

Chapter 7. Working with files

In this chapter we learn how to recognise, create, remove, copy and move files using commands like **file**, **touch**, **rm**, **cp**, **mv** and **rename**.

7.1. All files are case sensitive

Linux is **case sensitive**, this means that **FILE1** is different from **file1**, and **/etc/hosts** is different from **/etc/Hosts** (the latter one does not exist on a typical Linux computer).

This screenshot shows the difference between two files, one with upper case **W**, the other with lower case **w**.

```
paul@laika:~/Linux$ ls
winter.txt  Winter.txt
paul@laika:~/Linux$ cat winter.txt
It is cold.
paul@laika:~/Linux$ cat Winter.txt
It is very cold!
```

7.2. Everything is a file

A **directory** is a special kind of **file**, but it is still a (case sensitive!) **file**. Even a terminal window (**/dev/pts/4**) or a hard disk (**/dev/sdb**) is represented somewhere in the **file system** as a **file**. It will become clear throughout this course that everything on Linux is a **file**.

7.3. file

The **file** utility determines the file type. Linux does not use extensions to determine the file type. Your editor does not care whether a file ends in **.TXT** or **.DOC**. As a system administrator, you should use the **file** command to determine the file type. Here are some examples on a typical Linux system.

```
paul@laika:~$ file pic33.png
pic33.png: PNG image data, 3840 x 1200, 8-bit/color RGBA, non-interlaced
paul@laika:~$ file /etc/passwd
/etc/passwd: ASCII text
paul@laika:~$ file HelloWorld.c
HelloWorld.c: ASCII C program text
```

The `file` command uses a magic file that contains patterns to recognise file types. The magic file is located in `/usr/share/file/magic`. Type **man 5 magic** for more information.

It is interesting to point out **file -s** for special files like those in `/dev` and `/proc`.

```
root@debian6~# file /dev/sda
/dev/sda: block special
root@debian6~# file -s /dev/sda
/dev/sda: x86 boot sector; partition 1: ID=0x83, active, starthead...
root@debian6~# file /proc/cpuinfo
/proc/cpuinfo: empty
root@debian6~# file -s /proc/cpuinfo
/proc/cpuinfo: ASCII C++ program text
```

7.4. touch

One easy way to create a file is with **touch**. (We will see many other ways for creating files later in this book.)

```
paul@laika:~/test$ touch file1

paul@laika:~/test$ touch file2
paul@laika:~/test$ touch file555
paul@laika:~/test$ ls -l
total 0
-rw-r--r-- 1 paul paul 0 2007-01-10 21:40 file1
-rw-r--r-- 1 paul paul 0 2007-01-10 21:40 file2
-rw-r--r-- 1 paul paul 0 2007-01-10 21:40 file555
```

7.4.1. touch -t

Of course, `touch` can do more than just create files. Can you determine what by looking at the next screenshot? If not, check the manual for `touch`.

```
paul@laika:~/test$ touch -t 200505050000 SinkoDeMayo
paul@laika:~/test$ touch -t 130207111630 BigBattle
paul@laika:~/test$ ls -l
total 0
-rw-r--r-- 1 paul paul 0 1302-07-11 16:30 BigBattle
-rw-r--r-- 1 paul paul 0 2005-05-05 00:00 SinkoDeMayo
```

7.5. rm

When you no longer need a file, use **rm** to remove it. Unlike some graphical user interfaces, the command line in general does not have a *waste bin* or *trash can* to recover files. When you use **rm** to remove a file, the file is gone. Therefore, be careful when removing files!

```
paul@laika:~/test$ ls
BigBattle SinkoDeMayo
paul@laika:~/test$ rm BigBattle
paul@laika:~/test$ ls
SinkoDeMayo
```

7.5.1. rm -i

To prevent yourself from accidentally removing a file, you can type **rm -i**.

```
paul@laika:~/Linux$ touch brel.txt
paul@laika:~/Linux$ rm -i brel.txt
rm: remove regular empty file `brel.txt'? y
paul@laika:~/Linux$
```

7.5.2. rm -rf

By default, **rm -r** will not remove non-empty directories. However **rm** accepts several options that will allow you to remove any directory. The **rm -rf** statement is famous because it will erase anything (providing that you have the permissions to do so). When you are logged on as root, be very careful with **rm -rf** (the **f** means **force** and the **r** means **recursive**) since being root implies that permissions don't apply to you. You can literally erase your entire file system by accident.

```
paul@laika:~$ ls test
SinkoDeMayo
paul@laika:~$ rm test
rm: cannot remove `test': Is a directory
paul@laika:~$ rm -rf test
paul@laika:~$ ls test
ls: test: No such file or directory
```

7.6. cp

To copy a file, use **cp** with a source and a target argument. If the target is a directory, then the source files are copied to that target directory.

```
paul@laika:~/test$ touch FileA
paul@laika:~/test$ ls
FileA
paul@laika:~/test$ cp FileA FileB
paul@laika:~/test$ ls
FileA  FileB
paul@laika:~/test$ mkdir MyDir
paul@laika:~/test$ ls
FileA  FileB  MyDir
paul@laika:~/test$ cp FileA MyDir/
paul@laika:~/test$ ls MyDir/
```

7.6.1. cp -r

To copy complete directories, use **cp -r** (the **-r** option forces **recursive** copying of all files in all subdirectories).

```
paul@laika:~/test$ ls
FileA  FileB  MyDir
paul@laika:~/test$ ls MyDir/
FileA
paul@laika:~/test$ cp -r MyDir MyDirB
paul@laika:~/test$ ls
FileA  FileB  MyDir  MyDirB
paul@laika:~/test$ ls MyDirB
FileA
```

7.6.2. cp multiple files to directory

You can also use `cp` to copy multiple files into a directory. In this case, the last argument (a.k.a. the target) must be a directory.

```
cp file1 file2 dir1/file3 dir1/file55 dir2
```

7.6.3. cp -i

To prevent `cp` from overwriting existing files, use the `-i` (for interactive) option.

```
paul@laika:~/test$ cp fire water
paul@laika:~/test$ cp -i fire water
cp: overwrite `water'? no
paul@laika:~/test$
```

7.6.4. cp -p

To preserve permissions and timestamps from source files, use `cp -p`.

```
paul@laika:~/perms$ cp file* cp
paul@laika:~/perms$ cp -p file* cpp
paul@laika:~/perms$ ll *
-rwx----- 1 paul paul    0 2008-08-25 13:26 file33
-rwxr-x--- 1 paul paul    0 2008-08-25 13:26 file42

cp:
total 0
-rwx----- 1 paul paul 0 2008-08-25 13:34 file33
-rwxr-x--- 1 paul paul 0 2008-08-25 13:34 file42

cpp:
total 0
-rwx----- 1 paul paul 0 2008-08-25 13:26 file33
-rwxr-x--- 1 paul paul 0 2008-08-25 13:26 file42
```


7.7. mv

Use **mv** to rename a file or to move the file to another directory.

```
paul@laika:~/test$ touch file100
paul@laika:~/test$ ls
file100
paul@laika:~/test$ mv file100 ABC.txt
paul@laika:~/test$ ls
ABC.txt
paul@laika:~/test$
```

When you need to rename only one file then **mv** is the preferred command to use.

7.8. rename

The **rename** command can also be used but it has a more complex syntax to enable renaming of many files at once. Below are two examples, the first switches all occurrences of txt to png for all file names ending in .txt. The second example switches all occurrences of upper case ABC in lower case abc for all file names ending in .png. The following syntax will work on debian and ubuntu (prior to Ubuntu 7.10).

```
paul@laika:~/test$ ls
123.txt  ABC.txt
paul@laika:~/test$ rename 's/txt/png/' *.txt
paul@laika:~/test$ ls
123.png  ABC.png
paul@laika:~/test$ rename 's/ABC/abc/' *.png
paul@laika:~/test$ ls
123.png  abc.png
paul@laika:~/test$
```

On Red Hat Enterprise Linux (and many other Linux distributions like Ubuntu 8.04), the syntax of **rename** is a bit different. The first example below renames all *.conf files replacing any occurrence of conf with bak. The second example renames all (*) files replacing one with ONE.

```
[paul@RHEL4a test]$ ls
one.conf  two.conf
[paul@RHEL4a test]$ rename conf bak *.conf
[paul@RHEL4a test]$ ls
one.bak  two.bak
[paul@RHEL4a test]$ rename one ONE *
[paul@RHEL4a test]$ ls
ONE.bak  two.bak
[paul@RHEL4a test]$
```