Tutorial Link https://course.testpad.chitkara.edu.in/tutorials/Linux - Comparing Files/60dc5ec4ccde02f8205963b3

TUTORIAL

Linux - Comparing Files

Topics

File Comparison

While working in linux, sometimes it is necessary to compare files. Comparing files means finding the similarities and differences between these files.Linux provides command line utilities to compare files. Some of these are

1. **diff command**
2. **comm command**
3. **cmp command**

diff command

Comparing files with diff

diff is used to compare files(line by line) and directories. This often-used utility program has many useful options (see: man diff) including:

```
diff Option     Usage
-c         Provides a listing of differences that include three
lines of context before and after the lines differing in content
-r          Used to recursively compare subdirectories, as well as
the current directory
```

```
-i         Ignore the case of letters
-w         Ignore differences in spaces and tabs (white space)
-q         Be quiet: only report if files are different without
listing the differences
```

To compare two files, at the command prompt, type diff [options] <filename1> <filename2>. diff is meant to be used for text files; for binary files, one can use cmp.

Using diff3 and patch

You can compare three files at once using diff3, which uses one file as the reference basis for the other two. For example, suppose you and a co-worker both have made modifications to the same file working at the same time independently. diff3 can show the differences based on the common file you both started with. The syntax for diff3 is as follows:

```
$ diff3 MY-FILE COMMON-FILE YOUR-FILE
$ cat file1
This is file one
$ cat file2
This is file two
$ cat file3
This is file three
$ diff3 file1 file2 file3
```

Many modifications to source code and configuration files are distributed utilizing patches, which are applied, not surprisingly, with the patch program. A patch file contains the deltas (changes) required to update an older version of a file to the new one. The patch files are actually produced by running diff with the correct options, as in:

```
$ diff -Nur originalfile newfile > patchfile
```

Distributing just the patch is more concise and efficient than distributing the entire file. For example, if only one line needs to change in a file that contains 1000 lines, the patch file will be just a few lines long.

To apply a patch, you can just do either of the two methods below:

```
$ patch -p1 < patchfile
$ patch originalfile patchfile
```

The first usage is more common, as it is often used to apply changes to an entire directory tree, rather than just one file, as in the second example. To understand the use of the -p1 option and many others, see the man page for patch.

```bash
diff file1.txt file2.txt
```

cmp command

cmp command in Linux is used to compare the two files byte by byte and helps to find out whether the two files are identical or not.

- When cmp is used for comparison between two files, it reports the location of the first mismatch to the screen if difference is found and if no difference is found i.e the files compared are identical.
- cmp displays no message and simply returns the prompt if the files compared are identical.

Syntax:

```
cmp [OPTION] FILE1 [FILE2]
```

The syntax of cmp command is quite simple to understand. When comparing two files, we will need their names as arguments (i.e as FILE1 & FILE2 in syntax).

```bash
echo -n "When content in files are different:"
cmp file1.txt file2.txt
echo -n "When content in files are same:"
cmp file1.txt file3.txt
```

comm command

Comm command is used to compare two sorted files line by line and write to standard output: the lines that are common, plus the lines that are unique.

Syntax for comm command is

```
comm options file1 file2
```

```bash
comm file1.txt file2.txt
```

file utility

In Linux, a file's extension often does not categorize it the way it might in other operating systems. One cannot assume that a file named file.txt is a text file and not an executable program. In Linux, a filename is generally more meaningful to the user of the system than the system itself. In fact, most applications directly examine a file's contents to see what kind of object it is rather than relying on an extension. This is very different from the way Windows handles filenames, where a filename ending with .exe, for example, represents an executable binary file.

The real nature of a file can be ascertained by using the file utility. For the file names given as arguments, it examines the contents and certain characteristics to determine whether the files are plain text, shared libraries, executable programs, scripts, or something else.

```
$ ls -F
$ file *
```