

UNIX / LINUX - FILE MANAGEMENT

<http://www.tutorialspoint.com/unix/unix-file-management.htm>

Copyright © tutorialspoint.com

Advertisements

In this chapter, we will discuss in detail about file management in Unix. All data in Unix is organized into files. All files are organized into directories. These directories are organized into a tree-like structure called the filesystem.

When you work with Unix, one way or another, you spend most of your time working with files. This tutorial will help you understand how to create and remove files, copy and rename them, create links to them, etc.

In Unix, there are three basic types of files –

- **Ordinary Files** – An ordinary file is a file on the system that contains data, text, or program instructions. In this tutorial, you look at working with ordinary files.
- **Directories** – Directories store both special and ordinary files. For users familiar with Windows or Mac OS, Unix directories are equivalent to folders.
- **Special Files** – Some special files provide access to hardware such as hard drives, CD-ROM drives, modems, and Ethernet adapters. Other special files are similar to aliases or shortcuts and enable you to access a single file using different names.

Listing Files

To list the files and directories stored in the current directory, use the following command –

```
$ls
```

Here is the sample output of the above command –

```
$ls
bin          hosts  lib     res.03
ch07        hw1    pub     test_results
ch07.bak    hw2    res.01  users
docs        hw3    res.02  work
```

The command `ls` supports the `-l` option which would help you to get more information about the listed files –

```
$ls -l
total 1962188

drwxrwxr-x  2 amrood amrood      4096 Dec  25  09:59  uml
-rw-rw-r--  1 amrood amrood      5341 Dec  25  08:38  uml.jpg
drwxr-xr-x  2 amrood amrood      4096 Feb  15  2006  univ
drwxr-xr-x  2 root   root        4096 Dec  9   2007  urlspedia
-rw-r--r--  1 root   root       276480 Dec  9   2007  urlspedia.tar
drwxr-xr-x  8 root   root        4096 Nov  25  2007  usr
drwxr-xr-x  2    200    300      4096 Nov  25  2007  webthumb-1.01
-rwxr-xr-x  1 root   root        3192 Nov  25  2007  webthumb.php
-rw-rw-r--  1 amrood amrood    20480 Nov  25  2007  webthumb.tar
-rw-rw-r--  1 amrood amrood      5654 Aug  9   2007  yourfile.mid
-rw-rw-r--  1 amrood amrood   166255 Aug  9   2007  yourfile.swf
drwxr-xr-x 11 amrood amrood      4096 May  29  2007  zlib-1.2.3
$
```

Here is the information about all the listed columns –

- **First Column** – Represents the file type and the permission given on the file. Below is the description of all type

of files.

- **Second Column** – Represents the number of memory blocks taken by the file or directory.
- **Third Column** – Represents the owner of the file. This is the Unix user who created this file.
- **Fourth Column** – Represents the group of the owner. Every Unix user will have an associated group.
- **Fifth Column** – Represents the file size in bytes.
- **Sixth Column** – Represents the date and the time when this file was created or modified for the last time.
- **Seventh Column** – Represents the file or the directory name.

In the **ls -l** listing example, every file line begins with a **d**, **-**, or **l**. These characters indicate the type of the file that's listed.

Sr.No.	Prefix & Description
1	- Regular file, such as an ASCII text file, binary executable, or hard link.
2	b Block special file. Block input/output device file such as a physical hard drive.
3	c Character special file. Raw input/output device file such as a physical hard drive.
4	d Directory file that contains a listing of other files and directories.
5	l Symbolic link file. Links on any regular file.
6	p Named pipe. A mechanism for interprocess communications.
7	s Socket used for interprocess communication.

Metacharacters

Metacharacters have a special meaning in Unix. For example, * and ? are metacharacters. We use * to match 0 or more characters, a question mark (?) matches with a single character.

For Example –

```
$ls ch*.doc
```

Displays all the files, the names of which start with **ch** and end with **.doc** –

```
ch01-1.doc  ch010.doc  ch02.doc  ch03-2.doc
ch04-1.doc  ch040.doc  ch05.doc  ch06-2.doc
ch01-2.doc  ch02-1.doc  c
```

Here, * works as meta character which matches with any character. If you want to display all the files ending with just **.doc**, then you can use the following command –

```
$ls *.doc
```

Hidden Files

An invisible file is one, the first character of which is the dot or the period character (.). Unix programs (including the shell) use most of these files to store configuration information.

Some common examples of the hidden files include the files –

- **.profile** – The Bourne shell (sh) initialization script
- **.kshrc** – The Korn shell (ksh) initialization script
- **.cshrc** – The C shell (csh) initialization script
- **.rhosts** – The remote shell configuration file

To list the invisible files, specify the **-a** option to **ls** –

```
$ ls -a
.          .profile  docs      lib       test_results
..         .rhosts   hosts     pub       users
.emacs    bin       hw1       res.01    work
.exrc     ch07     hw2       res.02
.kshrc    ch07.bak hw3       res.03
$
```

- **Single dot (.)** – This represents the current directory.
- **Double dot (..)** – This represents the parent directory.

Creating Files

You can use the **vi** editor to create ordinary files on any Unix system. You simply need to give the following command –

```
$ vi filename
```

The above command will open a file with the given filename. Now, press the key **i** to come into the edit mode. Once you are in the edit mode, you can start writing your content in the file as in the following program –

```
This is unix file....I created it for the first time.....
I'm going to save this content in this file.
```

Once you are done with the program, follow these steps –

- Press the key **esc** to come out of the edit mode.
- Press two keys **Shift + ZZ** together to come out of the file completely.

You will now have a file created with **filename** in the current directory.

```
$ vi filename
$
```

Editing Files

You can edit an existing file using the **vi** editor. We will discuss in short how to open an existing file –

```
$ vi filename
```

Once the file is opened, you can come in the edit mode by pressing the key **i** and then you can proceed by editing the file. If you want to move here and there inside a file, then first you need to come out of the edit mode by pressing the key **Esc**. After this, you can use the following keys to move inside a file –

- **l** key to move to the right side.
- **h** key to move to the left side.
- **k** key to move upside in the file.
- **j** key to move downside in the file.

So using the above keys, you can position your cursor wherever you want to edit. Once you are positioned, then you can use the **i** key to come in the edit mode. Once you are done with the editing in your file, press **Esc** and finally two keys **Shift + ZZ** together to come out of the file completely.

Display Content of a File

You can use the **cat** command to see the content of a file. Following is a simple example to see the content of the above created file –

```
$ cat filename
This is unix file....I created it for the first time.....
I'm going to save this content in this file.
$
```

You can display the line numbers by using the **-b** option along with the **cat** command as follows –

```
$ cat -b filename
1 This is unix file....I created it for the first time.....
2 I'm going to save this content in this file.
$
```

Counting Words in a File

You can use the **wc** command to get a count of the total number of lines, words, and characters contained in a file. Following is a simple example to see the information about the file created above –

```
$ wc filename
2 19 103 filename
$
```

Here is the detail of all the four columns –

- **First Column** – Represents the total number of lines in the file.
- **Second Column** – Represents the total number of words in the file.
- **Third Column** – Represents the total number of bytes in the file. This is the actual size of the file.
- **Fourth Column** – Represents the file name.

You can give multiple files and get information about those files at a time. Following is simple syntax –

```
$ wc filename1 filename2 filename3
```

Copying Files

To make a copy of a file use the **cp** command. The basic syntax of the command is –

```
$ cp source_file destination_file
```

Following is the example to create a copy of the existing file **filename**.

```
$ cp filename copyfile
$
```

You will now find one more file **copyfile** in your current directory. This file will exactly be the same as the original file **filename**.

Renaming Files

To change the name of a file, use the **mv** command. Following is the basic syntax –

```
$ mv old_file new_file
```

The following program will rename the existing file **filename** to **newfile**.

```
$ mv filename newfile
$
```

The **mv** command will move the existing file completely into the new file. In this case, you will find only **newfile** in your current directory.

Deleting Files

To delete an existing file, use the **rm** command. Following is the basic syntax –

```
$ rm filename
```

Caution – A file may contain useful information. It is always recommended to be careful while using this **Delete** command. It is better to use the **-i** option along with **rm** command.

Following is the example which shows how to completely remove the existing file **filename**.

```
$ rm filename
$
```

You can remove multiple files at a time with the command given below –

```
$ rm filename1 filename2 filename3
$
```

Standard Unix Streams

Under normal circumstances, every Unix program has three streams (files) opened for it when it starts up –

- **stdin** – This is referred to as the *standard input* and the associated file descriptor is 0. This is also represented as STDIN. The Unix program will read the default input from STDIN.
- **stdout** – This is referred to as the *standard output* and the associated file descriptor is 1. This is also represented as STDOUT. The Unix program will write the default output at STDOUT
- **stderr** – This is referred to as the *standard error* and the associated file descriptor is 2. This is also represented as STDERR. The Unix program will write all the error messages at STDERR.