

A Brief Introduction To C Shell Variables and The Unix ‘Dot’ Files

Norman Matloff

January 18, 2001

Contents

- 1 C Shell Variables
 - 1.1 \$cwd
 - 1.2 \$path
 - 1.3 \$term
- 2 Dot Files
 - 2.1 The .cshrc File
 - 2.2 .login
 - 2.3 Example
 - 2.4 Comments
- A Some Lines From My .cshrc File

1 C Shell Variables

Much of the Unix development environment involves use of shell variables. Some of these are variables which the user makes up himself/herself, but most of them are intrinsic to the shell.

Typing

```
set x = 2
```

would create a new shell variable named x (this would be a user-defined variable).¹

Here are some examples of important shell variables:

1.1 \$cwd

This variable consists of a string which records the current directory.

Typing

```
set cwd = b
```

would have the same effect as

```
cd b
```

while typing

```
echo $cwd
```

would have the same effect as

```
pwd
```

(the **echo** command simply writes the given string or strings to the screen).

1.2 \$path

This is an extremely important variable. When you issue a command to the shell, the shell will search through various directories to find the executable file for that command, so that the command can be run.

A typical value for \$path might be

```
(. /usr/local/bin /usr/ucb /usr/bin /usr/etc /etc /bin /usr/bin/X11 )
```

Suppose we give the shell the command `z`. The shell will first search for a file named `z` in our current directory (`.`); if not found there, the shell will next look for the file in the directory `/usr/local/bin`; and so on. (Many of the commands you have been using so far—e.g. **ls**, **mail**, etc.—are in the directory `/usr/ucb`. Take a look!)

If you create a new directory in which you put executable files which you use often, you should add this directory to your path, so that you can conveniently execute the programs from any other directory. Suppose the full directory name is `/a/b/c`. Then to add it to your the end of your path, add a line

```
set path = ( $path /a/b/c )
```

to your `.cshrc` file. This means that the directory `/a/b/c` will be checked last. If you wish it to be checked first (there may be files of the same name in different portions of the path), add the line

```
set path = ( /a/b/c $path )
```

By the way, if you add a new program to your system, or use **mv** to rename it, you probably will then need to run **rehash** to let your shell know that the set of executables it found before needs to be updated.

1.3 \$term

This one is also quite important. Without it, programs like **emacs**, **vi**, **talk**, etc. would not know your terminal type, and would be useless.

2 Dot Files

Many commands have **startup** files from which the command will get some initialization directions which the user has put in to customize the command for his/her convenience. Usually these reside in the user's home directory, and have names beginning with a period (thus the term "dot files").

For example, you have already seen the `.mailrc` file, in which the program **mail** will get information such as mail aliases which you have set up. You've also seen the `.emacs` file, from which **emacs** gets its customization.

2.1 The `.cshrc` File

If you use the C shell², **cs**h will get its initialization from the `.cshrc` file. For example, when you first log in, your initial C shell is started, and this file read and its commands executed. Afterward, every time you invoke a new C shell, the file is read/executed again; for instance, each X-window you create will be a new invocation of **cs**h.

The `.cshrc` file mainly sets shell variables and defines **aliases**.

An example of an alias is

```
alias ls 'ls -Fa'
```

This tells the shell that, whenever you type `ls`, you really mean `ls -Fa`, i.e. you want to include the `F` and a flags whenever you run the **ls** command.

2.2 `.login`

Another file, `.login`, is executed only once per session, right after the initial C shell is set up (so `.cshrc` executes first, then `.login`). The file `.login` should contain those commands that need only be executed once.

2.3 Example

In my `.cshrc` file I (NM) have a line

```
alias rm 'mv \!* ~/Trash'
```

Here I am telling the shell that whenever I issue the **rm** command, I don't really want that file removed, but instead want it moved to a subdirectory named `Trash` in my home directory. That way, if I accidentally delete a file, I can recover it from my `Trash` directory. The item

```
\!*
```

means "all parameters from the command line." If, say, I type

```
rm a b
```

the parameters are `a` and `b`, and my alias will mean that my command is really

```
mv a b ~/Trash
```

i.e. move the files `a` and `b` to the `Trash` directory, instead of actually removing them from the disk.

Well, the `Trash` directory will keep growing and growing, so once in a while I will want to (truly)

remove those files. I decided that I would do so whenever the total size of Trash exceeded 1 megabyte (1,000 kilobytes, i.e. 1 million bytes), and that I would check for this every time I logged in. So, I put the following lines in my `.login` file:

```
set trashsize = (`du -s ~/Trash`)
if ($trashsize[1] > 1000) then
    /bin/rm -r ~/Trash/*
endif
```

Here are the details: I run the `du` command to see how big Trash is, and assign the output of that command to the variable `$trashsize`, which will look like, say

```
830      /usr/home/matloff/Trash
```

There are two components here, ‘830’ and ‘/usr/home/matloff/Trash’. I’m interested in the first of these, which I can refer to as `$trashsize[1]`. I then use the C shell’s if-then-else capability³ to check whether the size of Trash has exceeded 1,000 kilobytes, and if so, to remove all files in that directory.⁴

More examples are shown in the Appendix.

2.4 Comments

In most dot files, any line beginning with ‘#’ is treated as a comment. These are useful in the same way as comments in ordinary C programs are.

A Some Lines From My `.cshrc` File

```
# misc. set conditions:
set mail=(5 /usr/spool/mail/matloff 300 /etc/motd 120 /usr/messages/bounds)
set history=50
set noclobber
set ignoreeof
# make default file-access permissions for group/others nil
umask 077
```

```
# class directories; e.g. `cd a40`:
set a40 = ~/Cou*/40
set a40hand = $a40/Handouts
set a40progs = $a40/Progs
set a50 = ~/Cou*/50
set a70 = ~/Cou*/70/Book
set a70hand = $a70/Han*
set a158 = ~/Cou*/158
set a171 = ~/Cou*/171
set a250a = ~/Cou*/250A
set a282 = ~/Cou*/282
```

```
# abbreviations for frequently-used commands:
alias m more
alias hi history
alias e exit
alias f finger
alias cps compress
alias ucps uncompress
```

```

# misc. home-grown commands:
# get directory, executable, etc. information whenever do ls
alias ls 'ls -F'
# give user execute permission
alias ux 'chmod u+x \!*'
# quick process kill
alias k9 'kill -9 \!*'
# to view *.1 man files
alias nman 'nroff -man \!*.1 | m'
# to view *.man man files
alias nman2 'nroff -man \!*.man | m'
alias fnd "find . -name '\!*' -print"

# change-directory tools:
# push current directory onto the stack; change directory and report
#   what is there
alias cd "pushd \!* ; ls"
# "cd silent"
alias cds "pushd \!*"
# pop directory and report what is in new current directory
alias p "popd ; ls"
alias p2 "popd; popd; ls"
alias p3 "popd; popd; popd; ls"
# swap the current directory with the one second in the stack
alias s "pushd ; ls"
# "up (one level in) directory"
alias up 'cd ..'

# remove-file tools:
# safe remove; builds up a directory ~/Trash, but removes that once
#   a day by .login, which does rm -r ~/Trash/*
alias rm 'mv \!* ~/Trash'
#   if need a "real" remove, use brm
alias brm '/bin/rm'

# facilitate updating .cshrc:
# change it
alias ec 'emacs ~/.cshrc'
# implement the new one without logging out and back in again
alias sc 'source ~/.cshrc'
# check a definition
alias ag 'grep \!* ~/.cshrc'

alias rlc "rlogin \!*.engr.ucdavis.edu"
alias ftpc "ftp \!*.engr.ucdavis.edu"
# Melvyl on-line UC library catalog
alias melvyl 'telnet melvyl.ucop.edu'
# Library of Congress card catalog
alias lc 'telnet dra.com'
# set terminal type

# ftp-archive file tools:
# uncompress, untar ftp archive files:
alias untar "uncompress \!*.tar.Z; tar xf \!*.tar"
alias untar2 "mv \!*.tar-z \!*.tar.Z; untar \!*"
alias untar3 "uncompress *.Z; tar xf *.tar"
# Washington University at St. Louis, huge archive
alias ftparch "ftp 128.252.135.4"
# alternate to Wash. U.:
alias ftpoak "ftp rigel.acs.oakland.edu"

```

Footnotes:

¹Typing ‘set’ with no argument produces a listing of all variables defined so far, and their current values.

²This is determined when you log in. Unix will look at the file /etc/passwd, where an entry will indicate which shell to use. That entry can be changed, using the **chsh** command.

³So you can see that the C shell can be regarded as a programming language. Setting up commands like this is known as **shell programming**.

⁴Some of those files may in turn be directories, so I needed the -r option of the **rm** command. By the way, note that I did need to use ‘/bin/rm’ here, not just ‘rm’, because in my .cshrc file I had already aliased ‘rm’ to ‘mv...’.

File translated from T_EX by T_TH, version 2.60.

On 18 Jan 2001, 13:39.