File Management in Unix / Linux

All data in Unix is ??organized in files. All files are organized in folders. These directories are organized in a tree structure that is called the file system.

All data in Unix is ??organized in files. All files are organized in folders. These directories are organized in a tree structure that is called the file system.

When you work with Unix, one way or another, you spend most of your time working with files. This chapter will show you how to create and move files, copy and rename them, create links for them .

In Unix, there are three basic file types:

1. **Ordinary File** : A normal file, a file on the system that stores data, text, or program instructions. In this chapter, you will be taught how to work with these files.
2. **Directory** : Folders, store both regular and special files. Similar to folders in Windows, Mac OS, folders in Unix are folders.
3. **Special File** : A special file, some special files that provide access to hardware such as hard drives, CD-ROMs, modems and Ethernet readers. Some other special files are similar to shortcuts and make it possible to access a particular file using different names.

List files in Unix / Linux

To list files and folders stored in the current directory, use the following command:

```
$ ls
```

Below is an example of the result of the above command.

```
$ ls bin hosts lib res .  03 ch07 hw1 pub test_results ch07 .  bak hw2 res .  (
```

The **ls** command supports the **-l** option to help you get more information about the listed files.

```
$ ls - l 1962188 drwxrwxr - x 2 amrood amrood 4096 Dec 25 09 : 59 uml - rw - rw
```

Below is the information on all the listed columns:

1. The first column: represents the file type and permission provided for the file. Below is a description of the file types.
2. Column 2: represents the amount of memory that the file or directory occupies.
3. Column 3: represents the ownership of the file. That is the Unix user who created this file.
4. Column 4: representation of the owner group. Each Unix user has an affiliate group.

5. Column 5: represents capacity in bytes of the file.
6. Column 6: represents the date that the file was last created or edited.
7. Column 7: represents file or directory name.

In the ls - l example, each file line starts with a character d, -, or l. These characters indicate the type of file listed.

Prefix Description **-** Common file such as an ASCII text file, binary or hard link. **b** Special file group. Group input / output device files such as a physical hard disk. **c** Special character file. Raw input / output device file such as a physical hard disk. **d** A directory that contains a list of other files and directories. **l** File linking characters. Connect on any regular file. **p** Pipe named. A structure for inter-process communication. **s** Socket is used for inter-process communication.

Super characters in Unix / Linux

Super characters are important in Unix. Example * and? is super characters. We use * to match 0 or more characters, use? to connect to a single character.

For example:

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

Displays all the files whose name starts with ch characters and ends with .doc.

```
ch01 - 1.doc ch010 .  doc ch02 .  doc ch03 - 2.doc ch04 - 1.doc ch040 .  doc ch
```

Here the * character is matched to any character. If you want to display all files ending with .doc, then use the following command:

```
$ ls *.  doc
```

Hidden files in Unix / Linux

An invisible file is a file whose first characters are dots (.). Unix programs (including shells) use most of these files to keep configuration information.

Some examples of hidden files include:

.profile : Bourne shell initialization script (sh)

.kshrc : is the Korn shell initialization script (ksh)

.cshrc : is the C shell initialization script (csh)

.hosts : is the file that configures the remote control shell.

To list invisible files (hidden), specify the -a option for the ls command:

```
$ ls - a .  .  profile docs lib test_results .  rhosts hosts pub users .  emacs
```

Single dot (.): Represents the current directory

Double dots (.): represent the root directory.

Create files in Unix / Linux

You can use the **micro** editor to create files commonly named **filename** on Unix systems. You simply need to use the following command:

```
$ vi filename
```

The above command will open a file with the name provided. You will need to press the i key to enter edit mode. Once you are in edit mode, you can start writing the content to the file as shown below:

```
  ?ây là t?p tin không h?p l? . I t?o nó cho th?i gian ??
u tiên . I 'm going to save này n?i dung trong này t?p tin.
```

When you're done, follow these steps:

Press **esc** to exit this mode

Press **Shift + ZZ** key combination to exit the file completely.

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

```
$ vi filename $
```

Editing files in Unix / Linux

You can edit existing files using the vi editor. We will go into each section in a specific tutorial. But shortened here, you can open an existing file as follows:

```
$ vi filename
```

Once this file is open, you can enter editing mode by pressing the **i** key and then you can edit the content as you like. If you want to move in one place or another inside the file, then you first need to exit edit mode by pressing **esc** , then 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;

K key to move upward in a file;

J key to move downwards in a file

So by using the above keys, you can place the cursor wherever you want to edit. Once you have determined the location, then you can use the **i** key to advance into edit mode. After you're done, press **esc** and finally press **Shift + ZZ** to exit the file completely.

Display the contents of a file in Unix / Linux

You can use the cat command to view the contents of a file. Below is a simple example to observe the content of a created file.

```
  $ cat tên t?p tin này không th? ???c t?p tin . I t?o nó cho th?i gian ??
u tiên . I m ?? l?u t?p tin này trong này t?p tin.  $
```

You can display the number of lines using the **-b** option with the **cat** command as follows:

```
  $ cat - b filename 1 ?ây là unix t?p tin . I t?o nó cho th?i gian ??
u tiên . 2 I 'm s? ?? l?u t?p tin này trong này t?p tin.  $
```

Calculate the number of words in a file in Unix / Linux

You can use the **wc** command to get the line number, the number of words and the number of characters in a file. Below is a simple example to observe the file information created on.

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

Below are details in 4 columns:

1. Column 1: represents the total number of lines in that file;
2. Column 2: represents the total number of words in that file;
3. Column 3: represents the total amount of bytes that the file occupies. This is the actual size of the file.
4. Column 4: represents the file name.

You can simultaneously get information about multiple files at once with the following simple syntax:

```
  $ wc filename1 filename2 filename3
```

Copy files in Unix / Linux

To make a copy of a file, you use the **cp** command. The simple syntax of this command is:

```
  $ cp source_file destination_file
```

In it, **source_file** is the file containing the information you need to copy to **destination_file** . Below is an example to create a copy of an existing **filename** :

```
  $ cp filename copyfile $
```

Now you will find a **copyfile** in your current directory. This file will look exactly like the **filename** .

Rename the file name in Unix / Linux

To change the name of a file, use the mv command. The simple syntax of this command is:

```
  $ mv old_file new_file
```

Here is an example that resets an existing **filename** to **newfile** :

```
$ mv filename newfile $
```

The **mv** command will move all existing files into the new file. So in this situation you will only find a **newfile** only in the current directory.

Delete files in Unix / Linux

To delete an existing file, use the rm command. Its simple syntax is:

```
$ rm filename
```

Note : It can be very dangerous to delete a file because it contains useful information. So you need to be careful while executing this command. It suggests the -i option in parallel with the **rm** command.

Here's an example that completely removes an existing **filename** :

```
$ rm filename $
```

You can remove multiple files at the same class as follows:

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

Standard Unix Streams in Unix / Linux

The details in the Unix program have 3 streams (or files) that are opened for it when it starts:

1. **stdin** : It is as a standard input (standard input) and the link file description element is 0. It is also represented as STDIN. Unix program will read the default input from STDIN.
2. **stdout** : It is like a standard output (standard output) and the link file description element is 1. It is also represented as STDOUT. Unix program will write the default output at STDOUT.
3. **stderr** : It is like a standard error (standard error) and the link file description element is 2. It is also represented as STDERR. The Unix program will write down all the error information at STDERR.

According to Tutorialspoint

Previous post: What is Unix / Linux?

Next article: Managing folders in Unix / Linux