File/Directory Operation Utilities

mv

The `mv` or "move" command is used to move files from one directory to another. > Syntax: $ `mv` [source file] [destination file]

```
$ mv /dir_one/file /dir_two/file
```

It is also used to rename files, with or without moving them.

```
$ mv file1 file2
```

cp

The `cp` or "copy" command is used to copy files, in the same or different directory. > Syntax: $ `cp` [source file] [destination file]

```
$ cp /dir_one/og_file /dir_one/copy_file
```

rm

The `rm` or "remove" command is used to delete one or more files.

```
$ rm file1
```

Advanced uses All the above three tools have a same flag, which is so similar in it's action that it is better to cover it once.

The mv, cp, and rm commands have a `-r` flag, which is used to repeat the option recursively for all folders and subfolders, when applied on a directory.

```
$ cp dir1/ dir2/
```

If in the above example, the directory "dir1/" has a lot of folders and those folders have subfolders, they will be also copied(or moved or deleted for the respective other commands), preserving their original file structure.

find

This is one very confusing tool in some ways, and a very useful one in other ways. The easiest use of find is that it will list all the files, folders, subfolders, everything in a given file or folder, which is obviously very useful for searching for a particular item.

```
$ find .
./03 File.md
./README.md
./01 Basic.md
./02 System.md
./00 Miscellaneous.md
```

> By default, `find` also lists the hidden files and folders, the contents of my ".git" folder had to be omitted from this output.

But, the real place where `find` shines is to actually lookup those files for us.

> Syntax: `find` [directory to look in] -name [filename]

```
$ find . -name README*
./README.md
```

file

`file` is a very useful command. It gives basic details about any filename given to it, without having the necessary tools to open the file even.

```
$ file README.md
README.md: exported SGML document, ASCII text, with very long lines

$ file /usr/bin/ls
/usr/bin/ls: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, int
```

head

The `head` tool, by default, reads the first **10** lines of the given files, and prints them to the standard output.

```
$ head file.txt
# Add ten random lines here.
```

The `-n` flag can be used to make it print a different number of lines from the starting of the file.

```
$ head -n 6 file.txt
# Add six same random lines here.
```

tail

The `tail` tool, by default, reads the last **10** lines of the given files, and prints them to the standard output.

```
$ tail file.txt
# Add ten random lines here.
```

The `-n` flag can be used to make it print a different number of lines from the ending of the file.

```
$ tail -n 6 file.txt
# Add six same random lines here.
```

wc

The `wc` tool is used to print the **word count** and more of a file. It can be used to print the number of lines, words, characters, and the number of bytes it's occupying.

```
$ wc data.txt
 5 15 73 data.txt
```

The output is in the following order:

Number of lines : Use the `-l` flag Number of words : Use the `-w` flag Number of characters : Use the `-m` flag

Also, we can print the size of the file in bytes, using the `-c` flag.

```
$ wc -c data.txt
73 data.txt
```

Note that the file size is given in bytes, and is merely co-incidentally equal to the number of characters; and this may not be true for every file.

cmp

The `cmp` tool is a good tool for finding if two files are equal or not. This is faster than the `diff` and `comm` tools because unlike them, it compares the files character by character and terminates the checking as soon as the first difference in the two files is found.

```
$ cmp data.txt data1.txt
data.txt data1.txt differ: byte 1, line 1
```

diff

comm