Chapter 6. Working with directories

To explore the Linux **file tree**, you will need some basic tools.

This chapter is a small overview of the most common commands to work with directories : **pwd, cd, ls, mkdir, rmdir**. These commands are available on any Linux (or Unix) system.

This chapter also discusses **absolute** and **relative paths** and **path completion** in the **bash** shell.

6.1. 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 (like gnome-terminal, konsole, xterm, or a tty) and type **pwd**. The tool displays your **current directory**.

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

6.2. cd

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

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

6.2.1. cd ~

You can pull off a trick with cd. Just typing **cd** without a target directory, will put you in your home directory. Typing **cd ~** has the same effect.

```
paul@laika$ cd /etc
paul@laika$ pwd
/etc
paul@laika$ cd
paul@laika$ wd
/home/paul
paul@laika$ cd ~
paul@laika$ pwd
/home/paul
```

6.2.2. cd ..

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

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

*To stay in the current directory, type **cd .** ;-)* We will see useful use of the **.** character representing the current directory later.

6.2.3. cd -

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

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

6.3. absolute and relative paths

You should be aware of **absolute and relative paths** in the file tree. When you type a path starting with a **slash (/)**, then the **root** of the file tree is assumed. If you don't start your path with a slash, then the current directory is the assumed starting point.

The screenshot below first shows the current directory **/home/paul**. From within this directory, you have to type **cd /home** instead of **cd home** to go to the **/home** directory.

```
paul@laika$ pwd
/home/paul
paul@laika$ cd home
bash: cd: home: No such file or directory
paul@laika$ cd /home
paul@laika$ pwd
/home
```

When inside **/home**, you have to type **cd paul** instead of **cd /paul** to enter the subdirectory **paul** of the current directory **/home**.

```
paul@laika$ pwd
/home
paul@laika$ cd /paul
bash: cd: /paul: No such file or directory
paul@laika$ cd paul
paul@laika$ pwd
/home/paul
```

In case your current directory is the **root directory /**, then both **cd /home** and **cd home** will get you in the **/home** directory.

```
paul@laika$ pwd
/
paul@laika$ cd home
paul@laika$ pwd
/home
paul@laika$ cd /
paul@laika$ cd /home
paul@laika$ pwd
```

This was the last screenshot with **pwd** statements. From now on, the current directory will often be displayed in the prompt. Later in this book we will explain how the shell variable **$PS1** can be configured to show this.

6.4. path completion

The **tab key** can help you in typing a path without errors. Typing **cd /et** followed by the **tab key** will expand the command line to **cd /etc/**. When typing **cd /Et** followed by the **tab key**, nothing will happen because you typed the wrong **path** (uppercase E).

You will need fewer keystrokes when using the **tab key**, and you will be sure your typed **path** is correct!

6 .5. ls

You can list the contents of a directory with **ls**.

```
paul@pasha:~$ ls
allfiles.txt  dmesg.txt  httpd.conf  stuff  summer.txt
paul@pasha:~$
```

6.5.1. 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 Unix file system starts with a dot, it is considered a hidden file and it doesn't show up in regular file listings.

```
paul@pasha:~$ ls
allfiles.txt  dmesg.txt  httpd.conf  stuff  summer.txt
paul@pasha:~$ ls -a
.   allfiles.txt   .bash_profile  dmesg.txt   .lesshst  stuff

..  .bash_history  .bashrc        httpd.conf  .ssh      summer.txt
paul@pasha:~$
```

6.5.2. 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@pasha:~$ ls -l
total 23992
-rw-r--r-- 1 paul paul 24506857 2006-03-30 22:53 allfiles.txt
-rw-r--r-- 1 paul paul    14744 2006-09-27 11:45 dmesg.txt
-rw-r--r-- 1 paul paul     8189 2006-03-31 14:01 httpd.conf
drwxr-xr-x 2 paul paul     4096 2007-01-08 12:22 stuff
-rw-r--r-- 1 paul paul        0 2006-03-30 22:45 summer.txt
```

6.5.3. 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.

```
paul@pasha:~$ ls -l -h
total 24M
-rw-r--r-- 1 paul paul  24M 2006-03-30 22:53 allfiles.txt
-rw-r--r-- 1 paul paul  15K 2006-09-27 11:45 dmesg.txt
-rw-r--r-- 1 paul paul 8.0K 2006-03-31 14:01 httpd.conf
drwxr-xr-x 2 paul paul 4.0K 2007-01-08 12:22 stuff

-rw-r--r-- 1 paul paul    0 2006-03-30 22:45 summer.txt
paul@pasha:~$ ls -lh
total 24M
-rw-r--r-- 1 paul paul  24M 2006-03-30 22:53 allfiles.txt
-rw-r--r-- 1 paul paul  15K 2006-09-27 11:45 dmesg.txt
-rw-r--r-- 1 paul paul 8.0K 2006-03-31 14:01 httpd.conf
drwxr-xr-x 2 paul paul 4.0K 2007-01-08 12:22 stuff
-rw-r--r-- 1 paul paul    0 2006-03-30 22:45 summer.txt
paul@pasha:~$ ls -hl
total 24M
-rw-r--r-- 1 paul paul  24M 2006-03-30 22:53 allfiles.txt
-rw-r--r-- 1 paul paul  15K 2006-09-27 11:45 dmesg.txt
-rw-r--r-- 1 paul paul 8.0K 2006-03-31 14:01 httpd.conf
drwxr-xr-x 2 paul paul 4.0K 2007-01-08 12:22 stuff
-rw-r--r-- 1 paul paul    0 2006-03-30 22:45 summer.txt
paul@pasha:~$ ls -h -l
total 24M
-rw-r--r-- 1 paul paul  24M 2006-03-30 22:53 allfiles.txt
-rw-r--r-- 1 paul paul  15K 2006-09-27 11:45 dmesg.txt
-rw-r--r-- 1 paul paul 8.0K 2006-03-31 14:01 httpd.conf
drwxr-xr-x 2 paul paul 4.0K 2007-01-08 12:22 stuff
-rw-r--r-- 1 paul paul    0 2006-03-30 22:45 summer.txt
```

6.6. 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@laika:~$ mkdir MyDir
paul@laika:~$ cd MyDir
paul@laika:~/MyDir$ ls -al
total 8
drwxr-xr-x  2 paul paul 4096 2007-01-10 21:13 .
drwxr-xr-x 39 paul paul 4096 2007-01-10 21:13 ..
paul@laika:~/MyDir$ mkdir stuff
paul@laika:~/MyDir$ mkdir otherstuff
paul@laika:~/MyDir$ ls -l
total 8
drwxr-xr-x 2 paul paul 4096 2007-01-10 21:14 otherstuff
drwxr-xr-x 2 paul paul 4096 2007-01-10 21:14 stuff
```

6.6.1. mkdir -p

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

```
paul@laika:~$ mkdir -p MyDir2/MySubdir2/ThreeDeep
paul@laika:~$ ls MyDir2
MySubdir2
paul@laika:~$ ls MyDir2/MySubdir2
ThreeDeep
paul@laika:~$ ls MyDir2/MySubdir2/ThreeDeep/
```

6.7. rmdir

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

```
paul@laika:~/MyDir$ rmdir otherstuff
paul@laika:~/MyDir$ ls
stuff
paul@laika:~/MyDir$ cd ..
paul@laika:~$ rmdir MyDir
rmdir: MyDir/: Directory not empty
paul@laika:~$ rmdir MyDir/stuff
paul@laika:~$ rmdir MyDir
```

6.7.1. rmdir -p

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

```
paul@laika:~$ mkdir -p dir/subdir/subdir2
paul@laika:~$ rmdir -p dir/subdir/subdir2
paul@laika:~$
```