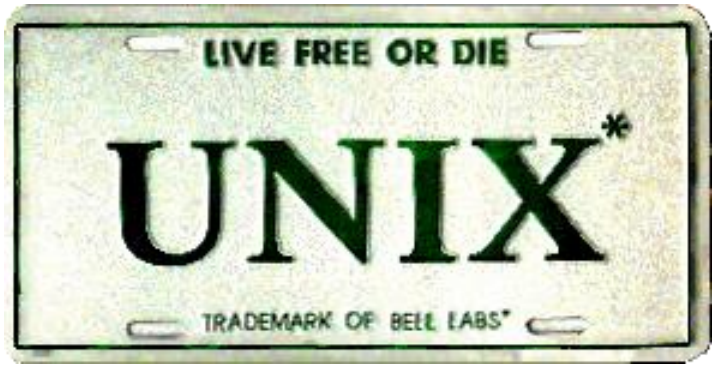

UNIX for Programmers and Users

“UNIX for Programmers and Users”
Third Edition, Prentice-Hall, GRAHAM GLASS, KING ABLES

Renaming/Moving a File: mv

mv -i oldFileName newFileName
mv -i fileName directoryName
mv -i oldDirectoryName newDirectoryName

- The first form of mv renames oldFileName as newFileName.
- The second form allows you to move a collection of files to a directory.
- The third form allows you to move an entire directory.
- The -i option prompts you for confirmation if newFileName already exists so that you do not accidentally replace its contents. You should learn to use this option (or set a convenient shell alias that replaces “mv” with “mv –i”; we will come back to this later).

Renaming/Moving Files: mv

- Here's how to rename the file using the first form of the mv utility:

```
$ mv  heart   heart.ver1                    --> rename to "heart.ver1".
$ ls
heart.ver1
$ _
```

Making Directory: mkdir

mkdir -p newDirectoryName

- The mkdir utility creates a directory. The -p option creates any parent directories in the newDirectoryName pathname that do not already exist.

- If newDirectoryName already exists, an error message is displayed and the existing file is not altered in any way.

```
$ mkdir lyrics                  --> creates a directory called “lyrics”.
$ ls –lF                --> check the directory listing in order
                                --> to confirm the existence of the
                                --> new directory.
-rw-r--r--      1   glass   106   Jan  30  23:28    heart.ver1
drwxr-xr-x      2   glass   512   Jan  30  19:49    lyrics/
$ _
```

Moving Files

- Once the “lyrics” directory is created, we can move the “heart.ver1” file into its new location. To do so, used mv and confirm the operation using ls:

```
$ mv heart.ver1  lyrics  --> move into “lyrics”.

$ ls                                --> list the current directory.
lyrics/                    --> “heart.ver1” has gone.

$ ls lyrics                         --> list the “lyrics” directory.
heart.ver1                          --> “heart.ver1” has moved.
$ _
```

Changing Directories: cd

- **cd directoryName**
- The following might be inconvenient; especially if we deal with large hierarchy:

$cat lyrics/heart.ver1 --> display

- Instead, change directory:

$ cd lyrics --> change directory
$ cat heart.ver1 --> display

- The cd shell command changes a shell's current working directory to be directoryName.
- If the directoryName argument is omitted, the shell is moved to its owner's home directory.

Reorganizing Directories

```
$ pwd                                   --> display where I am
/home/glass

$ cd lyrics                  --> move into the “lyrics” directory
$ pwd
/home/glass/lyrics

$ cd ..                                 --> move up one level
$ pwd                                   --> display new position
/home/glass

$ cd lyrics                  --> move into the “lyrics” directory
$ pwd
/home/glass/lyrics

$ ls ~/                                 --> “~/” refers to home directory
/home/glass
$ _
```

Copying Files: cp

- To copy the file, I used the cp utility, which works as follows:

cp -i oldFileName newFileName
cp -ir fileName directoryName

- The first form of cp copies the contents of oldFileName to newFileName.

- If the label newFileName already exists, its contents are replaced by the contents of oldFileName.

- The -i option prompts you for confirmation if newFileName already exists so that you do not accidentally overwrite its contents. Like with mv, it is a good idea to use this option or create an alias.

Copying Files: cp

- The -r option causes any source files that are directories to be recursively copied, thus copying the entire directory structure.

- cp actually does two things
  - It makes a physical copy of the original file's contents.
  - It creates a new label in the directory hierarchy that points to the copied file.

```
$ cp heart.ver1  heart.ver2          --> copy to "heart.ver2".
$ ls -l heart.ver1  heart.ver2       --> confirm the existence of both files.
-rw-r--r--  1  glass  106  Jan  30  23:28  heart.ver1
-rw-r--r--  1  glass  106  Jan  31  00:12  heart.ver2
$ cp -i heart.ver1  heart.ver2       --> what happens?
```

Deleting a Directory: rmdir

rmdir directoryName

- The rmdir utility removes all of the directories in the list of directory names provided in the command. A directory must be empty before it can be removed.
- To recursively remove a directory and all of its contents, use the rm utility with the -r option.
- Here, we try to remove the “lyrics.draft” directory while it still contains the draft versions, so we receive the following error message:

```
$ rmdir  lyrics.draft
rmdir : lyrics.draft : Directory not empty.
$ _
```

Deleting Directories: rm -r

- The rm utility allows you to remove a file's label from the hierarchy.
- Here's a description of rm:

rm -fir fileName

- The rm utility removes a file's label from the directory hierarchy.
- If the filename doesn't exist, an error message is displayed.
- The -i option prompts the user for confirmation before deleting a filename. It is a very good idea to use this option or create a shell alias that translates from "rm" to "rm –i". If you don't, you will loose some files one day – you have been warned!
- If fileName is a directory, the -r option causes all of its contents, including subdirectories, to be recursively deleted.
- The -f option inhibits all error messages and prompts. It overrides the –i option (also one coming from an alias). This is dangerous!

Removing Directories with Files

- The -r option of rm can be used to delete the “lyrics.draft” directory and all of its contents with just one command:

```
$ cd                        --> move to my home directory.
$ rm  -r   lyrics.draft     --> recursively delete directory.
$ _
```

Counting Lines, Words and Characters in Files: wc

wc -lwc fileName

- The wc utility counts the number of lines, words, and/or characters in a list of files.
- If no files are specified, standard input is used instead.

- The -l option requests a line count,
- the -w option requests a word count,
- and the -c option requests a character count.

- If no options are specified, then all three counts are displayed.
-
- A word is defined by a sequence of characters surrounded by tabs, spaces, or new lines.

Counting Lines, Words and Characters in Files: wc

- **For example, to count lines, words and characters in the "heart.final" file, we used:**

```
$ cd  ~/lyrics.final
$ wc  heart.final   --> obtain a count of the number of lines,
                                --> words, and characters.
9     43    213    heart.final
$ _
```

Determining Type of a File: file

file fileName

- The file utility attempts to describe the contents of the fileName argument(s), including the language in which any of the text is written.
- file is not reliable; it may get confused.
- When file is used on a symbolic-link file, file reports on the file that the link is pointing to, rather than on, the link itself.
- For example,

```
$ file heart.final                    --> determine the file type.
heart.final: ascii text
$ _
```

File Permissions (Security)

•File permissions are the basis for file security. They are given in three clusters. In the example, the permission settings are

"rw-r--r--":

-rw-r--r-- 1 glass cs 213 Jan 31 00:12 heart.final

User (owner)	Group	Others
rw-	r--	r--

clusters

Each cluster of three letters has the same format:

Read permission	Write permission	Execute permission
r	w	x

File Permission

The meaning of the read, write, and execute permissions depends on the type of file:

	Regular file	Directory file
Read	**read the contents**	**read the directory (list the names of files that it contains)**
Write	**change the contents**	**Add files to the directory**
Execute	**execute the file if the file is a program**	**access files in the directory**

Change File Permissions: chmod

chmod -R change fileName

- The chmod utility changes the modes (permissions) of the specified files according to the change parameters, which may take the following forms:

clusterSelection+newPermissions (add permissions)
clusterSelection-newPermissions (subtract permissions)
clusterSelection=newPermissions (assign permissions absolutely)

- where clusterSelection is any combination of:
  - u (user/owner)
  - g (group)
  - o (others)
  - a (all)

  and newPermissions is any combination of
  - r (read)
  - w (write)
  - x (execute)
  - s (set user ID/set group ID)

Changing File Permissions

- Note that changing a directory's permission settings doesn't change the settings of the files that it contains.

- The **-R** option recursively changes the modes of the files in directories.

Ex: Remove read permission from groups

$ ls -lg heart.final --> to view the settings before the change.

```
-rw-r-----   1    glass     music  213  Jan  31  00:12  heart.final
```

$ chmod g-r heart.final

$ ls -lg heart.final

```
-rw-------   1    glass     music  213  Jan  31  00:12  heart.final
```

$ _

Changing File Permissions: examples

Requirement	Change parameters
Add group write permission	g+w
Remove user read and write permission	u-rw
Add execute permission for user, group, and others.	a+x
Give the group read permission only.	g=r
Add write permission for user, and remove group read permission.	u+w,g-r

Changing File Permission: examples

- Example:

```
$ cd              --> change to home directory.
$ ls  -ld   .              --> list attributes of home directory.
drwxr-xr-x  45   glass   4096  Apr   29   14:35
$ chmod   o-rx                  --> update permissions.
$ ls  -ld   .                   --> confirm.
drwxr-x---    45   glass   4096  Apr  29  14:35
$ _
```

Changing File Permissions Using Octal Numbers

- The chmod utility allows to specify the new permission setting of a file as an octal number.
- Each octal digit represents a permission triplet.

For example, for a file to have the permission settings of rwxr-x---

the octal permission setting would be 750, calculated as follows:

	User	Group	Others
setting	rwx	r-x	---
binary	111	101	000
octal	7	5	0

Changing File Permissions Using Octal Numbers

- The octal permission setting would be supplied to chmod as follows:

```
$ chmod  750   .                    --> update permissions.

$ ls  -ld  .                    --> confirm.
drwxr-x---      45   glass     4096    Apr  29 14:35
$ _
```