directory) command is used to create directories:

```
$ mkdir progs
$ ls
knave  progs  queen
$ ls -F
knave  progs/  queen
```

The previous example shows that the command ls alone would not mark directories differently. The option `-F` prints a slash behind directory names.

You can move directories, just like files:

```
$ mkdir progs
$ mv progs /tmp
knave  progs  queen
$ ls -F
knave  progs/  queen
```

The `rmdir` (remove directory) command removes the directory specified as an argument to it:

```
$ ls -F
knave  progs/  queen
$ rmdir progs
OR
$ rmdir progs
rmdir: failed to remove ‘progs’: Directory not empty
$ rmdir -rf progs
$ ls
knave  queen
```

```
Leisurely you can try the following and see what happens.
$ mkdir {Mint,Most,Popular,Linux,Distro}
$ mkdir {Debian,Debian/Ubuntu,Debian/Ubuntu/Mint}
$ mkdir {Abstraction,Abstraction/Inheritance} && touch
Abstraction/Inheritance/{Encapsulation,Polymorphism}
```

4.3 Assignment for the week

The agents of a call centre were maintaining their logs by throwing all their log files in to a single directory. Each agent had his own way of naming files. You have the job of bringing order in to this. Your proposed solution is the hierarchical directory structure: the top directory, called calls, will have subdirectories tech (the tech team), sales (the sales team), etc. Within those team directories the agents will have their own directories, named after their IDs, say anula, saman, ...

Assignment:

- Crete the top directory calls in your home.
- Under calls create the directory tech.
- Inside tech create the directories anula and saman.
- Inside anula create two empty files jan and feb.

Submission: Make a record of all the commands you issued in you practice Linux to the assignment tool in Moodle so that you will have clear record like in the examples in the lecture note. Also add your comments to the right of the commands. Your solution finally looks like this:

```
$ cd              # change to my home
$ mkdir ...       # create ...
$ cd ...          # change to ...
...
```

Summary

You have learned the meaning and the basic usage of the following commands:

- ls (list)
- od (octal dump)

- cat (concatenate)
- cp (copy)
- mv (move)
- rm (remove)
- pwd (print working directory)
- cd (change directory)
- mkdir (mkdir)
- rmdir (remove diectroy)

This handout was derived from the handout prepared by Dr. Visvanath Ratnaweera for the Unix one Course in the year 2017.