

Linux text editing

- Vi
- Emacs
- The grep utility
- The tr utility

The cut uti Linux diff Command

Lets understand Linux diff command through some practical examples.

Suppose we have these two files (file1 and file2) :

```
$ cat file1
Hi,
Hello,
How are you?
I am fine,
Thank you.
```

```
$ cat file2
Hello,
Hi,
How are you?
I am fine.
```

You can see that both the files have some minor differences. Now, lets see how the diff command can be used to bring out these differences.

Here is how you run diff command :

```
$ diff file1 file2
1d0
< Hi, 2a2 > Hi,
4,5c4
< I am fine,
< Thank you.

---
> I am fine.
```

So you can see that the diff command was provided with both the file names as command line arguments and it produced the differences in the output. The output by no means is easily comprehend-able. Reason being, it was designed to be used by the computers and not the humans. Nonetheless, lets decode the output piece by piece:

NOTE - The files file1 and file2 will be addressed as old file and new file in the following text.

```
1d0
< Hi,
```

Here, the line 1d0 means that the 1st line of the old file should be deleted (d) in order to sync up both the files beginning at line 0. The line to be deleted in old file is mentioned just below with < mark.

```
2a2
> Hi,
```

Here, the line 2a2 means that the second line from the new file should be added after second line of old file. The line to be added is displayed in the next line of the output with > mark.

```
4,5c4
< I am fine,
< Thank you.
```

```
---
```

```
> I am fine.
```

Here, the line 4,5c4 means that the line numbers ranging from 4 to 5 in the old file are now changed and should be replaced with the 4th line of the new file. The lines to be added and deleted are marked with > and < respectively.

So, to conclude,

- The first argument to diff command is regarded as old file while the second argument becomes new file.
- Expressions like 1d0 2a2, 4,5c4 can be decoded with the syntax **[line number or range from old file][action][line number or range from new file]**. Where, 'action' can be append, delete or changed-so-replace.
- The mark < represents the line to be deleted while > represents the line to be added.

In addition to files, the diff command can also be used to compare two directories. Lets learn this through an example.

Here are the contents of a directory named 'new_dir' :

```
$ ls new_dir/
file file2 frnd frnd1.cpp log1.txt log3.txt log5.txt
file1 file name with spaces frnd1 frnd.cpp log2.txt log4.txt
```

and here are the contents of a directory named 'old_dir' :

```
$ ls orig_dir/  
file1 file name with spaces frnd1 frnd.cpp log2.txt log4.txt test  
file2 frnd frnd1.cpp log1.txt log3.txt log5.txt
```

Now, here is the output when diff command was executed :

```
$ diff new_dir/ orig_dir/  
Only in new_dir/: file  
Only in orig_dir/: test
```

So you can see that when diff command is executed to compare these two directories, it easily displays the missing files in the directories.

Here are some of the commonly used command line options :

1. Use -i to ignore case differences

If two files contain same text but in different cases, the diff command still reports it as a difference by default.

For example :

```
$ cat file1  
HI  
  
$ cat file2  
hi  
  
$ diff file1 file2  
1c1  
< HI --- > hi
```

So you can see that the diff command reported the case difference in the output.

To get away with this default behaviour, use -i option.

Here is an example :

```
$ diff -i file1 file2  
$
```

So you can see that no output was produced, which is the default behaviour when both the files are same.

2. Report That The Files Are Same Using -s Option

Towards the end of the example 1, we saw that diff doesn't report anything if files are same. Though this default behaviour is fine but it could still confuse many, especially the newbies.

So, if you want the diff command to report explicitly that the files are same then use -s command line option.

Here is an example :

```
$ diff -is file1 file2
Files file1 and file2 are identical
```

So you can see that I added the -s option in the command that we used in last example and this time the diff command explicitly reported that both the files are identical.

3. Use -b To Ignore Spaces

Another common use case is the varying amount of spaces for which the diff command reports the files as different.

Here is an example :

```
$ cat file1
Hi, how are you?

$ cat file2
Hi, how are  you?
```

Observe that the only difference between the tow files is the extra space between the words 'are' and 'you' in file2. Now, when diff command is used to compare these two files, here is the output :

```
$ diff file1 file2
1c1
< Hi, how are you?

---
> Hi, how are  you?
```

So you can see that the diff command reported the difference. But if you want diff to ignore these spaces, use -b option :

```
$ diff -b file1 file2
$
```

So you can see that due to -b option, the diff command reported both the files as same.

The diff command provides many more command line options. Read it's [man page](#) for the complete list.

- lity
- The past utility

- The sort utility
- The wc utility
- The diff utility
- The lp utility
- The sed utility

Network administration

- **Network administration**
 - o Configuration UNIX/Linux network
- File and directory administration
- The chmod utility
- The umask command
- The find utility
- The df utility
- The du utility
- The ulimit utility
- The tar utility
- File compression and backup
- Device administration and system accounting

Shell programming and perl AWK

- Shell variables
- Regular expressions
- Branching and jumping
- Subroutines
- Awk