Lecture 16: File Manipulators (contd)

.mv

- $ mv <oldname> <newname>
- $ mv <file1…….filen> <newdir>

The mv command moves the contents of one file to another i.e., it renames an existing file. The files that are being renamed should be ordinary files.

The directory that has to be renamed must have the write and the execute permission. Only the owner of a file or a duly authorised user can rename a file.

The command line format to rename a file or directory is:

```
$ mv <oldname>      <newname>                     <Enter>
```

To "move" one or more files into a specified directory, with their original filenames, the following command line format is used:

```
$ mv <filel [file2 [file3] ].filen> <dirname>          <Enter>
```

Gomilitary.in

Example:

The following files are present in a directory:

appeal
program1
program2
script
wordlist

1. To rename word list as dictionary:

```
$ mv wordlist dictionary          <Enter>
```

2. To move appeal and program1 to Microsoft directory:

```
$ mv appeal program1 Microsoft    <Enter>
```

3. To rename the directory Microsoft to Micros1:

```
$ mv Microsoft Micros1            <Enter>
```

Note: 1. The directory Microsoft is a child of the current directory.
2. The mv command supports the "wild card" characters.

.rm

This command is used to remove (ie. delete or erase) one or more files and / or directory. The user must have the write permission to the file or the directory.

If the user is the owner of the file to be removed and the file has no write permission for the owner, rm lets the user override the write protection. If the user does not own the file then rm will refuse to remove it.

The command line format for the rm command is:

```
$ rm [option] <filename>      <Enter>

$ rm poem2 <Enter>            will remove the file poem2.

$rm  *.bak <Enter>            will remove all files whose last 4 characters are.bak.
```

."-i" option interactively asks for confirming the deletion of the files. It is useful in avoiding accidental erasure of the file.

“-r” II *option provides a convenient way to erase a directory even if it* is *not* empty. It contrasts with rmdir command that can only remove a directory, if that is empty.

"-f" option will forcefully remove a file even if it is write protected.

Examples:

1. Forcefully remove the file poem2:

```
$rm -f        poem2        <Enter>
```

2. Remove the entire directory test2

```
$ rm –r       test2        <Enter>
```

Gomilitary.in

3. The output is produced with rm -i command:

```
$ rm -i *                <Enter>

rm : Microsoft directory
rm : test directory
appeal: n
appea12 : y
poem: n
$
```

Here, since '-r' option was not specified, rm first gives a warning message that **Microsoft** and **test** are directories.

The cursor pauses after the implied query, 'appeal'. If the user's response is 'n', rm will not remove that file and prompt *for* the next file and so on.   .

4. A session with the rm -ri command:

```
$rm     -ri       test                           <Enter>

directory test: y
directory test/subtest: y
test/subtest : y
test/appeal: y
test/poem: y
test : y
```

In the above example two kinds of implied queries have been merged:

It involves the recursive deletion of a directory and the files it contains. Directory test and directory test/subtest are being asked *for* permission so that they can be searched. If the answer is "yes", then the prompt appears to confirm the removal of *files* within the directory. .

The rm command supports the "wild card" characters.

Warning: rm -r_and rm -f are very powerful commands. Do not use them lightheartedly. Those who know DOS note specially that in UINX, deleted files cannot be "undeleted".

Comparing and Contrasting files

comm.	: display common lines in the two files
cmp	: to compare two files
diff	: reports the difference between files
uniq	: display the duplicated and unique lines of a file

It is often useful to compare two files to verify their contents, whether they are the same or different.

For example, a user who has been interrupted while editing a file may have forgotten exactly what changes were made. Comparing the "before editing file" with the "after editing file" gives a quick list of changes.

.comm

```
$ comm [-[1] [2] [3]] file1 file2
```

Gomilitary.in

options

1	Suppresses the display of the first column in the output
2	Suppresses the display of the second column in the output.
3	Suppresses the display of the third column in the output

The comp creates a report containing tines that are unique to each of the files and lines that are common to both files.

The command line format for using comm is:

```
$ comm [- [1] [2] [3]] filel file2       <Enter>
```

where 1,2 and 3 are the flags used to suppress the display of the corre_sponding column.

Example: To compare the files **CustomerA** and **CustomerB**, having the following data and sorted alphabetically:

CustomerA	**CustomerB**
Navneet	**Rachna**
Neeraj	**Ritu**
Ritu	**Neeru**
Jamil	**Joby**
Richa	**Seema**

```
$ comm CustomerA            CustomerB     <Enter>
```

will produce the output:

```
Navneet
Neeraj
Jamil
Richa

                        Ritu

        Rachna
        Neeru
```

```
Joby
Seema
```

where

The first column lists the lines unique to the first file
The second column lists the lines unique to the second file The third column lists the lines common to both the files.

Example: If the files CustomerA and CustomerS have the following data:

```
CustomerA          CustomerB

Navneet            Rachna
Neeraj              Ritu
Jamil               Neeru

Joby
Richa
```

```
$ comm CustomerA CustomerB      <Enter>
```

will produce the following output:

```
Navneet
Neeraj             Rachna
Jamil              Ritu
Joby               Neeru
```

Gomilitary.in

```
        Richa
```

here the third column is left blank since the two files have no common lines.

Option -12, will display only the third column and suppress the first and the second column.

```
        $ comm -12 CustomerA CustomerB            <Enter>
```

No output will be produced for the second example as there are no line common to both the files.

Classroom Exercise

What will the -1 option with comm command will do in the first example?

.cmp

Compare two files

```
$ cmp file1 file2
```

where file1 and file2 are the files to be compared

The cmp command compares the contents of two files, prints the position of first difference.

The command line format for cmp is:

```
$ cmp filel file2        <Enter>
```

Example:To compare the files Document1 and Document2, having the following data:

```
Docwnentl                     Docwnent2

This a document               This is Docwnent
for internal circulation.     for internal circulation.

$ cmp Docwnentl Docwnent2     <Enter>
```

the outputwill be:

```
DocwnentlDocwnent2       differ: char 6, line 1
```

The output indicates that the two files differ on line 1 at the 6th character.

Gomilitary.in

The cmp command indicates only the first character at which the files differ. A blank space is also treated as a character.

If the two files are identical then the cmp command does not give any output.

.diff

Reports the difference between two files

```
$ diff [options...] [file1] [file2]
```

can be used only on ASCII files

options:

-b ignores trailing blanks and other strings

-h used for files of unlimited length

The diff command reports the differences between two text files on a line by line basis. The report is displayed in terms of changes required in the first file so that the two files will not differ.

A point to be noted is that diff itself does not change the file, but merely reports how to make such transformations. It is not possible to compare binary files using diff.

The command line format for using diff is:

```
$ diff [option) filel file2        <Enter>
```

The output will show how file1 can be changed to look like file2

Examples:

To report the differences between the files file1 and file2 having the following data:

Filel	file2
cal	cal
cp	cp
crypt	crypt
z	cut

```
$ diff filel file2        <Enter>

    4c4

    <z
    -----

    >cut

    $
```

Where

4c4 indicates how line 4 of file1 should be changed to look like line 4 of file2.

Number(s) before the character refers to the line in file1 and the after the character refers

Gomilitary.in

to file2

    'a' stands for append

    'c' stands for change

    'd' stands for delete

The '<' belongs to file1

The ">" belongs to file2