hitbullseye

Linux File Types, find, locate, Changing Password, rm, mv, mkdir, rmdir

This lesson describes the various file formats in Linux. Learn about the file types supported by Linux and how to identify a file type in Linux. A file type assists us in determining the type of material recorded in the file. Linux supports seven distinct file formats. Regular files, Directory files, Link files, Character special files, Block special files, Socket files, and Named pipe files are examples of these file formats. The table below gives a brief summary of different file formats.

Everything in Linux is treated as a file. To handle files properly in Linux/UNIX, we must deal with a variety of file kinds.

Files in Linux/UNIX are divided into three categories:

1. Standard Files

2. Files in Directories

3. Special Records

Regular files and directories are the most common and evident file formats. However, the Linux operating system has additional file types to offer, since it also provides another 5 file types. This brief lecture will teach you how to identify all seven file types included in the Linux operating system.

Identifying Linux File types

There is only 1 command you need to know, which will help you to identify and categorize all the seven different file types found on the Linux system.

$ ls -ld <file name>

Here is an example output of the above command.

$ ls -ld /etc/services

-rw-r--r-- 1 root root 19281 Jan 10, 2023,/etc/services

The file type will be shown as an encoded symbol detected as the first character of the file permission portion by the ls command. "-" signifies "normal file" in this situation. It is critical to note that Linux file types should not be confused with file extensions. Let's take a quick look at the seven distinct sorts of Linux file types and ls command identifiers:

1. **–** : regular file
2. **d** : directory
3. **c** : character device file
4. **b** : block device file
5. **s** : local socket file
6. **p** : named pipe
7. **l** : symbolic link

Regular files and directories are the most common and evident file formats. However, the Linux operating system has additional file types to offer, since it also provides another 5 file types. This brief lecture will teach you how to recognize all seven file formats included in the Linux operating system.

Identifying Linux File Extensions

You just need to know one command to identify and categorize all of the seven distinct file types present on the Linux system.

$ ls -ld <file name>

Here is an example output of the above command.

$ ls -ld /etc/services

-rw-r--r-- 1 root root 19281 Feb 14 2022 /etc/services

ls command will show the file type as an encoded symbol found as the first character of the file permission part. In this case it is "-", which means "regular file". It is important to point out that Linux file types are not to be mistaken with file extensions. Let us have a look at a short summary of all the seven different types of Linux file types and **ls** command identifiers:

1. **–** : regular file
2. **d** : directory
3. **c** : character device file

4. **b** : block device file
5. **s** : local socket file
6. **p** : named pipe
7. **l** : symbolic link

Locate vs find: What is the difference?

Find and locate are two typical Linux commands for locating files. The find command might take a long time to scan all of the data depending on the size of your file system and the depth of your search. For instance, if you search your whole file system for files with the name data.txt:

```
# find / -name data.txt
```

More likely than not, this will take on the order of minutes, if not longer to return. A quicker method is to use the locate command:

```
# locate data.txt
```

However, this efficiency comes at a cost, the data reported in the output of locate isn't as fresh as the data reported by the find command. By default, the system will run **updatedb** which takes a snapshot of the system files once a day, locate uses this snapshot to quickly report what files are where. However, recent file additions or removals (within 24 hours) are not recorded in the snapshot and are unknown to locate.

hitbullseye

find Command Examples:

1. Search your present working directory and its subdirectories for a particular file:
   - **$ find . -name "example.txt"**
2. **F**ind all **.png** image files in the **/home** directory and its subdirectories:
   - **$ find /home -name "*.png"**
3. Consider using the **type -f** option with **find** to only search for files (ignore directories), and the **-iname** option to make your search case insensitive:
   - **$ find /home -type f -iname "example.txt"**
4. Find all **.conf** files that have been modified in the last seven days, are owned by user **linuxconfig**, and exist in that user's home directory:
   - **$ find /home/linuxconfig -type f -user linuxconfig -mtime -7 -name "*.conf"**
5. If you don't want the **find** command to traverse too deeply into subdirectories, you can specify a limit with the **-maxdepth** option. For example, this command will limit **find** to a depth of two subdirectories:
   - **$ find . -type f -maxdepth 2 -name "example.txt"**

6. The **find** command can automatically delete files it finds if you specify the **-delete** option. Be very careful with this option, and be sure to first run the find command without it so you know exactly what it plans to delete.

   - **$ find . -type f -name "*.tmp" -delete**

The locate Command

The locate command performs a quick search for any specified string in file names and paths stored in the mlocate database. This database must be updated regularly for the search to be effective. The results displayed may be restricted to files that users have permission to access or execute.

- **Syntax**

The syntax of the locate command is:

```
# locate [options] {string}
```

locate Command Options:

The locate command supports different options that enable you to make your search more effective. Some of the options are described in the table.

Option	Used To
-r	Search for file names using regular expressions.
-c	Display only the number of matching entries found, rather than the file names.
-e	Return only files that exist at the time of search.
-i	Ignore the casing in file names or paths.
-n {number of entries}	Return only the first few matches up to the specified number.

locate Command Examples:

1. To locate any file:

```
# locate file.txt
```

2. To match only the basename against the pattern:

```
# locate -b file.txt
# locate --basename file.txt
```

3. To get the counts for matching entries:

```
# locate -c file.txt
# locate --count file.txt
```

4. To replace the default database with given:

```
# locate -d
# locate --database
```

5. To print the entries those exist at the time when locate was fired:

```
# locate -e filename
# locate --existing filename
```

6. To follow trailing symbolix links:

```
# locate -L text
# locate --follow text
```

7. To get the locate help:

```
# locate -h
# locate --help
```

8. To exit successfully after finding a specified number of entries:

```
# locate -l 10 text
# locate -n 10 text
# locate --limit 10 text
```

9. To avoid following symbolic links:

```
# locate -P text
# locate --nofollow text
```

```
# locate -H text
```

10. To separate the output entries by ASCI NULL character:

```
# locate -0 text
# locate --null text
```

11. To get the statistics about the read database:

```
# locate -S text
# locate --statistics text
```

12. To suppress any errors if occurred:

```
# locate -q text
# locate --quiet text
```

13. To get the version info:

```
# locate -V
# locate --version
```

14. To match the whole pathname:

```
# locate -w text
# locate --wholename text
```

Linux Set User Password:

In Linux, you can change the password of a user account with the passwd utility.

The encrypted users' passwords, as well as other passwords related information, are stored in the **/etc/shadow** file.

As a regular user, you can only change your own password. The root user and users with sudo privileges can change another user's passwords and define how the password can be used or changed. When changing the password, make sure you're using a strong and unique password.

For security reasons, it is recommended to update your password on a regular basis and use a unique password for each account.

Change Your User Password

To change your own user's account password, run the passwd command without any arguments:

passwdCopy

```
Changing password for linuxize.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

You will be prompted to enter your current password. If the password is correct, the command will ask you to enter and confirm the new password. Passwords are not shown on the screen when you enter them. The next time you log in to your system, use the new password.

Change Another User's Password: As we mentioned in the introduction, only the root user and users with sudo access can change the password of another user account. The following example assumes that you are logged in as a user with Sudo privileges. To change the password of another user account, run the passwd command, followed by the username. For example, to change the password of a user named linuxize, run the following command:

```
sudo passwd linuxize
```

You will be prompted to enter and confirm the new password:

Enter new UNIX password:

Retype new UNIX password:

On success, the command will print something like this:

passwd: password updated successfully

rm, mv, mkdir, rmdir -

rm: Linux command:

rm removes each specified file. By default, it does not remove directories.

If the *-I* or *--interactive=once* option is given, and there are more than three files or the *-r*, *-R*, or *--recursive* are given, then **rm** prompts the user for whether to proceed with the entire operation. If the response is not affirmative, the entire command is aborted.

EXAMPLES:

Ex1: Remove the file myfile.txt. If the file is write-protected, you will be prompted to confirm that you really want to delete it:

$ rm myfile.txt

Ex2: Remove the file myfile.txt. You will not be prompted, even if the file is write-protected; if rm can delete the file, it will:

$ rm -f myfile.txt

Ex3: Remove all files in the working directory. If it is write-protected, you will be prompted before rm removes it:

$ rm *

Ex4: Remove all files in the working directory. rm will not prompt you for any reason before deleting them:

$ rm -f *

Ex5: Attempt to remove every file in the working directory, but prompt before each file to confirm:

$ rm -i *

Ex6: Remove every file in the working directory; prompt for confirmation if more than three files are being deleted:

$ rm -I *

Ex7: Remove the directory mydirectory, and any files and directories it contains. If a file or directory that rm tries to delete is write-protected, you will be prompted to make sure that you really want to delete it:

$ rm -r mydirectory

mv : Linux command:

mv command is used to move files and directories.

mv command syntax

```
$ mv [options] source dest
```

mv command options

mv command main options:

option	description
mv -f	force move by overwriting destination file without prompt
mv -i	interactive prompt before overwrite
mv -u	update - move when the source is newer than the destination
mv -v	verbose - print source and destination files
man mv	help manual

mv command Examples:

Move *main.c def.h* files to */home/usr/rapid/* directory:

```
$ mv main.c def.h /home/usr/rapid/
```

Move all C files in current directory to subdirectory *bak* :

```
$ mv *.c bak
```

Move all files in subdirectory *bak* to current directory :

```
$ mv bak/* .
```

Rename file *main.c* to *main.bak*:

```
$ mv main.c main.bak
```

Rename directory *bak* to *bak2*:

```
$ mv bak bak2
```

Update - move when *main.c* is newer:

```
$ mv -u main.c bak
$
```

Move *main.c* and prompt before overwrite *bak/main.c*:

```
$ mv -v main.c bak
'bak/main.c' -> 'bak/main.c'
$
```

mkdir: Linux command:

The command mkdir stands for “make directory”. It creates each directory specified on the command line in the order given. It reports an error if DIRECTORY already exists unless the -p option is given. The mkdir command in Linux/Unix allows users to create or make new directories (also referred to as folders in some operating systems).

This command creates the DIRECTORY(ies), if they do not already exist. This command can create multiple directories at once as well as set the permissions for the directories (folders).

Ex1: Create a simple directory at the current folder /directory

```
$ mkdir book_titles
```

output:

```
$ ls
book_titles
```

Ex2: Create the directory at Home

```
$ mkdir ~/examples
```

output:

```
$ cd ~
$ ls
examples
```

Ex3: Create the directory at the desired location

```
$ mkdir /tmp/examples
```

output:

```
$ cd /tmp
$ ls
examples
```

rmdir: Linux command:

As the name suggests, the **rmdir** command is used to remove the directory. However, it is important to note that it can remove empty directories only. In this section, we will see the basic usage of the **rmdir** command.

Delete an Empty Directory in Linux

First, create a few empty directories:

```
$ mkdir dir1 dir2 dir3 dir4
$ mkdir dir1 dir2 dir3 dir4
```

Let's verify that the required directories have been created:

```
$ ls -l
```

Now, let's remove the **dir1** directory and verify that it has been removed:

```
$ rmdir dir1
$ ls -l
```

In a similar fashion, we can use the **rmdir** command to remove multiple empty directories at once. Let's remove the remaining directories:

```
$ rmdir dir2 dir3 dir4
```