

UNIX / LINUX - DIRECTORY MANAGEMENT

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

Copyright © tutorialspoint.com

Advertisements

In this chapter, we will discuss in detail about directory management in Unix.

A directory is a file the solo job of which is to store the file names and the related information. All the files, whether ordinary, special, or directory, are contained in directories.

Unix uses a hierarchical structure for organizing files and directories. This structure is often referred to as a directory tree. The tree has a single root node, the slash character (/), and all other directories are contained below it.

Home Directory

The directory in which you find yourself when you first login is called your home directory.

You will be doing much of your work in your home directory and subdirectories that you'll be creating to organize your files.

You can go in your home directory anytime using the following command –

```
$cd ~  
$
```

Here ~ indicates the home directory. Suppose you have to go in any other user's home directory, use the following command –

```
$cd ~username  
$
```

To go in your last directory, you can use the following command –

```
$cd -  
$
```

Absolute/Relative Pathnames

Directories are arranged in a hierarchy with root (/) at the top. The position of any file within the hierarchy is described by its pathname.

Elements of a pathname are separated by a /. A pathname is absolute, if it is described in relation to root, thus absolute pathnames always begin with a /.

Following are some examples of absolute filenames.

```
/etc/passwd  
/users/sjones/chem/notes  
/dev/rdisk/Os3
```

A pathname can also be relative to your current working directory. Relative pathnames never begin with /. Relative to user amrood's home directory, some pathnames might look like this –

```
chem/notes  
personal/res
```

To determine where you are within the filesystem hierarchy at any time, enter the command **pwd** to print the current working directory –

```
$pwd  
/user0/home/amrood
```

```
$
```

Listing Directories

To list the files in a directory, you can use the following syntax –

```
$ls dirname
```

Following is the example to list all the files contained in **/usr/local** directory –

```
$ls /usr/local
```

X11	bin	gimp	jikes	sbin
ace	doc	include	lib	share
atalk	etc	info	man	ami

Creating Directories

We will now understand how to create directories. Directories are created by the following command –

```
$mkdir dirname
```

Here, directory is the absolute or relative pathname of the directory you want to create. For example, the command –

```
$mkdir mydir  
$
```

Creates the directory **mydir** in the current directory. Here is another example –

```
$mkdir /tmp/test-dir  
$
```

This command creates the directory **test-dir** in the **/tmp** directory. The **mkdir** command produces no output if it successfully creates the requested directory.

If you give more than one directory on the command line, **mkdir** creates each of the directories. For example, –

```
$mkdir docs pub  
$
```

Creates the directories docs and pub under the current directory.

Creating Parent Directories

We will now understand how to create parent directories. Sometimes when you want to create a directory, its parent directory or directories might not exist. In this case, **mkdir** issues an error message as follows –

```
$mkdir /tmp/amrood/test  
mkdir: Failed to make directory "/tmp/amrood/test";  
No such file or directory  
$
```

In such cases, you can specify the **-p** option to the **mkdir** command. It creates all the necessary directories for you. For example –

```
$mkdir -p /tmp/amrood/test  
$
```

The above command creates all the required parent directories.

Removing Directories

Directories can be deleted using the **rmdir** command as follows –

```
$rmdir dirname
$
```

Note – To remove a directory, make sure it is empty which means there should not be any file or sub-directory inside this directory.

You can remove multiple directories at a time as follows –

```
$rmdir dirname1 dirname2 dirname3
$
```

The above command removes the directories `dirname1`, `dirname2`, and `dirname3`, if they are empty. The **rmdir** command produces no output if it is successful.

Changing Directories

You can use the **cd** command to do more than just change to a home directory. You can use it to change to any directory by specifying a valid absolute or relative path. The syntax is as given below –

```
$cd dirname
$
```

Here, **dirname** is the name of the directory that you want to change to. For example, the command –

```
$cd /usr/local/bin
$
```

Changes to the directory **/usr/local/bin**. From this directory, you can **cd** to the directory **/usr/home/amrood** using the following relative path –

```
$cd ../../home/amrood
$
```

Renaming Directories

The **mv** (**move**) command can also be used to rename a directory. The syntax is as follows –

```
$mv olddir newdir
$
```

You can rename a directory **mydir** to **yourdir** as follows –

```
$mv mydir yourdir
$
```

The directories . (dot) and .. (dot dot)

The **filename .** (dot) represents the current working directory; and the **filename ..** (dot dot) represents the directory one level above the current working directory, often referred to as the parent directory.

If we enter the command to show a listing of the current working directories/files and use the **-a option** to list all the files and the **-l option** to provide the long listing, we will receive the following result.

```
$ls -la
drwxrwxr-x   4  teacher  class   2048   Jul 16 17:56 .
drwxr-xr-x  60    root    1536   Jul 13 14:18 ..
-----    1  teacher  class   4210  May 1 08:27 .profile
-rwxr-xr-x   1  teacher  class   1948  May 12 13:42 memo
$
```