Securing One's UNIX Files[*]

Jeffrey D. Oldham

1999 Dec 20

Even though the Trinity University Code of Ethics for Computing prevents others from using your UNIX files with your explicit permission, you are still responsible for securing the privacy of your files. At the very least, doing so will reduce the chances you will be accused of counterfeit work if someone else steals your work and submits it.

This note assumes you are familiar with files, directories, and the directory hierarchy. If not, read the file tutorial first.

1 Brief Overview of UNIX Permissions

Each file and directory has three sets of permissions. To look at them, use the `ls -l` command. For example, the permissions on the file `~joldham/foo` may be listed as

```
-rw-r--r--
```

The first hyphen indicates `foo` is a file, not a directory for which a `d` will appear. `rw-` indicates I, as the owner, can read and write this file. For example, I can use a text editor to both look at the file's contents and change the file's contents. The next triple `r--` indicates anyone in the file's group can only read the file (except, of course, for the file's owner who can also write the file). The last triple `r--` indicates all other users have the same privileges as anyone in the group. Thus, we see there are three categories: the file's owner, the file's group, and all other users. (By default on the Trinity UNIX system, a file's group contains only the file's owner so the second triple of group permissions is not particularly useful.)

[*] ©1999 Jeffrey D. Oldham (`oldham@cs.trinity.edu`). All rights reserved. This document may not be redistributed in any form without the express permission of the author.

1.1 Types of Permissions

Permissions for files and directories vary slightly:

permission	files	directories
r	can look at file's contents	can list a directory's files
w	can change a file's contents	can create or modify files in directory
x	can execute a file, i.e., run the program	can access files in directory

For example, a directory permission of `--x` permits using any file whose name is known, but one cannot list a directory's contents. A directory permission of `r--` permits listing the files in a directory but prohibits use of these files.

2 Specifying Permissions for Existing Files

To change the permissions on existing files and directories, use the `chmod` command. Its form is

`chmod` *permissions file*

where the *permissions* parameter consists of three parts:

affected users: choose one or more of these:

- **u** user who owns the file
- **g** users in the file's group
- **o** all other users
- **a** synonym for ugo

operation: choose one of these:

- $+$ add the following permissions
- $-$ remove the following permissions
- $=$ set the following permissions

permissions: choose any number (even zero) of these:

- **r** read
- **w** write

x execute

For example, using the command

```
chmod go= foo
```

removes all permissions for the group and everyone else for the file `foo`. The command

```
chmod u+x foo
```

adds execute privileges for `foo`'s owner.

3 Specifying Permissions for New Files

The `chmod` command only applies to existing files. To ensure that new files we create, e.g., using a text editor, automatically have the correct permissions we can use the `umask` command. For example,

```
umask go=
```

causes all files created in this shell (or by a text editor started using a command in this shell) to give no privileges to the group or anyone else except the file's owner.

(Technical note: This umask notation only applies to users of the bash shell, not csh or tcsh. Read the tcsh manual page for its umask notation.)

4 Applying These Commands to Your Files

The instructions in this section assumes you want to restrict all access to your files to only you, the owner. If you have a WWW page in your home directory, do not follow these instructions.

To change the permissions on all existing files and directories, you want to apply `chmod go=` *file* to all your files. It would be annoying to have to do this for each individual existing file and directory. Instead, if your current directory is your home directory, you can use the `find` command:

```
find . -exec chmod go= {} \;
```

Starting at the current directory, this will walk through the directory hierarchy applying the specified `chmod` command to each file (denoted by {}).

It would be annoying to have to type a umask command every time one logs in, we can have the umask automatically set. In your `.bash_profile` file (do not forget the initial period), make sure the umask line is either `umask 077` or `umask go=`. This will ensure that newly created files can only be accessed by you, the owner, not others.

5 Further Reading

For more information, read the following:

- file permissions and how to set them, as described in the GNU file utilities information pages
- file and directory permissions and how to change them, as described by some of the inventors of UNIX: Section 2.4 of *The UNIX Programming Environment*, by Brian W. Kernighan and Rob Pike, Prentice-Hall, 1984.
- the commands' info pages, available using the UNIX command `info` *command-name*. For example, to learn more about `chmod`, type `info chmod`.
- the commands' manual pages, available using the UNIX command `man` *command-name*. If you do not know the name of a command but do know a keyword, type `man -k` *keyword*. For example, to obtain a list of commands dealing with time, use `man -k time`.