Moving to zsh, part 2: Configuration Files

By Published by ab Mac Admin, Consultant, and Author View all posts by ab

Published: 2019-06-18 · Archived: 2026-04-05 21:26:08 UTC

Apple has announced that in macOS 10.15 Catalina the default shell will be `zsh`.

In this series, I will document my experiences moving `bash` settings, configurations, and scripts over to `zsh`.

- Part 1: Moving to zsh
- Part 2: Configuration Files *(this article)*
- Part 3: Shell Options
- Part 4: Aliases and Functions
- Part 5: Completions
- Part 6: Customizing the `zsh` Prompt
- Part 7: Miscellanea
- Part 8: Scripting `zsh`

> This series has grown into a book: reworked and expanded with more detail and topics. Like my other books, I plan to update and add to it after release as well, keeping it relevant and useful. You can order it on the Apple Books Store now.

In part one I talked about Apple's motivation to switch the default shell and urge existing users to change to `zsh`.

Since I am new to `zsh` as well, I am planning to document my process of transferring my personal `bash` setup and learning the odds and ends of `zsh`.

Many websites and tutorials leap straight to projects like oh-my-zsh or prezto where you can choose from hundreds of pre-customized and pre-configured themes.

While these projects are very impressive and certainly show off the flexibility and power of `zsh` customization, I feel this will actually prevent an understanding of how `zsh` works and how it differs from `bash`. So, I am planning to build my own configuration 'by hand' first.

At first, I actually took a look at my current `bash_profile` and cleaned it up. There were many aliases and functions which I do not use or broke in some macOS update. I the end, this is what I want to re-create in `zsh`:

- aliases
  - mostly shortcuts to `open` files with a specific application
- functions
  - show man pages in a dedicated Terminal window
  - some more simple functions
  - get the frontmost Finder window path
- shell settings

    - case-insensitive globbing
    - case-insensitive path-completion (for `bash` this is set in `.inputrc`)
    - command history, shared across windows and sessions
    - use BBEdit as the editor
- prompt:
    - show current working dir
    - show a colored symbol showing the last command's exit code
    - update the Terminal window title bar to show the cwd

Most of these should be fairly easy to transfer. Some might be… interesting.

But first, where do we put our custom `zsh` configuration?

`bash` has a list of possible files that it tries in predefined order. I have the description in my post on the `bash_profile`.

`zsh` also has a list of files it will execute at shell startup. The list of possible files is even longer, but somewhat more ordered.

all users	user	login shell	interactive shell	scripts	Terminal.app
`/etc/zshenv`	`.zshenv`	√	√	√	√
`/etc/zprofile`	`.zprofile`	√	x	x	√
`/etc/zshrc`	`.zshrc`	x	√	x	√
`/etc/zlogin`	`.zlogin`	√	x	x	√
`/etc/zlogout`	`.zlogout`	√	x	x	√

The files in `/etc/` will be launched (when present) for all users. The `.z*` files only for the individual user.

By default, `zsh` will look in the root of the home directory for the user `.z*` files, but this behavior can be changed by setting the `ZDOTDIR` environment variable to another directory (e.g. `~/.zsh/`) where you can then group all user `zsh` configuration in one place.

On macOS you could set the `ZDOTDIR` to `~/Documents/zsh/` and then use iCloud syncing (or a different file sync service) to have the same files on all your Macs. (I prefer to use `git`.)

`bash` will either use `.bash_profile` for login shells, or `.bashrc` for interactive shells. That means, when you want to centralize configuration for all use cases, you need to `source` your `.bashrc` from `.bash_profile` or vice versa.

`zsh` behaves differently. `zsh` will run *all* of these files in the appropriate context (login shell, interactive shell) when they exist.

`zsh` will start with `/etc/zshenv`, then the user's `.zshenv`. The `zshenv` files are *always* used when they exist, *even for scripts* with the `#!/bin/zsh` shebang. Since changes applied in the `zshenv` will affect `zsh` behavior in *all* contexts, you should you should be very cautious about changes applied here.

Next, when the shell is a login shell, `zsh` will run `/etc/zprofile` and `.zprofile`. Then for interactive shells `/etc/zshrc` and `.zshrc`. Then, again, for login shells `/etc/zlogin` and `.zlogin`. Why are there two files for login shells? The `zprofile` exists as an analog for `bash`'s and `sh`'s profile files, and `zlogin` as an analog for `ksh` login files.

Finally, there are `zlogout` files that can be used for cleanup, when a login shell exits. In this case, the user level `.zlogout` is read first, then the central `/etc/zlogout`. If the shell is terminated by an external process, these files might not be run.

Apple Provided Configuration Files

macOS Mojave (and earlier versions) includes `/etc/zprofile` and `/etc/zshrc` files. Both are very basic.

`/etc/zprofile` uses `/usr/libexec/path_helper` to set the default `PATH`. Then `/etc/zshrc` enables UTF–8 with `setopt combiningchars`.

Like `/etc/bashrc` there is a line in `/etc/zshrc` that would load `/etc/zshrc_Apple_Terminal` if it existed. This is interesting as `/etc/bashrc_Apple_Terminal` contains quite a lot of code to help `bash` to communicate with the Terminal application. In particular `bash` will send a signal to the Terminal on every new prompt to update the path and icon displayed in the Terminal window title bar, and provides other code relevant for saving and restoring Terminal sessions between application restarts.

However, there is no `/etc/zshrc_Apple_Terminal` and we will have to provide some of this functionality ourselves.

> Note: As of this writing, `/etc/zshrc` in the macOS Catalina beta is different from the Mojave `/etc/zshrc` and provides more configuration. However, since Catalina is still beta, I will focus these articles on Mojave and earlier. Once Catalina is released, I may update these articles or write a new one for Catalina, if necessary.

Which File to use?

When you want to use the `ZDOTDIR` variable to change the location of the other `zsh` configuration files, setting that variable in `~/.zshenv` seems like a good choice. Other than that, you probably want to *avoid* using the `zshenv` files, since it will change settings for *all* invocations of `zsh`, including scripts.

macOS Terminal considers every new shell to be a login shell *and* an interactive shell. So, in Terminal a new `zsh` will potentially run *all* configuration files.

For simplicity's sake, you should use just one file. The common choice is `.zshrc`.

Most tools you can download to configure `zsh`, such as 'prezto' or 'oh-my-zsh', will override or re-configure your `.zshrc`. You could consider moving your code to `.zlogin` instead. Since `.zlogin` is sourced *after* `.zshrc` it can override settings from `.zshrc`. However, `.zlogin` is *only* called for login shells.

The most common situation where you do *not* get a login shell with macOS Terminal, is when you switch to `zsh` from another shell by typing the `zsh` command.

I would recommend to put your configuration in *your* `.zshrc` file and if you want to use any of the theme projects, read and follow their instructions closely as to how you can preserve your configurations together with theirs.

Managing the shell for Administrators

MacAdmins may have the need to manage certain shell settings for their users, usually environment variables to configure certain command line tool's behaviors.

The most common need is to expand the `PATH` environment variable for third party tools. Often the third party tools in question will have elaborate postinstall scripts that attempt to modify the current user's `.bash_profile` or `.bashrc`. Sometimes, these tools even consider that a user might have changed the default shell to something other than `bash`.

On macOS, system wide changes to the `PATH` should be done by adding files to `/etc/paths.d`.

As an administrator you should be on the lookout for scripts and installers that attempt to modify configuration files on the user level, disable the scripts during deployment, and manage the required changes centrally. This will allow you to keep control of the settings even as tools change, are added or removed from the system, while preserving the user's custom configurations.

To manage environment variables other than `PATH` centrally, administrators should consider `/etc/zshenv` or adding to the existing `/etc/zshrc`. In these cases you should always monitor whether updates to macOS overwrite or change these files with new, modified files of their own.

Summary

There are many possible files where the `zsh` can load user configuration. You should use `~/.zshrc` for your personal configurations.

There are many tools and projects out there that will configure `zsh` for you. This is fine, but might keep you from really understanding how things work.

MacAdmins who need to manage these settings centrally, should use `/etc/paths.d` and similar technologies or consider `/etc/zshenv` or `/etc/zshrc`.

Apple's built-in support for `zsh` in Terminal is not as detailed as it is for `bash`.

Next: Part 3 – Shell Options
