

Comparing Files and File Types

Comparing Files with diff:

1. **diff** is used to compare files 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

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

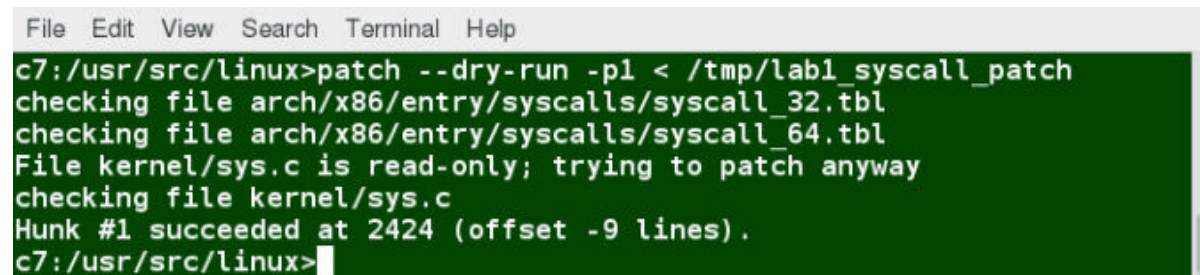
Using diff3 and patch:

1. You can compare three files at once using **diff3**, which uses one file as the reference basis for the other two.
2. 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.
3. The syntax for **diff3** is as follows:
diff3 MY-FILE COMMON-FILE YOUR-FILE

```
File Edit View Search Terminal Help
c7:/tmp>cat file1
Hello
This is file1
c7:/tmp>cat file2
Hello
This is file2
c7:/tmp>cat file3
Hello
This is file3
c7:/tmp>diff3 file1 file2 file3
===
1:2c
  This is file1
2:2c
  This is file2
3:2c
  This is file3
c7:/tmp>
```

Using diff3

4. Many modifications to source code and configuration files are distributed utilizing patches, which are applied, not surprisingly, with the patch program.
5. A patch file contains the deltas (changes) required to update an older version of a file to the new one.
6. The patch files are actually produced by running diff with the correct options, as in:
diff -Nur originalfile newfile > patchfile
7. Distributing just the patch is more concise and efficient than distributing the entire file.
8. 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.



```
File Edit View Search Terminal Help
c7:/usr/src/linux>patch --dry-run -p1 < /tmp/lab1_syscall_patch
checking file arch/x86/entry/syscalls/syscall_32.tbl
checking file arch/x86/entry/syscalls/syscall_64.tbl
File kernel/sys.c is read-only; trying to patch anyway
checking file kernel/sys.c
Hunk #1 succeeded at 2424 (offset -9 lines).
c7:/usr/src/linux>
```

Using patch

9. To apply a patch, you can just do either of the two methods below:
patch -p1 < patchfile
or
patch originalfile patchfile
10. The above 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 above second example.
11. To understand the use of the **-p1** option and many others, see the man page for patch.

Using the file Utility:

1. In Linux, a file's extension often does not categorize it the way it might in other operating systems.
2. One cannot assume that a file named **file.txt** is a text file and not an executable program.
3. In Linux, a filename is generally more meaningful to the user of the system than the system itself.
4. 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.
5. The real nature of a file can be ascertained by using the file utility.
6. 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.