The MacOS X Command Line

My wife thinks I bought my Mac laptop to use as a status symbol. But every hacker knows I bought it because I wanted a decent Unix laptop.

What you will learn…

- Apple-specific command line tools
- Opening most files
- Working with the clipboard
- Taking screen shots

What you should know…

- The Unix Command line
- How to get around MacOS X

The fact it was based on BSD was even better. MacOS X features a command line interface that is as authentic as any Unix interface because BSD runs at the core of MacOS X. But Apple has provided a number of command line tools to enhance the experience and this article outlines the author's favorites.

open

MacOS X provides a command line tool to open applications and files. MacOS X applications are actually collections of files residing within one directory with a name ending in .app. I usually use open at the command line to start most applications, leaving the Dock clear of applications not running:

```
howardjp@thermopylae:~$ open /Applications/Safari.app
```

is enough to start Safari and if the browser is already running, it will open a new window. The `open` command also works on individual files and will open the file in its associated application. For instance, running open on a PDF will open the file in Preview. And running `open` on a normal directory (as opposed to an application package) will open the directory in Finder. The `open` command provides a number of useful options. The option `-t` treats the file, regardless of type, as a text file and opens it in the default text editor. A related option, `-e` simplifies the process and opens the file in TextEdit, the native text editor provided with MacOS X. Also related is `-f`, which reads from the standard input and passes the input to the default text editor.

It is also possible to override the default application with other types of files using the option `-a`. But it is important to remember the full path to the application must be given:

```
open -a /Applications/Adobe\ Reader\ 9/Adobe\
Reader.app/foo.pdf
```

This form is quite cumbersome, but it may be appropriate in some circumstances. One last option worth mentioning is `-R` which find the references file in Finder, instead of opening the file itself. Finally, the `open` also supports URLs:

```
open http://www.jameshoward.us
```

will open my website directly in the default browser.

pbcopy and pbpaste

The Unix command line has historically interacted poorly with the numerous graphical interfaces that have been stacked upon it. One key area lacking support is the clipboard. MacOS X brings two utilities to close that gap, `pbcopy` and `pbpaste`. These commands together provide complete access to the MacOS X clipboard (which Apple calls the pasteboard, explaining the names of these two commands). The first of

the two, pbcopy, takes its input from the standard input and adds it to the system clipboard. The command only accepts one option, -pboard, which accepts one of four suboptions, *general'*, *ruler*, *find*, and *font*, all of which are different system clipboards available on MacOS X. The general pasteboard is the main system clipboard and the others are for special use. The pbpaste pulls data from the clipboard and prints it to the standard output. Like pbcopy, pbpaste accepts the option \opt{pboard} to determine which pasteboard to acquire data from. The pbpaste command adds a second option, -Prefer which takes three possible options *txt*, *rtf*, and *ps*. These options direct pbpaste to looks for a certain type of formatted information on the pasteboard. The *txt* flag suggests standard text data. The *rtf* and *ps* suggest Rich Text Format and PostScript, respectively. Despite this option, it is not possible to direct the exact output pbpaste prints. This option only tells pbpaste what type of information to return first. These two commands offer the MacOS X command line warrior a simple and fairly complete set of tools for working with and manipulating the Mac OS X pasteboards.

Screencapture

Another command line gem in MacOS X is a screen capturing program called screencapture. This command line application accepts a handful of options making the tool quite powerful. The program requires a single command line option, a file name to store the screen capture in. Without any other options, this will copy the full screen to the named file, which is stored in PNG format by default. The file format can be changed with the option -t which accepts *pdf*, *jpg*, and *tiff* as acceptable formats. The manual page suggests other formats are permissible. Experimentally, *gif* works and *ps* does not. The option -w instructs screencapture to only capture a single window and highlights the current window. Moving the mouse will allow the user to select a different window for capture. The -o option forces screencapture to ignore the shadow when capturing a single window. Like other screen capture utilities, screencapture allows the user to select a delay before taking the image with the T option, which accepts a number as the number of seconds to wait. The screencapture command provides other useful options. When the screen is captured with this utility, it triggers a sound like camera shutter opening and closing to signal the capture has been taken. This can be disabled, probably for nefarious purposes, using the -x option. Also when using the option -P, the utility will automatically open the saved image file in the Preview.app application. The screencapture command provides other options for controlling how a window can be selected and also for opening the screen capture in a new Mail.app message.

binhex and macbinary

If you have been a Macintosh user since before MacOS X, then you may have a collection of files stored in some of Apple's unique formats, such as BinHex or MacBinary. Apple has provided a command line tool for creating and converting these file formats. Prior to adopting the Unix-like structure of MacOS X, Apple used a proprietary disk format called HFS (an extended version called HFS Plus was also available). This disk format broke files into multiple components called forks. There were normally two forks with the first being traditional data. The second, called the *resource fork* included metadata applicable to the file, such as associated applications or icons. To simplify transfer of these files, the MacBinary format was created, that combined the forks of a file into a single package suitable for transport. They typically had a file name ending in .bin or *.macbin*. Apple provides macbinary for working with these types of files. The macbinary command takes a subcommand as its first option. Available subcommands are *encode*, which creates a new MacBinary file, *decode* which unpackages an existing MacBinary file, and *probe* which attempts to determine if the files listed are MacBinary files. Similar to MacBinary is that the BinHex format packages the different forks of an HFS-based file into one file, but also makes that file 7-bit clean for transferring over ASCII connections, such as email. This is similar to the use of uuencode on the Unix platform. These files typically had the extension *.hqx*. Apple also provides binhex to work with these files and it takes the same options as macbinary. Both commands take several options, but the most useful is -c which makes the two commands read from the standard input for *decode* and write to the standard output for *encode*.

Other Tools

The traditional Unix command uname is available for interested users, but Apple has provided a second command for MacOS X specific information. That command, sw_vers, will provide the product name (distinguishing between MacOS X and MacOS X Server), the operating system version, and the build number. In addition, there are a collection of utilities for accessing XCode, the native IDE for MacOS X, package building, and other developer tools. These were not including in this overview due to their technical nature, but they are useful to understand Apple has considered the needs of programmers when deviating from common practice in the Unix world.

JAMES P. HOWARD, II

The author is a senior analyst in Washington, DC, in the United States where he focuses on statistical and mathematical systems. He can be reached at jh@jameshoward.us or via Twitter @howardjp.