AT&T

UNIX System Laboratories, Inc.
A Subsidiary of AT&T

UNIX® SYSTEM V RELEASE 4

User's Guide

Published by Prentice-Hall, Inc.
A Division of Simon & Schuster
Englewood Cliffs, New Jersey 07632

IMPORTANT NOTE TO USERS

While every effort has been made to ensure the accuracy of all information in this document, AT&T assumes no liability to any party for any loss or damage caused by errors or omissions or by statements of any kind in this document, its updates, supplements, or special editions, whether such errors are omissions or statements resulting from negligence, accident, or any other cause. AT&T further assumes no liability arising out of the application or use of any product or system described herein; nor any liability for incidental or consequential damages arising from the use of this document. AT&T disclaims all warranties regarding the information contained herein, whether expressed, implied or statutory, *including implied warranties of merchantability or fitness for a particular purpose.* AT&T makes no representation that the interconnection of products in the manner described herein will not infringe on existing or future patent rights, nor do the descriptions contained herein imply the granting or license to make, use or sell equipment constructed in accordance with this description.

AT&T reserves the right to make changes without further notice to any products herein to improve reliability, function, or design.

TRADEMARKS

DOCUMENTER'S WORKBENCH is a registered trademark of UNIX System Laboratories, Inc.
IBM is a registered trademark of International Business Machines.
Teletype is a registered trademark of AT&T.
UNIX is a registered trademark of UNIX System Laboratories, Inc.

10 9 8 7 6 5 4 3 2

ISBN 0-13-947052-2

UNIX
PRESS
A Prentice Hall Title

If the -w or -c option had been specified instead, the command would have reported the number of words or characters, respectively, in the file.

Figure 3-21 summarizes the syntax and capabilities of the wc command.

Figure 3-21: Summary of the wc Command

Command Recap wc – count lines, words, and characters in a file		
command	*options*	*arguments*
wc	-l, -w, -c	*file(s)*
Description:	wc counts lines, words, and characters in the specified file(s), keeping a total count of all tallies when more than one file is specified.	
Options	-l counts the number of lines in the specified file(s) -w counts the number of words in the specified file(s) -c counts the number of characters in the specified file(s)	
Remarks:	When a file name is specified in the command line, it is printed with the count(s) requested.	

Protecting Your Files: the chmod Command

The command chmod (short for change mode) allows you to decide who can read, write, and use your files and who cannot. Because the UNIX operating system is a multi-user system, you usually do not work alone in the file system. System users can follow path names to various directories and read and use files belonging to one another, as long as they have permission to do so.

If you own a file, you can decide who has the right to read it, write in it (make changes to it), or, if it is a program, to execute it. You can also restrict permissions for directories with the `chmod` command. When you grant execute permission for a directory, you allow the specified users to `cd` to it and list its contents with the `ls` command.

To assign these types of permissions, use the following three symbols:

`r`	allows system users to read a file or to copy its contents
`w`	allows system users to write changes into a file (or a copy of it)
`x`	allows system users to run an executable file

To specify the users to whom you are granting (or denying) these types of permission, use these three symbols:

`u`	you, the owner of your files and directories (`u` is short for user)
`g`	members of the group to which you belong (the group could consist of team members working on a project, members of a department, or a group arbitrarily designated by the person who set up your UNIX system account)
`o`	all other system users

When you create a file or a directory, the system automatically grants or denies permission to you, members of your group, and other system users. You can alter this automatic action by modifying your environment (see Chapter 9 for details). Moreover, regardless of how the permissions are granted when a file is created, as the owner of the file or directory you always have the option of changing them. For example, you may want to keep certain files private and reserve them for your exclusive use. You may want to grant permission to read and write changes into a file to members of your group and all other system users as well. Or you may share a program with members of your group by granting them permission to execute it.

How to Determine Existing Permissions

You can determine what permissions are currently in effect on a file or a directory by using the command that produces a long listing of a directory's contents: `ls -l`. For example, typing `ls -l` and pressing the `RETURN` key while in the directory named `starship/bin` in the sample file system produces the following output:

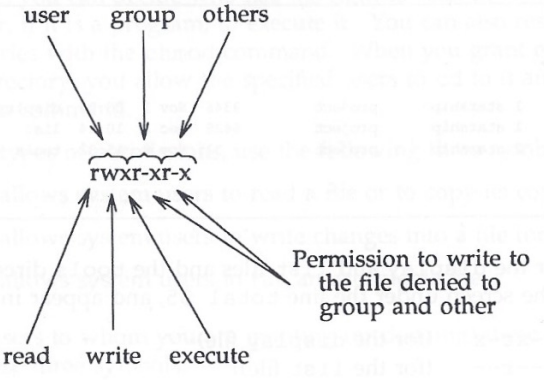
```
$ ls -l<CR>
total 35
-rwxr-xr-x   1 starship     project        9346  Nov 1   08:06  display
-rw-r--r--   1 starship     project        6428  Dec 2   10:24  list
drwx--x--x   2 starship     project          32  Nov 8   15:32  tools
$
```

Permissions for the `display` and `list` files and the `tools` directory are shown on the left of the screen under the line `total 35`, and appear in this format:

```
-rwxr-xr-x    (for the display file)
-rw-r--r--    (for the list file))
drwx--x--x    (for the tools directory)
```

After the initial character, which describes the file type (for example, a - (dash) symbolizes a regular file and a `d` a directory), the other nine characters that set the permissions comprise three sets of three characters. The first set refers to permissions for the owner, the second set to permissions for group members, and the last set to permissions for all other system users. Within each set of characters, the `r`, `w`, and `x` show the permissions currently granted to each category. If a dash appears instead of an `r`, `w`, or `x`, permission to read, write, or execute is denied.

The following diagram summarizes this breakdown for the file named `display`.



As you can see, the owner has `r`, `w`, and `x` permissions and members of the group and other system users have `r` and `x` permissions.

There are two exceptions to this notation system. Occasionally the letter `s` or the letter `l` may appear in the permissions line, instead of an `r`, `w` or `x`. The letter `s` (short for set user ID or set group ID) represents a special type of permission to execute a file. It appears where you normally see an `x` (or –) for the user or group (the first and second sets of permissions). From a user's point of view it is equivalent to an `x` in the same position; it implies that execute permission exists. It is significant only for programmers and system administrators. (See the *System Administrator's Guide* for details about setting the user or group ID.) The letter `l` indicates that locking will occur when the file is accessed. It does not mean that the file has been locked.

How to Change Existing Permissions

After you have determined what permissions are in effect, you can change them by executing the `chmod` command in the following format:

`chmod` *who+permission file(s)*`<CR>`

or

`chmod` *who=permission file(s)*`<CR>`

The following list defines each component of this command line.

`chmod`	name of the program
who	one of three user groups (`u`, `g`, or `o`) `u` = user `g` = group `o` = others
`+` or `-`	instruction that grants (+) or denies (–) permission
permission	any combination of three authorizations (`r`, `w`, and `x`) `r` = read `w` = write `x` = execute
file(s)	file (or directory) name(s) listed; assumed to be branches from your current directory, unless you use full pathnames.

NOTE

The `chmod` command will not work if you type a space(s) between *who*, the instruction that gives (+) or denies (–) permission, and the *permission*.

The following examples show a few possible ways to use the `chmod` command. As the owner of `display`, you can read, write, and run this executable file. You can protect the file against being accidentally changed by denying yourself write (`w`) permission. To do this, type the command line:

```
chmod u-w display<CR>
```

After receiving the prompt, type `ls -l` and press the `RETURN` key to verify that this permission has been changed, as shown in the following screen.

```
$ chmod u-w display<CR>
$ ls -l<CR>
total 35
-r-xr-xr-x   1 starship      project        9346  Nov 1  08:06  display
rw-r--r--    1 starship      project        6428  Dec 2  10:24  list
drwx--x--x   2 starship      project          32  Nov 8  15:32  tools
$
```

As you can see, you no longer have permission to write changes into the file. You will not be able to change this file until you restore write permission for yourself.

Now consider another example. Notice that permission to write into the file `display` has been denied to members of your group and other system users. However, they do have read permission. This means they can copy the file into their own directories and then make changes to it. To prevent all system users from copying this file, you can deny them read permission by typing:

```
chmod go-r display<CR>
```

The `g` and `o` stand for group members and all other system users, respectively, and the `-r` denies them permission to read or copy the file. Check the results with the `ls -l` command.

```
$ chmod go-r display<CR>
$ ls -l<CR>
total 35
-rwx--x--x   1 starship      project        9346  Nov 1  08:06  display
rw-r--r--    1 starship      project        6428  Dec 2  10:24  list
drwx--x--x   2 starship      project          32  Nov 8  15:32  tools
$
```

A Note on Permissions and Directories

You can use the `chmod` command to grant or deny permission for directories as well as files. Simply specify a directory name instead of a file name on the command line.

However, consider the impact on various system users of changing permissions for directories. For example, suppose you grant read permission for a directory to yourself (u), members of your group (g), and other system users (o). Every user who has access to the system will be able to read the names of the files contained in that directory by running the `ls -l` command. Similarly, granting write permission allows the designated users to create new files in the directory and remove existing ones. Granting permission to execute the directory allows designated users to move to that directory (and make it their current directory) by using the `cd` command.

An Alternative Method

There are two methods by which the `chmod` command can be executed. The method described above, in which symbols such as `r`, `w`, and `x` are used to specify permissions, is called the symbolic method.

An alternative method is the octal method. Its format requires you to specify permissions using three octal numbers, ranging from 0 to 7. (The octal number system is different from the decimal system that we typically use on a day-to-day basis.) To learn how to use the octal method, see the `chmod`(1) entry in the *User's Reference Manual*.

Figure 3-22 summarizes the syntax and capabilities of the `chmod` command.