

Your File Permissions

The output in the figure below is generated if we use the command `'ls -l'` in order to list the contents of a folder. The first character in the first column of the table below tells us what kind of file this is. The dash '-' represents a normal file. The 'd' represents a directory. The remaining characters describe the file's permissions.

Let's take a closer look at the first column. Like we said, the first letter tells you whether you're dealing with a regular file or a directory. The next letters tell the system what access is permitted for this file. hence the name "permissions."

In the listing below, we've eliminated some columns and added spaces to the permissions column to make it easier to read.

FIG. 1

```
- rwx r-xr-x  joe  acctg  archive.sh
- rw- rw-r--  joe  acctg  orgchart.gif
- rw- rw-r--  joe  acctg  personnel.txt
- rw- r--r--  joe  acctg  publicity.html
d rwx r-xr-x  joe  acctg  sales
- rw- r----- joe  acctg  topsecret.inf
- rwx r-xr-x  joe  acctg  wordmatic
```

The first set of three letters after the file type show what you, the owner of the file, have permission to do. An **r** in the first position means you are permitted to read the file. A **w** in the second position means you may write the file. This includes the ability to delete a file. An **x** in the third position means you may execute the file. A hyphen in any position means that you don't have that particular permission.

As you can see above, joe, the user who owns the file, can read and write all the files. He can execute the shell script `archive.sh` and the program `wordmatic`. In addition, when a directory has the `x` set, this takes the special meaning of "permitted to search this directory".

Group Permissions

The next three letters after the user's permission are the group's permissions. Other people in the `acctg` group can not write the `archive.sh` script, `publicity.html` and `topsecret.inf` files, or `wordmatic` program. They may write the `orgchart.gif` and `personnel.txt` files. Also, they may execute `archive.sh` and `wordmatic`, and search the `sales` directory.

Others' Permissions

The last three letters in the permissions column tell us what everyone else in the world, the "others" may do. The "others" lead highly restricted lives. They can't write any files or directories, and they have absolutely no access to the `topsecret.inf` file. They may still run the `archive.sh` script and the `wordmatic` program, and they may search the `sales` directory.

The chmod Command

We use the chmod command to change the access mode of a file. This command comes in many flavors, but we'll be talking primarily about one of them.

chmod who=permissions filename

This gives “**who**” the specified permissions for a given filename.

Who

The “who” is a list of letters that specifies whom you’re going to be giving permissions to. These may be specified in any order.

Letter	Meaning
u	The user who owns the file (this means “you.”)
g	The group the file belongs to.
o	The other users
a	all of the above (an abbreviation for ugo)

Permissions

Of course, the permissions are the same letters that you see in the directory listing:

r	Permission to read the file.
w	Permission to write (or delete) the file.
x	Permission to execute the file, or, in the case of a directory, search it.

Note: Do not put blanks around the equal sign, or your command will not work!

chmod Examples

Let's change some of the permissions. Here's the way our files are now:

```
-rwxr-xr-x  joe  acctg  archive.sh
-rw-rw-r--  joe  acctg  orgchart.gif
-rw-rw-r--  joe  acctg  personnel.txt
-rw-r--r--  joe  acctg  publicity.html
drwxrwxr-x  joe  acctg  sales
-rw-r-----  joe  acctg  topsecret.inf
-rwxr-xr-x  joe  acctg  wordmatic
```

First, let's prevent outsiders from executing `archive.sh`

Before:	<code>-rwxr-xr-x archive.sh</code>
Command:	<code>chmod o=r archive.sh</code>
After:	<code>-rwxr-xr-- archive.sh</code>

Take away all permissions for the group for `topsecret.inf` We do this by leaving the permissions part of the command empty.

Before:	<code>-rw-r----- topsecret.inf</code>
Command:	<code>chmod g= topsecret.inf</code>
After:	<code>-rw----- topsecret.inf</code>

Open up `publicity.html` for reading and writing by anyone.

Before:	<code>-rw-r--r-- publicity.html</code>
Command:	<code>chmod og=rw publicity.html</code>
After:	<code>-rw-rw-rw- publicity.html</code>

chmod Shortcuts

Let's say we have these files:

```
-rwxrwxrwx  joe  acctg  wordmatic
-r--r--r--  joe  acctg  calcmatic
```

We'd like to remove write permission for the **group** and **others** on `wordmatic`, and add write and execute permission for **all** users on `calcmatic`.

Rather than try to figure out what the new permissions are and do these commands:

```
chmod go=rwx wordmatic
chmod a=rwx calcmatic
```

The `chmod` command literally lets us add and subtract permissions from an existing set by using `+` or `-` instead of `=`.

Thus, we can take away the first file's write permission for the **group** and **others** with this command:

```
chmod go-w wordmatic
```

And we can add write and execute permission to the second file for **all** users with:

```
chmod a+wrx calcmatic
```

chmod by the Numbers

Up to this point, we've been setting the mode with letters. It turns out that you can also set the mode numerically. Here's how it works:

1. Write the permissions you want the file to have. To make your life easier, write the permissions grouped into sets of three letters. For example, let's say you want file `info.sh` to have these permissions `-rwx r-x r-- info.sh`
2. Under each letter, write a digit 1; under each dash write a digit zero. Ignore the dash at the very beginning that tells you whether it's a file or directory. This gives you three binary numbers.
`- rwx r-x r-- info.sh`
`111 101 100`
3. Now convert each set of three digits to a single digit using this table:

Binary	Becomes	Binary	Becomes
000	0	100	4
001	1	101	5
010	2	110	6
011	3	111	7

4. Now use that number in a `chmod` command to set your desired permissions on the file:
`chmod 754 info.sh`