Open Computing ‘‘Hands-On’’: ‘‘Answers to Unix’’ Column: January 94

Can You Find the Time?

Finding file creation times, fixing a C shell script, and a public-domain program for converting numbers to words.

By Ray Swartz

This month Joe Walker requests a way to track when a file was created. Because the Unix file system doesn’t store this time, I’ve provided a script that stores the file’s birth date in a simple database. Jase Wells asks for help with a C shell script that doesn’t process all the desired files. The problem is how the command-line arguments are referenced. In our feedback item, Tilman Schmidt identifies a public-domain program that implements a simple translator of digits to words in various languages, an extension of the simple digit-to-English-word script that I presented in my August 1993 column and have reproduced here in Listing 1.

So, What’s New?

Question: Is there some way to find out when a file was created?

Joe Walker / Carson City, Nev.

Answer: The Unix file system maintains three times for each file stored on the system: the last time the file was modified (written to), accessed (read from), or its inode entry was last changed (through various ways).

Use the `ls` command with the appropriate option to note these times: `ls -l` lists the file’s modification time, `ls -lu` its access time, and `ls -lc` the inode change time.

When the file is first created, all three times are set to its creation time. However, as soon as a process writes to the file, the modification and inode change times are updated. Similarly, the access time no longer reflects the creation time after the file is read.

Because the file system doesn’t record when the file was created, there is no Unix command that will tell you when a file first appeared in the file system. As a result, you will have to create your own data file and program to provide this information.

One solution, named `cmpfiles`, was presented in the May 1993 ‘‘Wizard’s Grabbag’’ column. This program is a large script with many options that maintains a database of file-modification times. To create its data, `cmpfiles` runs `ls - l` on all the files located by the find command. This approach requires a good deal of system resources.

Also, `cmpfiles` searches the entire file system. Today, many systems have directories mounted over the network. My solution, `find.new.files` [see Listing 2A], only searches for files actually stored on this machine using find's fstype 4.2 option. The 4.2 argument works on my Sun Microsystems Inc.'s workstation and should work for other BSD- based systems. Your system may require other options that tell `find` to search files actually located on your local disk.

The task is much simpler if you are willing to settle for knowing the day the file's name first appeared in the file system. This solution can be implemented with a shell script that is run daily by `cron`. [See Listing 2B for a sample entry.]

The 20-line `find.new.files` script locates new files by comparing entries in a file (`/usr/local/data/file.list`) with the output of the `find` command. Existing files will be in both listings. New files will be listed by `find`, but won't be in `/usr/local/data/file.list`. Deleted files appear in `/usr/local/data/file.list` but not in `find`'s output.

The `find.new.files` script employs a common shell-programming trick. It creates a list, sorts it, and then uses `uniq` to find the duplicates (files still on the system) and the unique lines (files deleted or created today). To distinguish between deleted and created files, `find.new.files` uses a data flag of `99/99/99`. Any file with this marker is a new file.

After setting some variables, `find.new.files` checks whether its data file exists. If not, it creates one (lines 6-8). If its data file exists, `find.new.files` makes a copy of the data file for archival purposes (line 10). The `find` command (line 11) is run to locate all files stored on this machine. These file names are piped to `awk` for formatting (line 12). This output needs to be combined with the current data file (line 13) so that new and deleted files can be found.

The parentheses surrounding lines 11-13 tell the shell to run these commands in a subshell. The effect is to redirect the output of both the pipeline and the `cat` command into the pipeline.

The `sort` on line 14 orders the files by path name. (The character inside the single quotes is a tab.) The output of `sort` will be two entries for an existing file and only one entry for new or deleted files.

The `uniq` command is used to count the number of entries (line 15). The `-c -3` options tell `uniq` to place the number of entries in front of each line and to skip the three words in front of the path name.

The `uniq` utility reports one of three things: Lines beginning with a ''2'' are existing files that need to be put into the data file along with their previous creation time. Lines beginning with ''1'' but containing ''99/99/99'' are new files.

Lines beginning with ''1'' without ''99/99/99'' are files that have been deleted. The `egrep` in line 16 searches this output for existing files and new files, throwing away those no longer on the file system. (Note that there is a tab inside the single quotes at the end of the first argument.)

The `sed` command in line 17 removes the counts inserted by `uniq` and exchanges today's date for the new file marker (99/99/99). When the data file `/usr/local/data/file.list` is created for the first time, all the files currently on the system will be listed as being created on that day.

The `find.new.files` program is designed to be run once a day. The best method is to have `cron` run it when file system is usage is minimal, say in the middle of the night. Be careful with `find.new.files` because it searches the entire directory hierarchy so it may consume a good deal of your system's capacity when it executes.

Sending Him for a Loop

Question: I've written a simple shell script (called `g2j`) that automates the process of converting files from GIF format to JPEG format, calling on the `cjpeg` program. I call it with the command line: **`g2j *.gif`** [see Listing 3].

It just uses a ''for each'' loop to send each file whose name ends with the ''.gif'' extension to the `cjpeg` program (because `cjpeg` only handles one file at a time). For some reason, this script works fine for the first time through the loop, then exits without processing the rest of the files.

Do you have any idea what I may have done wrong?

Jase Wells / Martinez, Calif.

Answer: The problem results from a misconception about how arguments are passed to a script. When an argument on a command line contains file matching metacharacters, such as `*.gif`, the shell expands that single argument into all the file names that the specified pattern matches. Thus, what appears to be a single argument on the command line often turns into many arguments when the command line is actually passed to a script or a program.

Your ''foreach'' loop is controlled by the output of `ls $1`. Because the argument, `*.gif`, has already been expanded, `ls $1` only identifies the first of these expanded arguments, which is why your script stops after the first file is processed.

To correct this problem, use a shell matching metacharacter that identifies all the arguments on the command line. For the C shell, replace `$1` with `$argv[*]`.

Go to the Source

Tilman Schmidt of Montrouge, France, writes: ''Reading your August 1993 column, I thought it might be interesting for other readers to know that there is a program published on Usenet that converts numbers not only to English words, but to a host of other languages also. And more can easily be added. The program is called `number` and it was posted to the comp.sources.unix newsgroup in 1987 and is part of the volume 11 source distribution. It is available by anonymous FTP from ftp://ftp.uu.net/usenet/comp.sources.unix/volume11/number (34K) and other archive servers all over the world. Use Archie to find some mirror sites.''

Copyright © 1995 The McGraw-Hill Companies, Inc. All Rights Reserved.
Edited by Becca Thomas / Online Editor / UnixWorld Online / beccat@wcmh.com

Go To Content   Search

Last Modified: Tuesday, 22-Aug-95 15:45:39 PDT