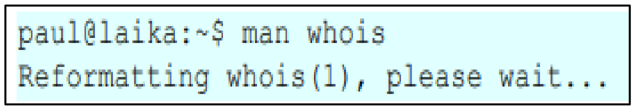
LINUS FUNDAMENTELS
Linus Commands

1) Man pages :-

This chapter will explain the use of man pages (also called manual pages) on your Unix or Linux computer. You will learn the man command together with related commands like whereis, whatis and mandb. Most Unix files and commands have pretty good man pages to explain their use. Man pages also come in handy when you are using multiple flavours of Unix or several Linux distributions since options and parameters sometimes vary.

1.1. man $command

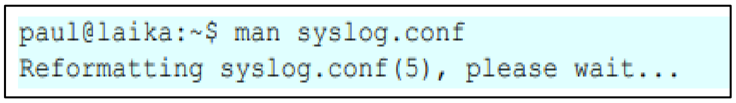
Type man followed by a command (for which you want help) and start reading. Press q to quit the manpage. Some man pages contain examples (near the end).



```
paul@laika:~$ man whois
Reformatting whois(1), please wait...
```

1.2. man $configfile

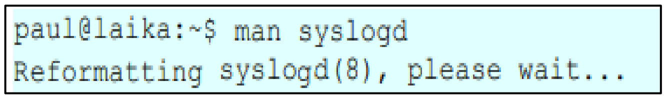
Most configuration files have their own manual.



```
paul@laika:~$ man syslog.conf
Reformatting syslog.conf(5), please wait...
```

1.3. man $daemon

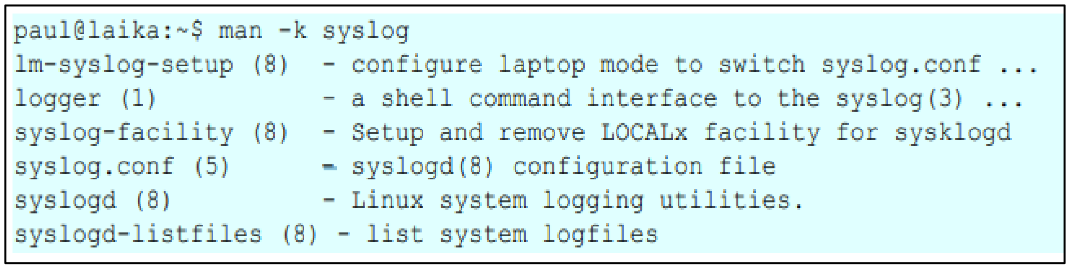
This is also true for most daemons (background programs) on your system.



```
paul@laika:~$ man syslogd
Reformatting syslogd(8), please wait...
```

1.4. man -k (apropos)

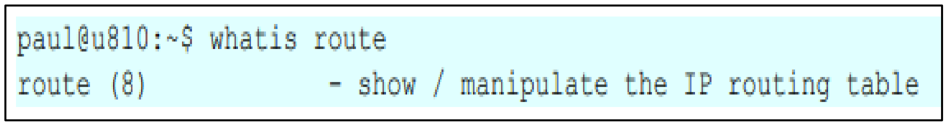
man -k (or apropos) shows a list of man pages containing a string.



```
paul@laika:~$ man -k syslog
lm-syslog-setup (8)  - configure laptop mode to switch syslog.conf ...
logger (1)           - a shell command interface to the syslog(3) ...
syslog-facility (8)  - Setup and remove LOCALx facility for sysklogd
syslog.conf (5)      - syslogd(8) configuration file
syslogd (8)          - Linux system logging utilities.
syslogd-listfiles (8) - list system logfiles
```

1.5. whatis

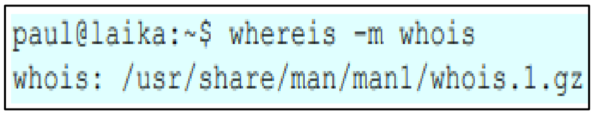
To see just the description of a manual page, use whatis followed by a string.



```
paul@u810:~$ whatis route
route (8)            - show / manipulate the IP routing table
```

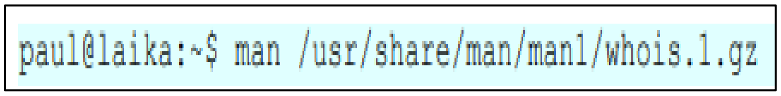
1.6. Whereis

The location of a manpage can be revealed with whereis.



```
paul@laika:~$ whereis -m whois
whois: /usr/share/man/man1/whois.1.gz
```

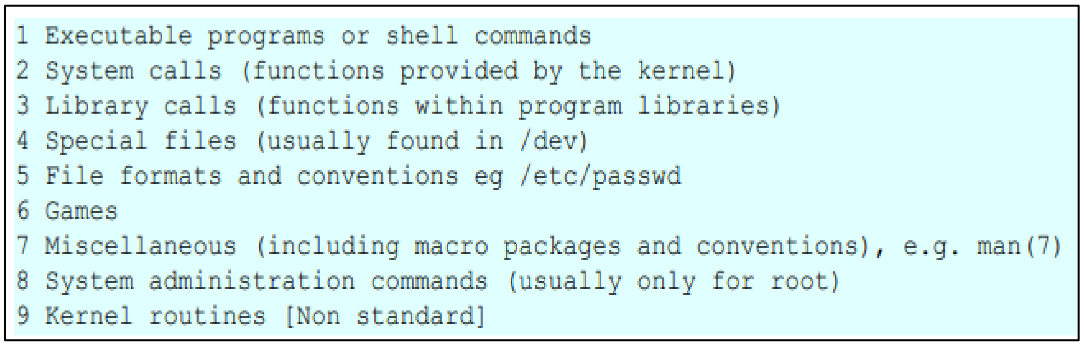
This file is directly readable by man.



```
paul@laika:~$ man /usr/share/man/man1/whois.1.gz
```

1.7. man sections

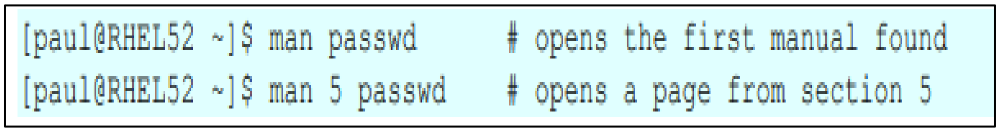
By now you will have noticed the numbers between the round brackets. man man will explain to you that these are section numbers. Executable programs and shell commands reside in section one.



```
1 Executable programs or shell commands
2 System calls (functions provided by the kernel)
3 Library calls (functions within program libraries)
4 Special files (usually found in /dev)
5 File formats and conventions eg /etc/passwd
6 Games
7 Miscellaneous (including macro packages and conventions), e.g. man(7)
8 System administration commands (usually only for root)
9 Kernel routines [Non standard]
```

1.8. man $section $file

Therefore, when referring to the man page of the passwd command, you will see it written as passwd(1); when referring to the passwd file, you will see it written as passwd(5). The screenshot explains how to open the man page in the correct section.

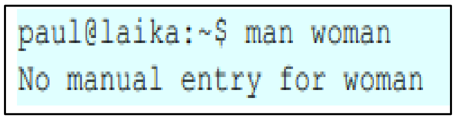


```
[paul@RHEL52 ~]$ man passwd      # opens the first manual found
[paul@RHEL52 ~]$ man 5 passwd    # opens a page from section 5
```

1.9. man man

If you want to know more about man, then Read The Fantastic Manual (RTFM).

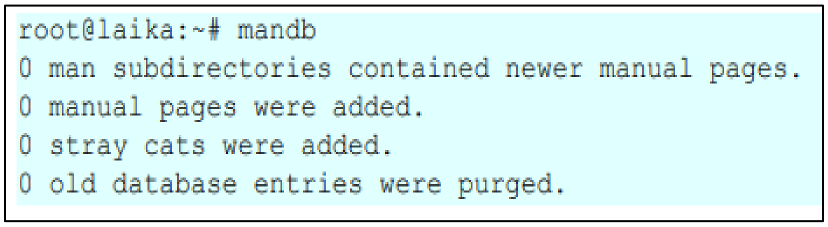
Unfortunately, manual pages do not have the answer to everything...



```
paul@laika:~$ man woman
No manual entry for woman
```

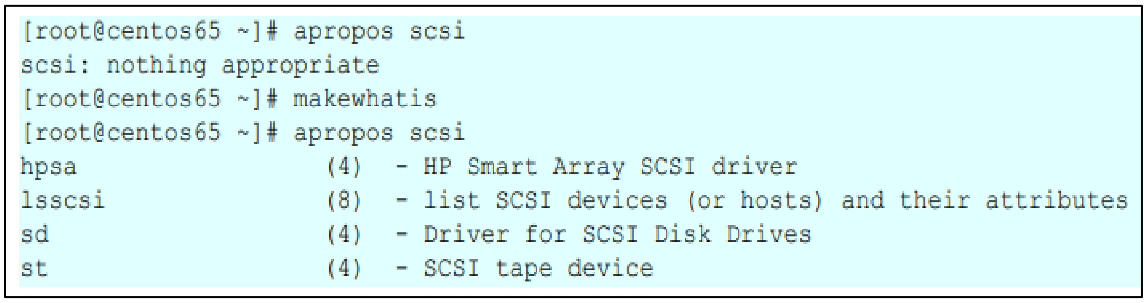
1.10. mandb

Should you be convinced that a man page exists, but you can't access it, then try running mandb on Debian/Mint.



```
root@laika:~# mandb
0 man subdirectories contained newer manual pages.
0 manual pages were added.
0 stray cats were added.
0 old database entries were purged.
```

Or run makewhatis on CentOS/Redhat.



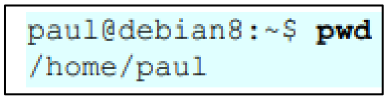
```
[root@centos65 ~]# apropos scsi
scsi: nothing appropriate
[root@centos65 ~]# makewhatis
[root@centos65 ~]# apropos scsi
hpsa                 (4)  - HP Smart Array SCSI driver
lsscsi               (8)  - list SCSI devices (or hosts) and their attributes
sd                   (4)  - Driver for SCSI Disk Drives
st                   (4)  - SCSI tape device
```

2) Working with directories :-

This module is a brief overview of the most common commands to work with directories: pwd, cd, ls, mkdir and rmdir. These commands are available on any Linux (or Unix) system. This module also discusses absolute and relative paths and path completion in the bash shell.

2.1. pwd

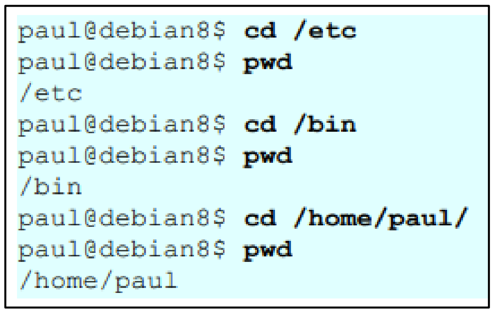
The you are here sign can be displayed with the pwd command (Print Working Directory). Go ahead, try it: Open a command line interface (also called a terminal, console or xterm) and type pwd. The tool displays your current directory.



```
paul@debian8:~$ pwd
/home/paul
```

2.2. cd

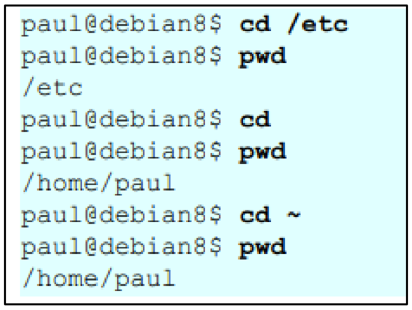
You can change your current directory with the cd command (Change Directory).



```
paul@debian8$ cd /etc
paul@debian8$ pwd
/etc
paul@debian8$ cd /bin
paul@debian8$ pwd
/bin
paul@debian8$ cd /home/paul/
paul@debian8$ pwd
/home/paul
```

2.2.1. cd ~

The cd is also a shortcut to get back into your home directory. Just typing cd without a target directory, will put you in your home directory. Typing cd ~ has the same effect.



```
paul@debian8$ cd /etc
paul@debian8$ pwd
/etc
paul@debian8$ cd
paul@debian8$ pwd
/home/paul
paul@debian8$ cd ~
paul@debian8$ pwd
/home/paul
```

2.2.2. cd ..

To go to the parent directory (the one just above your current directory in the directory tree), type cd .. .

```
paul@debian8$ pwd
/usr/share/games
paul@debian8$ cd ..
paul@debian8$ pwd
/usr/share
```

2.2.3. cd –

Another useful shortcut with cd is to just type cd - to go to the previous directory.

```
paul@debian8$ pwd
/home/paul
paul@debian8$ cd /etc
paul@debian8$ pwd
/etc
paul@debian8$ cd -
/home/paul
paul@debian8$ cd -
/etc
```

2.3. ls

You can list the contents of a directory with ls.

```
paul@debian8:~$ ls
allfiles.txt  dmesg.txt  services   stuff  summer.txt
paul@debian8:~$
```

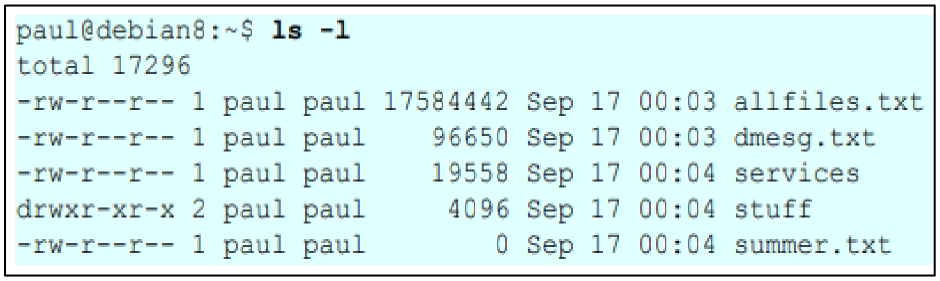
2.3.1. ls -a

A frequently used option with ls is -a to show all files. Showing all files means including the hidden files. When a file name on a Linux file system starts with a dot, it is considered a hidden file and it doesn't show up in regular file listings.

```
paul@debian8:~$ ls
allfiles.txt  dmesg.txt  services  stuff  summer.txt
paul@debian8:~$ ls -a
.   allfiles.txt   .bash_profile  dmesg.txt   .lesshst  stuff
..  .bash_history  .bashrc        services    .ssh      summer.txt
paul@debian8:~$
```

2.3.2. ls -l

Many times you will be using options with ls to display the contents of the directory in different formats or to display different parts of the directory. Typing just ls gives you a list of files in the directory. Typing ls -l (that is a letter L, not the number 1) gives you a long listing.

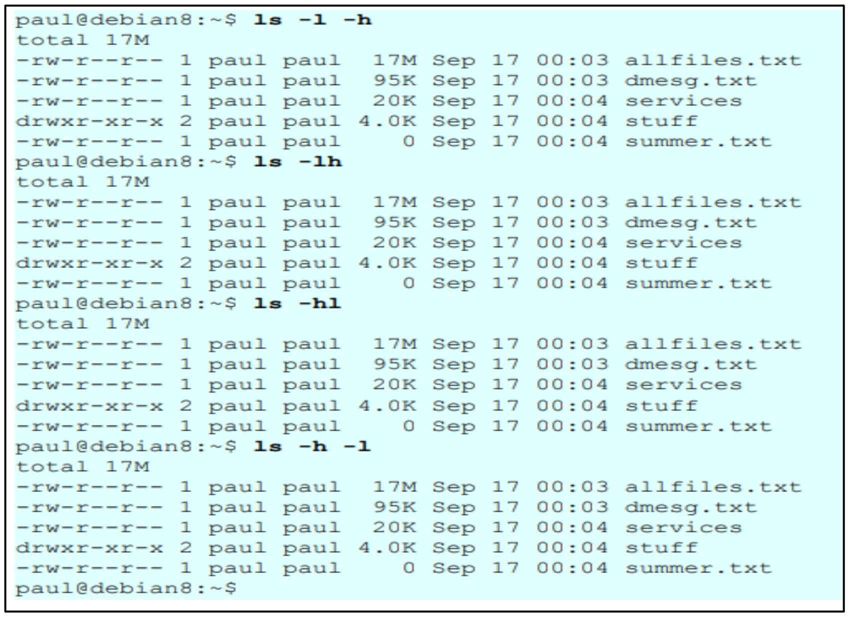


```
paul@debian8:~$ ls -l
total 17296
-rw-r--r-- 1 paul paul 17584442 Sep 17 00:03 allfiles.txt
-rw-r--r-- 1 paul paul    96650 Sep 17 00:03 dmesg.txt
-rw-r--r-- 1 paul paul    19558 Sep 17 00:04 services
drwxr-xr-x 2 paul paul     4096 Sep 17 00:04 stuff
-rw-r--r-- 1 paul paul        0 Sep 17 00:04 summer.txt
```

2.3.3. ls -lh

Another frequently used ls option is -h. It shows the numbers (file sizes) in a more human readable format. Also shown below is some variation in the way you can give the options to ls. We will explain the details of the output later in this book.

Note that we use the letter L as an option in this screenshot, not the number 1.



```
paul@debian8:~$ ls -l -h
total 17M
-rw-r--r-- 1 paul paul  17M Sep 17 00:03 allfiles.txt
-rw-r--r-- 1 paul paul  95K Sep 17 00:03 dmesg.txt
-rw-r--r-- 1 paul paul  20K Sep 17 00:04 services
drwxr-xr-x 2 paul paul 4.0K Sep 17 00:04 stuff
-rw-r--r-- 1 paul paul    0 Sep 17 00:04 summer.txt
paul@debian8:~$ ls -lh
total 17M
-rw-r--r-- 1 paul paul  17M Sep 17 00:03 allfiles.txt
-rw-r--r-- 1 paul paul  95K Sep 17 00:03 dmesg.txt
-rw-r--r-- 1 paul paul  20K Sep 17 00:04 services
drwxr-xr-x 2 paul paul 4.0K Sep 17 00:04 stuff
-rw-r--r-- 1 paul paul    0 Sep 17 00:04 summer.txt
paul@debian8:~$ ls -hl
total 17M
-rw-r--r-- 1 paul paul  17M Sep 17 00:03 allfiles.txt
-rw-r--r-- 1 paul paul  95K Sep 17 00:03 dmesg.txt
-rw-r--r-- 1 paul paul  20K Sep 17 00:04 services
drwxr-xr-x 2 paul paul 4.0K Sep 17 00:04 stuff
-rw-r--r-- 1 paul paul    0 Sep 17 00:04 summer.txt
paul@debian8:~$ ls -h -l
total 17M
-rw-r--r-- 1 paul paul  17M Sep 17 00:03 allfiles.txt
-rw-r--r-- 1 paul paul  95K Sep 17 00:03 dmesg.txt
-rw-r--r-- 1 paul paul  20K Sep 17 00:04 services
drwxr-xr-x 2 paul paul 4.0K Sep 17 00:04 stuff
-rw-r--r-- 1 paul paul    0 Sep 17 00:04 summer.txt
paul@debian8:~$
```

2.4. mkdir

Walking around the Unix file tree is fun, but it is even more fun to create your own directories with mkdir. You have to give at least one parameter to mkdir, the name of the new directory to be created. Think before you type a leading / .

```
paul@debian8:~$ mkdir mydir
paul@debian8:~$ cd mydir
paul@debian8:~/mydir$ ls -al
total 8
drwxr-xr-x  2 paul paul 4096 Sep 17 00:07 .
drwxr-xr-x 48 paul paul 4096 Sep 17 00:07 ..
paul@debian8:~/mydir$ mkdir stuff
paul@debian8:~/mydir$ mkdir otherstuff
paul@debian8:~/mydir$ ls -l
total 8
drwxr-xr-x 2 paul paul 4096 Sep 17 00:08 otherstuff
drwxr-xr-x 2 paul paul 4096 Sep 17 00:08 stuff
paul@debian8:~/mydir$
```

2.4.1. mkdir -p

When given the option -p, then mkdir will create parent directories as needed.

```
paul@debian8:~$ mkdir -p mydir2/mysubdir2/threedirsdeep
paul@debian8:~$ cd mydir2
paul@debian8:~/mydir2$ ls -l
total 4
drwxr-xr-x 3 paul paul 4096 Sep 17 00:11 mysubdir2
paul@debian8:~/mydir2$ cd mysubdir2
paul@debian8:~/mydir2/mysubdir2$ ls -l
total 4
drwxr-xr-x 2 paul paul 4096 Sep 17 00:11 threedirsdeep
paul@debian8:~/mydir2/mysubdir2$ cd threedirsdeep/
paul@debian8:~/mydir2/mysubdir2/threedirsdeep$ pwd
/home/paul/mydir2/mysubdir2/threedirsdeep
```

2.5. rmdir

When a directory is empty, you can use rmdir to remove the directory.

```
paul@debian8:~/mydir$ ls -l
total 8
drwxr-xr-x 2 paul paul 4096 Sep 17 00:08 otherstuff
drwxr-xr-x 2 paul paul 4096 Sep 17 00:08 stuff
paul@debian8:~/mydir$ rmdir otherstuff
paul@debian8:~/mydir$ cd ..
paul@debian8:~$ rmdir mydir
rmdir: failed to remove 'mydir': Directory not empty
paul@debian8:~$ rmdir mydir/stuff
paul@debian8:~$ rmdir mydir
paul@debian8:~$
```

2.5.1. rmdir -p

And similar to the mkdir -p option, you can also use rmdir to recursively remove directories.

```
paul@debian8:~$ mkdir -p test42/subdir
paul@debian8:~$ rmdir -p test42/subdir
paul@debian8:~$
```

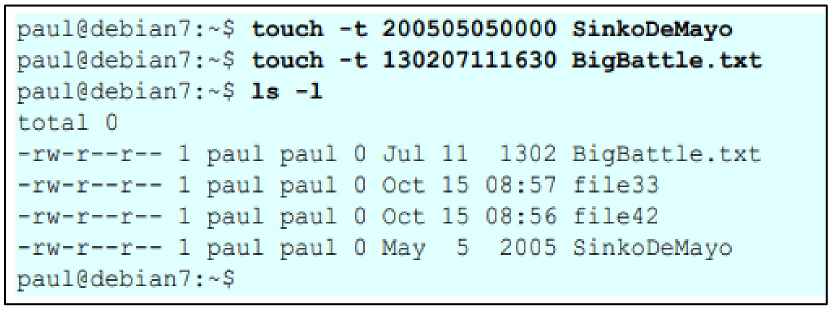
3) <u>Working with files</u> :-

We learn how to recognise, create, remove, copy and move files using commands like touch, rm, cp, mv and rename.

3.1. touch

3.1.1. touch -t

The touch command can set some properties while creating empty files.

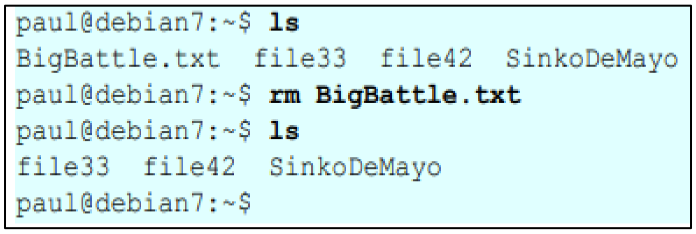


```
paul@debian7:~$ touch -t 200505050000 SinkoDeMayo
paul@debian7:~$ touch -t 130207111630 BigBattle.txt
paul@debian7:~$ ls -l
total 0
-rw-r--r-- 1 paul paul 0 Jul 11  1302 BigBattle.txt
-rw-r--r-- 1 paul paul 0 Oct 15 08:57 file33
-rw-r--r-- 1 paul paul 0 Oct 15 08:56 file42
-rw-r--r-- 1 paul paul 0 May  5  2005 SinkoDeMayo
paul@debian7:~$
```

3.2. rm

3.2.1. remove forever

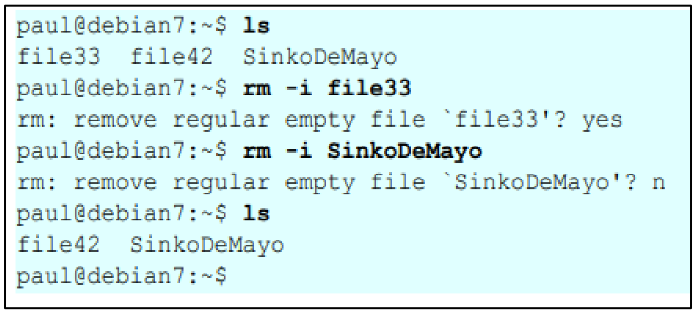
When you no longer need a file, use rm to remove it. Unlike some graphical user interfaces, the command line in general does not have a waste bin or trash can to recover files. When you use rm to remove a file, the file is gone. Therefore, be careful when removing files!.



```
paul@debian7:~$ ls
BigBattle.txt  file33  file42  SinkoDeMayo
paul@debian7:~$ rm BigBattle.txt
paul@debian7:~$ ls
file33  file42  SinkoDeMayo
paul@debian7:~$
```

3.2.2. rm -i

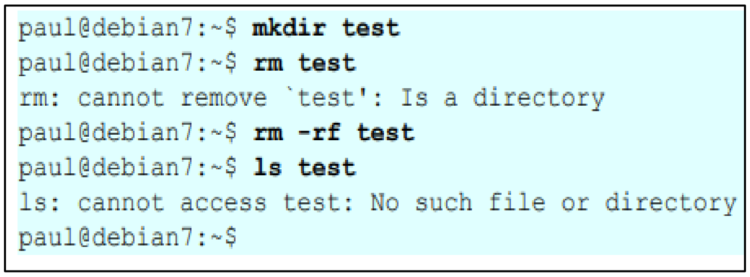
To prevent yourself from accidentally removing a file, you can type rm -i.



```
paul@debian7:~$ ls
file33  file42  SinkoDeMayo
paul@debian7:~$ rm -i file33
rm: remove regular empty file `file33'? yes
paul@debian7:~$ rm -i SinkoDeMayo
rm: remove regular empty file `SinkoDeMayo'? n
paul@debian7:~$ ls
file42  SinkoDeMayo
paul@debian7:~$
```

3.2.3. 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.

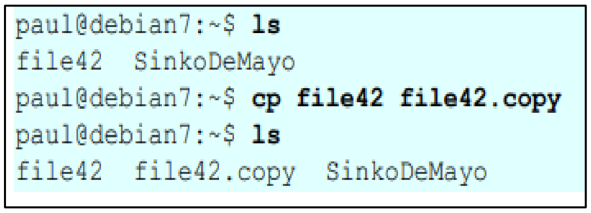


```
paul@debian7:~$ mkdir test
paul@debian7:~$ rm test
rm: cannot remove `test': Is a directory
paul@debian7:~$ rm -rf test
paul@debian7:~$ ls test
ls: cannot access test: No such file or directory
paul@debian7:~$
```

3.3. cp

3.3.1. copy one file

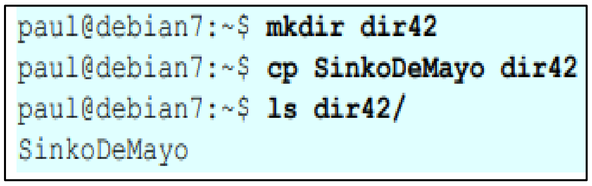
To copy a file, use cp with a source and a target argument.



```
paul@debian7:~$ ls
file42  SinkoDeMayo
paul@debian7:~$ cp file42 file42.copy
paul@debian7:~$ ls
file42  file42.copy  SinkoDeMayo
```

3.3.2. copy to another directory

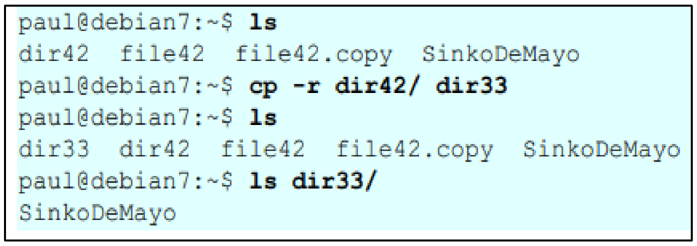
If the target is a directory, then the source files are copied to that target directory.



```
paul@debian7:~$ mkdir dir42
paul@debian7:~$ cp SinkoDeMayo dir42
paul@debian7:~$ ls dir42/
SinkoDeMayo
```

3.3.3. cp -r

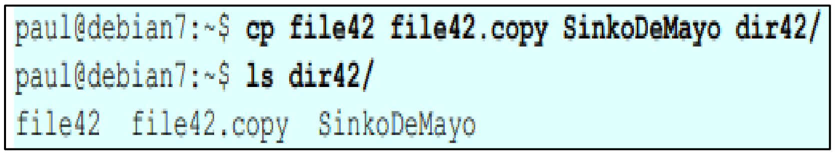
To copy complete directories, use cp -r (the -r option forces recursive copying of all files in all subdirectories).



```
paul@debian7:~$ ls
dir42  file42  file42.copy  SinkoDeMayo
paul@debian7:~$ cp -r dir42/ dir33
paul@debian7:~$ ls
dir33  dir42  file42  file42.copy  SinkoDeMayo
paul@debian7:~$ ls dir33/
SinkoDeMayo
```

3.3.4. copy multiple files to directory

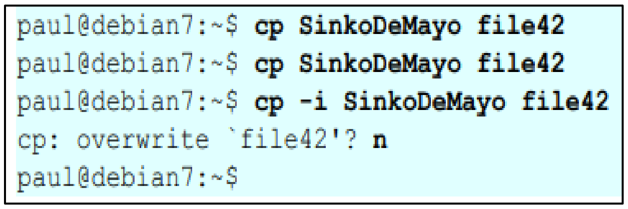
We can also use cp to copy multiple files into a directory. In this case, the last argument (a.k.a. the target) must be a directory.



```
paul@debian7:~$ cp file42 file42.copy SinkoDeMayo dir42/
paul@debian7:~$ ls dir42/
file42  file42.copy  SinkoDeMayo
```

3.3.5. cp -i

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

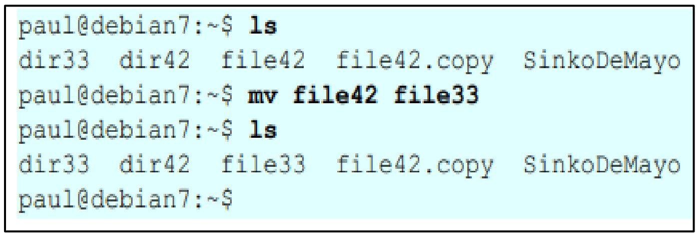


```
paul@debian7:~$ cp SinkoDeMayo file42
paul@debian7:~$ cp SinkoDeMayo file42
paul@debian7:~$ cp -i SinkoDeMayo file42
cp: overwrite `file42'? n
paul@debian7:~$
```

3.4. mv

3.4.1. rename files with mv

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

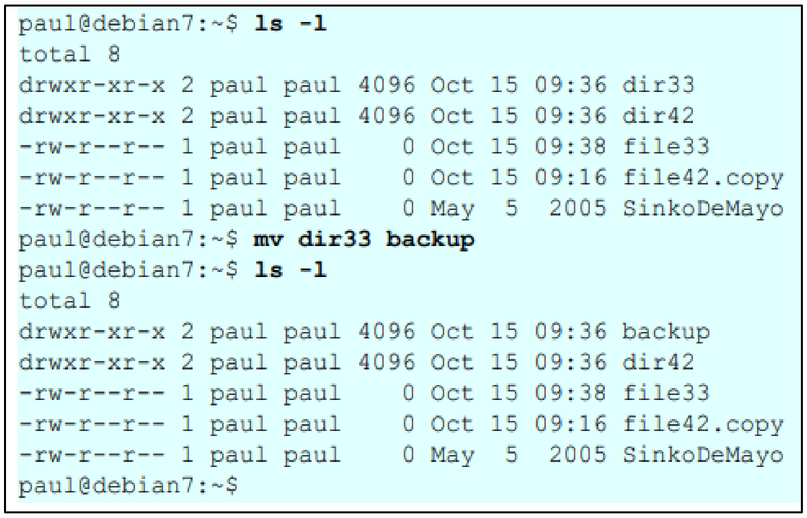


```
paul@debian7:~$ ls
dir33  dir42  file42  file42.copy  SinkoDeMayo
paul@debian7:~$ mv file42 file33
paul@debian7:~$ ls
dir33  dir42  file33  file42.copy  SinkoDeMayo
paul@debian7:~$
```

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

3.4.2. rename directories with mv

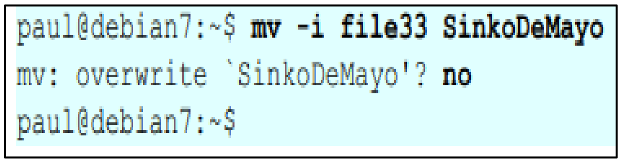
The same mv command can be used to rename directories.



```
paul@debian7:~$ ls -l
total 8
drwxr-xr-x 2 paul paul 4096 Oct 15 09:36 dir33
drwxr-xr-x 2 paul paul 4096 Oct 15 09:36 dir42
-rw-r--r-- 1 paul paul    0 Oct 15 09:38 file33
-rw-r--r-- 1 paul paul    0 Oct 15 09:16 file42.copy
-rw-r--r-- 1 paul paul    0 May  5  2005 SinkoDeMayo
paul@debian7:~$ mv dir33 backup
paul@debian7:~$ ls -l
total 8
drwxr-xr-x 2 paul paul 4096 Oct 15 09:36 backup
drwxr-xr-x 2 paul paul 4096 Oct 15 09:36 dir42
-rw-r--r-- 1 paul paul    0 Oct 15 09:38 file33
-rw-r--r-- 1 paul paul    0 Oct 15 09:16 file42.copy
-rw-r--r-- 1 paul paul    0 May  5  2005 SinkoDeMayo
paul@debian7:~$
```

3.4.3. mv -i

The mv also has a -i switch similar to cp and rm. this screenshot shows that mv -i will ask permission to overwrite an existing file.



```
paul@debian7:~$ mv -i file33 SinkoDeMayo
mv: overwrite `SinkoDeMayo'? no
paul@debian7:~$
```

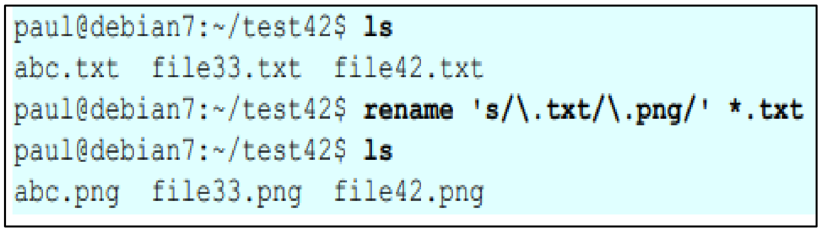
3.5. rename

3.5.1. about rename

The rename command is one of the rare occasions where the Linux Fundamentals book has to make a distinction between Linux distributions. Almost every command in the Fundamentals part of this book works on almost every Linux computer. But rename is different. Try to use mv whenever you need to rename only a couple of files.

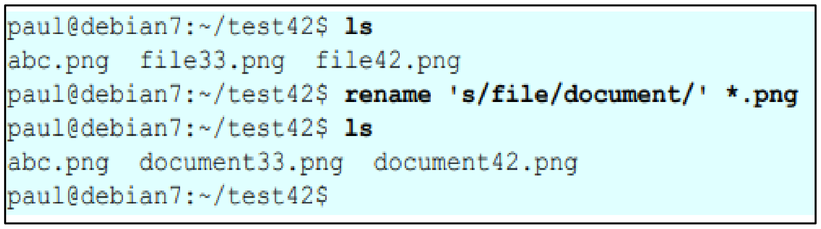
3.5.2. rename on Debian/Ubuntu

The rename command on Debian uses regular expressions (regular expression or shor regex are explained in a later chapter) to rename many files at once. Below a rename example that switches all occurrences of txt to png for all file names ending in .txt.



```
paul@debian7:~/test42$ ls
abc.txt  file33.txt  file42.txt
paul@debian7:~/test42$ rename 's/\.txt/\.png/' *.txt
paul@debian7:~/test42$ ls
abc.png  file33.png  file42.png
```

This second example switches all (first) occurrences of file into document for all file names ending in .png.

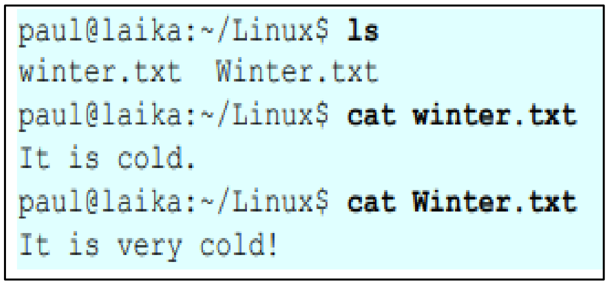


```
paul@debian7:~/test42$ ls
abc.png  file33.png  file42.png
paul@debian7:~/test42$ rename 's/file/document/' *.png
paul@debian7:~/test42$ ls
abc.png  document33.png  document42.png
paul@debian7:~/test42$
```

3.6. file

3.6.1. all files are case sensitive

Files on Linux (or any Unix) are 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.



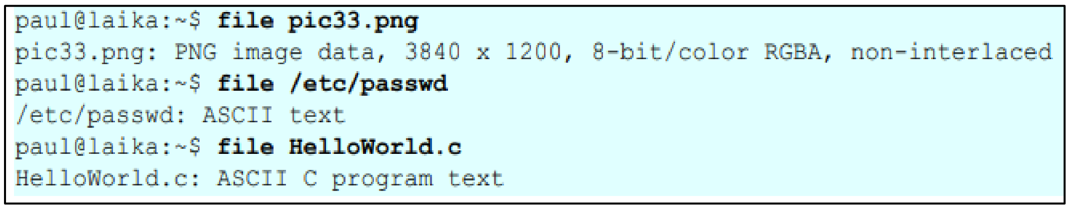
```
paul@laika:~/Linux$ ls
winter.txt  Winter.txt
paul@laika:~/Linux$ cat winter.txt
It is cold.
paul@laika:~/Linux$ cat Winter.txt
It is very cold!
```

3.6.2. everything is a file

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.

3.6.3. file

The file utility determines the file type. Linux does not use extensions to determine the file type. The command line does not care whether a file ends in .txt or .pdf. 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
```

4) Working with file contents :-

We will look at the contents of text files with head, tail, cat, tac, more, less and strings. We will also get a glimpse of the possibilities of tools like cat on the command line.

4.1. head

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

```
paul@debian7~$ 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
root@debian7~#
```

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

```
paul@debian7~$ 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
paul@debian7~$
```

And head can also display the first n bytes.

```
paul@debian7~$ head -c14 /etc/passwd
root:x:0:0:roopaul@debian7~$
```

4.2. tail

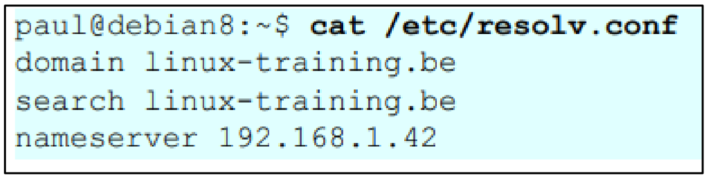
Similar to head, the tail command will display the last ten lines of a file.

```
paul@debian7~$ 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
dircproxy       57000/tcp                       # Detachable IRC Proxy
tfido           60177/tcp                       # fidonet EMSI over telnet
fido            60179/tcp                       # fidonet EMSI over TCP

# Local services
paul@debian7~$
```

4.3. cat

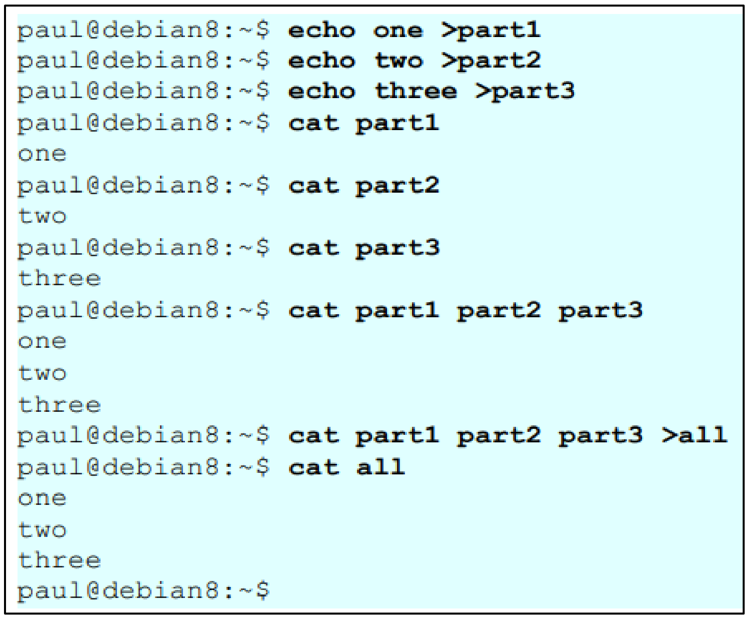
The cat command is one of the most universal tools, yet all it does is copy standard input to standard output. In combination with the shell this can be very powerful and diverse. Some examples will give a glimpse into the possibilities. The first example is simple, you can use cat to display a file on the screen. If the file is longer than the screen, it will scroll to the end.



```
paul@debian8:~$ cat /etc/resolv.conf
domain linux-training.be
search linux-training.be
nameserver 192.168.1.42
```

4.3.1. concatenate

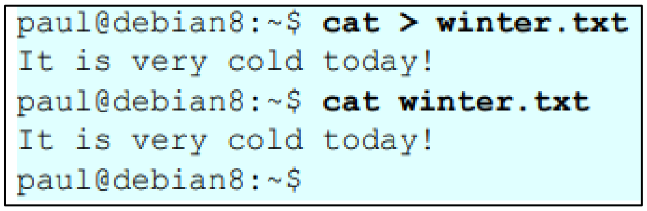
cat is short for concatenate. One of the basic uses of cat is to concatenate files into a bigger (or complete) file.



```
paul@debian8:~$ echo one >part1
paul@debian8:~$ echo two >part2
paul@debian8:~$ echo three >part3
paul@debian8:~$ cat part1
one
paul@debian8:~$ cat part2
two
paul@debian8:~$ cat part3
three
paul@debian8:~$ cat part1 part2 part3
one
two
three
paul@debian8:~$ cat part1 part2 part3 >all
paul@debian8:~$ cat all
one
two
three
paul@debian8:~$
```

4.3.2. 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.



```
paul@debian8:~$ cat > winter.txt
It is very cold today!
paul@debian8:~$ cat winter.txt
It is very cold today!
paul@debian8:~$
```

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

4.3.3. custom end marker

You can choose an end marker for cat with << as is shown in this screenshot. This construction is called a here directive and will end the cat command.

```
paul@debian8:~$ cat > hot.txt <<stop
> It is hot today!
> Yes it is summer.
> stop
paul@debian8:~$ cat hot.txt
It is hot today!
Yes it is summer.
paul@debian8:~$
```

4.3.4. copy files

```
paul@debian8:~$ cat winter.txt
It is very cold today!
paul@debian8:~$ cat winter.txt > cold.txt
paul@debian8:~$ cat cold.txt
It is very cold today!
paul@debian8:~$
```

4.4. tac

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

```
paul@debian8:~$ cat count
one
two
three
four
paul@debian8:~$ tac count
four
three
two
one
```

4.5. more and less

The more command is useful for displaying files that take up more than one screen. More will allow you to see the contents of the file page by page. Use the space bar to see the next page, or q to quit. Some people prefer the less command to more.

4.6. strings

With the strings command you can display readable ascii strings found in (binary) files. This example locates the ls binary then displays readable strings in the binary file (output is truncated).

```
paul@laika:~$ which ls
/bin/ls
paul@laika:~$ strings /bin/ls
/lib/ld-linux.so.2
librt.so.1
__gmon_start__
_Jv_RegisterClasses
clock_gettime
libacl.so.1
...
```