

Basic Linux Command

In the following subsections, we'll look at common commands, most of which are installed on most Linux systems by default.

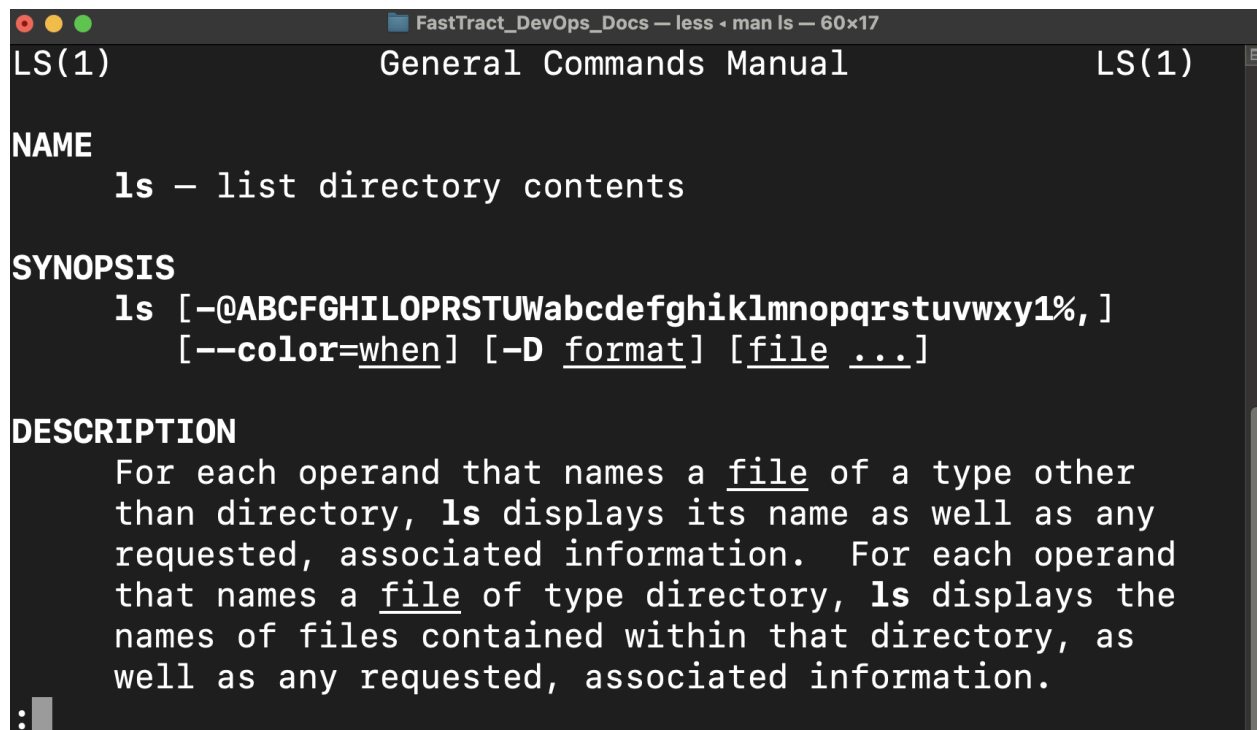
Reading the Manual with the man Command

I'll mention many programs throughout this book. For most of them, I'll only go into about 5–10% of their usage at best. If you want to explore these programs deeper, it's important you learn the man command which can be found on almost all Linux operating systems.

`man` is short for manual. It is used by running the command and passing in the name of another Linux command-line program. For example, if we wanted to get more information on the command `ls`, we would run the following:

```
man ls
```

This returns a description of the program and how to use it, as shown in Figure 1 below:



```
LS(1)                                General Commands Manual                                LS(1)

NAME
    ls - list directory contents

SYNOPSIS
    ls [-@ABCFGHILOPRSTUWabcdefghiklmnopqrstuvwxy1%, ]
        [--color=when] [-D format] [file ...]

DESCRIPTION
    For each operand that names a file of a type other
    than directory, ls displays its name as well as any
    requested, associated information.  For each operand
    that names a file of type directory, ls displays the
    names of files contained within that directory, as
    well as any requested, associated information.
```

If you need to search through a man page to find some specific keyword, there is a built-in search function. To search, press / and then type in your term and press enter. You'll be taken to the first occurrence if one exists. To go to the next occurrence, tap n; each tap of n will bring you to the next instance. If you want to go back an instance, press capital N, as with n each press goes back an instance.

Numbered man Pages

In some cases, there may be multiple man pages for a single program. For example, with the program stat, we can run

```
man 1 stat
```

Or we can run:

```
man 2 stat
```

These commands will bring us to different man pages which concern different aspects of the program. See Table 1-1 for a list of the different page numbers and what information they include.

Table 1-1. Description of information found in numbered man pages

DESCRIPTION

The **man** utility finds and displays online manual documentation pages. If mansect is provided, **man** restricts the search to the specific section of the manual.

The sections of the manual are:

1. General Commands Manual
2. System Calls Manual
3. Library Functions Manual
4. Kernel Interfaces Manual
5. File Formats Manual
6. Games Manual
7. Miscellaneous Information Manual
8. System Manager's Manual
9. Kernel Developer's Manual

Options that **man** understands:

Useful Commands for Navigating

Some commands you'll want to get familiar with for navigating and creating new folders are listed in Table 2.

Table 2. Commands for navigating and working with files/directories

Command	Description
ls	List directory contents
cd	Change directory
pwd	Print working directory
mkdir	Make directory
rmdir	Remove directory (<i>only works if empty</i>)

Navigating the Filesystem with ls and cd

The first commands most users learn when introduced to the filesystem are ls, short for “list directory contents,” and cd, short for “change directory.”

One thing to keep in mind when using cd. At any time you can run cd without any folder name to return to your home directory. If you go into a directory with cd and want to return to the directory containing the one you're in, you can use .., for example:

```
cd ..
```

Or if you wanted to return two folders up:

```
cd ../..
```

Some common things you might want to do with `ls` are list additional details beyond just the names of the files and folders; this can be done with the `-l` flag:

```
ls -l
```

If you want to sort by time last modified, you can use the `-t` flag, which is best combined with `-l`:

```
ls -lt
```

If you want to reverse the results so the oldest files are at the top, add the `-r` flag for reverse:

```
ls -ltr
```

Invisible Files (dot files)

It's important to know that files which start with a `.` in Linux will not normally appear when using `ls` or a graphical file explorer. These files are meant for configuration and are *hidden* for convenience. Often we'll want to edit or look at these files so it's important to know about the `-a` flag for `ls`. The `-a` stands for all and will show all files including hidden ones.

```
ls -a
```

Get Current Directory with `pwd`

With all this navigating, it's easy to forget exactly where you are on the filesystem. If this happens, there is an easy solution to figuring out exactly where you are. Simply run `pwd` which will return the full path of your current location.

```
pwd
```

Make a Directory

Part of navigating the filesystem is creating new directories to put your files and subfolders in. This is relatively easy with the `mkdir` command which takes the folder name and will create the directory based on your current location. For example, if we run the following command in our home directory:

```
mkdir music
```

we'll end up with a folder called music. There is no limit to how many you can create at once. Say we want to create two additional subfolders, we could run

```
mkdir music/rock music/classical
```

It's also possible to use a full path instead of a relative one. For example, if I'm in my home directory and I want to make a new folder in my /tmp folder:

```
mkdir /tmp/test
```

This isn't unique to mkdir; essentially all programs where you can use a relative path also allow you to make use of a full path; it only requires starting the path with a "/".

Recursively Make Directories

Often when creating a folder, you already have a structure which is multiple directories deep in mind. For example, say we want to create a new folder called movies, with a subfolder for horror and another subfolder for 2012. If we run

```
mkdir movies/horror/2012
```

We'll get back an error saying "No such file or directory". The -p tag provides a way around this. -p stands for create parent directories, meaning if the parent directories of the directory we want to create don't exist, they will be created. Running the following command works as expected, leaving us with three new folders:

```
mkdir -p movies/horror/2012
```

Delete a Directory

After creating a directory, you may decide that you want to delete it. One way of doing this is with the `rmdir` command which is used similarly to `mkdir`; simply pass it the name of the directory you want to remove:

```
rmdir music/classical
```

Unfortunately, `rmdir` has a major limitation in that it can only delete a completely empty directory. Trying to use `rmdir` on any directory which contains a file or subdirectory will

return “Directory not empty”. Thus, in practice many people instead always use the command

```
rm -r music
```

The -r in this command stands for recursive. This command is practical as it will work on both files and directories regardless of whether the directory contains any content.