

Linux tutorial

Darin Brezeale

August 29, 2009

GNU/Linux, more commonly referred to just as Linux, is a Unix-like operating system. While modern distributions of Linux include graphical interfaces, a.k.a. GUIs, there is much that can be done from a command-prompt. This tutorial will focus on basic commands for working from the command-prompt.

I have also included some books in the reference that you might find useful. I cited the editions that I have, but several of these have newer editions.

1 Home Directory

Because Linux is a multi-user operating system, each user has his own directory which is known as the home directory. When you log in, you will see a command-prompt. Command-prompts are customizable, but for this tutorial we will assume that a user with a user ID of `abc1234` will see the following prompt at the root level of his home directory when logging in:

```
[abc1234] $
```

where the cursor would be after the dollar sign. This is where you type your commands for compiling programs, editing files, creating directories, and so forth. In this tutorial, I will show a command at the prompt followed by the results of running the command.

The Unix/Linux philosophy for command-line utilities is that they should each do one thing and do it well, for example, the `sort` command sorts the lines of a file. While each command may do a single thing, they often include options. For example, if I have a file called `numbers.txt` with the contents

```
one
two
three
four
```

then applying `sort` to this file produces the following:

```
[abc1234] $ sort numbers.txt
four
one
three
two
```

We can use the `-r` option to reverse the order:

```
[abc1234] $ sort -r numbers.txt
two
three
one
four
```

How do we know what options a command might have? We consult the manual page for the command, using the `man` command:

```
[abc1234] $ man sort
```

displays the man page for `sort`. To move down in the display, you press the space bar. To end the display, press `q`.

When logged into Linux, we don't deal directly with the operating system. Instead, we work in an environment called a *shell*. There are many different shells, but one of the more common is **bash** [NR98]. To see which shell you are using, use

```
[abc1234] $ echo $SHELL
/bin/bash
```

2 General Navigation

In Windows, files are grouped into folders. In Linux, instead of folder we use the term *directory*. Below are some commands for working with directories.

pwd is used to show what directory you are currently in:

```
[abc1234/] $ pwd
/home/a/ab/abc1234
```

cd changes to another directory:

```
[abc1234] $ cd cse1310
[abc1234/cse1310] $
```

ls shows the contents of a directory:

```
[abc1234/cse1310] $ ls
numbers.txt test  test.c
```

We can see the names of the files and directories in the directory we are currently in. Each file and directory has information associated with it: the size in bytes, timestamp when it was last edited, and so forth. We can see this information with the `-l` (that's a lowercase letter L) option for `ls`:

```
[abc1234/cse1310] $ ls -l
total 16
-rw-r--r-- 1 abc1234 students  19 Aug 29 17:16 numbers.txt
-rwxr-xr-x 1 abc1234 students 6720 Aug 25 13:45 test
-rw-r--r-- 1 abc1234 students  99 Aug 25 13:45 test.c
```

To return to our home directory, we use

```
[abc1234/cse1310] $ cd
[abc1234/] $
```

To return to whatever directory we were last in, we use

```
[abc1234/] $ cd -
[abc1234/cse1310] $
```

Additional features of the bash shell (as well as others) is command-line completion and shell history. When applying a command to a file, pressing the `tab` key will complete the filename if it is unambiguous. For example, if the directory I am in has the following contents:

```
[abc1234/cse1310] $ ls
numbers.txt test  test.c
```

then to sort the file `numbers.txt`, I would type `sort n<TAB>` where `<TAB>` indicates pressing the `tab` key.

Often we will wish to reuse a command. To go back through our history of commands, we can use the `UP` and `DOWN` keys.

3 File Manipulation Commands

Much of what we do from the command-line involves files, so there are many commands for managing them.

mkdir creates a directory:

```
[abc1234/cse1310] $ mkdir homework
[abc1234/cse1310] $ ls
homework numbers.txt test  test.c
```

rmdir deletes an empty directory:

```
[abc1234/cse1310] $ rmdir homework
[abc1234/cse1310] $ ls
numbers.txt test  test.c
```

cat displays the contents of a file:

```
[abc1234/cse1310] $ cat numbers.txt
one
two
three
four
```

rm deletes a file:

```
[abc1234/cse1310] $ rm test
[abc1234/cse1310] $ ls
numbers.txt  test.c
```

cp copies a file:

```
[abc1234/cse1310] $ cp test.c newtest.c
[abc1234/cse1310] $ ls
newtest.c  numbers.txt  test.c
```

mv moves a file to a new location or a new name (i.e., renames it):

```
[abc1234/cse1310] $ ls
newtest.c  numbers.txt  test.c
[abc1234/cse1310] $ mv newtest.c oldtest.c
[abc1234/cse1310] $ ls
numbers.txt  oldtest.c  test.c
```

4 Compiling a C Program

We have the following source code:

```
#include <stdio.h>

int main(void)
{
    int age = 42;

    printf("I am %d years old.\n", age);
}
```

which is saved as `test.c`. To compile this source code, we can use:

```
[abc1234/cse1310] $ gcc test.c
```

which produces an executable program called `a.out`:

```
[abc1234/cse1310] $ ls
a.out  numbers.txt  oldtest.c  test.c
```

We run this program from the current directory using

```
[abc1234/cse1310] $ ./a.out
I am 42 years old.
```

My preference when compiling is to choose the name of the executable program that is produced by using `gcc`'s `-o` option:

```
[abc1234/cse1310] $ gcc oldtest.c -o oldtest
[abc1234/cse1310] $ ls
a.out  numbers.txt  oldtest  oldtest.c  test.c
```

References

- [Fou09] Free Software Foundation. GNU Coreutils, retrieved August 29, 2009.
- [NR98] Cameron Newham and Bill Rosenblatt. *Learning the bash Shell*. O'Reilly, Sebastopol, CA, 2nd edition, 1998.
- [POL97] Jerry Peek, Tim O'Reilly, and Mike Looukides. *Unix Power Tools*. O'Reilly, Sebastopol, CA, 2nd edition, 1997.
- [RB05] Arnold Robbins and Nelson H.F. Beebe. *Classic Shell Scripting*. O'Reilly, Sebastopol, CA, 2005.