UnixWorld Online: "Answers to Unix" Column: No. 001

Sharing Files and Creating Form Letters

By Ray Swartz

Questions regarding this column should be sent to the author at ray@cse.ucsc.edu.

Jeyakesavan Veerasamy would like to share files with other users on his system without giving them complete access to this entire directory hierarchy. A directory of shared files can be created by using links. Karl Kulling wants to combine addresses with a letter file to create form letters. I provide two solutions, one running Awk, the other Perl.

A Trip to the Links

Question: I have a multi-level directory structure under my home directory. If I want to make the lowest-level directory files visible or accessible to others, not only do I have to give proper permissions for the files in that directory, but also I have to provide execute (search) permission for all directories on the path to reach that directory, which I would prefer not to do. For example, if I want to make the files in `~veerasam/lang/c++/conf` accessible to others, I have to give search permissions for `~veerasam`, `~veerasam/lang` and `~veerasam/lang/c++`. Is there a simple one-step method to make my life little easier?

Jeyakesavan Veerasamy / Richardson, Texas.

Answer: UNIX system files consists of two parts: an inode and a name. The inode holds all the information about the file, including--among other things--the file's owner, work group, permissions, size, and a list of file blocks that belong to this file. A directory entry associates the inode with a file name.

File inodes are identified by number with the directory mapping the inode number to the name. You can list the file name -inode number pairs by specifying the `-i` option with the `ls` command. Listing 1A shows sample output from the `ls -i` command line. The output identifies the inode numbers assigned to the names in the current directory.

The inode number for a file doesn't change after the file is created. That is, the file's contents are associated with a specific inode, and that inode will always refer to that file's data. By contrast, the file's name can be changed easily (for instance, by using the `mv` command). However, even when a file's name is changed, its inode number remains the same.

In fact, a Unix file can have more than one name. More accurately, a file's inode can have several links. A new link to a file is created with the `ln` command. The syntax for this link command is `ln` *old-name new-name*.

Because a link connects a name to an inode number, every link will represent the same file. In other words, every link will have the same permissions, owner, work group, file times and all other file-related information, with one exception--the name. Each link identifies the file with its assigned name.

The inode must keep track of the number of links that have been connected to it. Without this knowledge, the system wouldn't know when the last link to a file was removed so it can recycle the inode and any allocated blocks. The best place to keep this link count information is in the inode itself.

Every time a link to an inode is added (say, with `ln`) or removed (say, with `rm`), the link count stored in the inode is updated. When the link count reaches zero, the inode is cleared, its data blocks are freed, and the inode is made available for reassignment.

A file's link count is displayed in the second column of output from the `ls -l` command line. Listing 1B shows an interactive session that creates a link, lists it, and then removes it. Note that both links to the same file refer to the same inode number.

The solution to your problem is creating links to the files that you wish to share in the directory where you want them to be visible. This way, any user with the appropriate access to the "sharing" directory can access or copy the files you want to make available to them. Note that you will have to give the users in question search and read access to the "sharing" directory. Also, the files will have to have proper permissions assigned to them so that users can view and copy them.

Listing 1C shows an example. Here, the directory `hidden` contains the files `code1.c` and `code2.c`. These are linked to the `shared` directory using the `ln` command with the names changed to `program1.c` and `program2.c`, respectively.

Open A Mailing Frequency

Question: I have a data base of addresses where each address is one line long. The fields on the line are separated by colons (`:`). I would like to print form letters for each address in the data base [format shown in Listing 2A]. The same form letter can be used for each recipient and their name and address should be printed at the top of each letter. No special salutation is required. What can you suggest?

Karl Kulling / Pawling, NY

Answer: I've provided two ways to produce the output you request. One is with a shell script that primarily depends on the Unix Awk utility. The other is a Perl program. I've provided both to demonstrate two different ways to do the same task.

The basic design is to read the letter into an array and then to process the addresses. For each address, a formatted letter is sent to the standard output. Both programs take exactly two arguments. The first one is the name of the form letter file and the second is the address data base file name.

The Awk version, named `merge.awk`, is shown in Listing 2B. In this program, the information from both files is sent to the program's standard input. After the argument count is checked (lines 5-8), the form letter is sent first, followed by a separator token (`@@@`), and finally the list of addresses (line 9).

The `merge.awk` program has two states that match the input it is getting: either form letter or addresses.

The states are identified by the value stored in `addrs`. A zero means the separating token hasn't been seen yet and the input is the form letter (line 14). A one means the token has been seen and the input is an address line (lines 10-13). When the token is found (line 15), the value of `addrs` is incremented from zero to one. Also, the field separator is set to a colon.

The Perl version, named `merge.perl`, is shown in Listing 2C. After testing the number of arguments (line 5), this program opens the form- letter file (line 8), reads the letter into the `@letter` array (line 11), and stores the line count of this array in the variable `$letlen` (line 14). After the letter is read into the array, the address file is opened (line 17) and then read line by line (line 20). Each line is split into fields (line 22) and the fields are listed at the top of a letter (line 23) and then the letter is printed (lines 24-26). Each letter is terminated with a form feed character.

Edited by Becca Thomas / Online Editor / UnixWorld Online / beccat@wcmh.com

Go To Content   Search

Last Modified: Tuesday, 22-Aug-95 15:55:15 PDT