Help Topic: Customizing the Shell Environment

Spring 2024

Michael Benjamin, mikerb@mit.edu
Department of Mechanical Engineering
MIT, Cambridge MA 02139

How to Customize Your Shell Environment

The command line experience is driven by a special program called the shell. The shell utilizes a number of common commands such as `ls`, `rm`, `mkdir` and so on. It turns out, there are a number of different shells to choose from. While they are all very similar, they have important differences, slightly different syntax, and users tend to have a strong allegiance to their choice of a shell. Perhaps the most common choices are the *bash*, *zsh* and *tcsh* shells. The bash shell is what is typically running out-of-the-box upon a GNU/Linux install. MacOS by default runs *zsh*, but we configured it to run *bash* instead in the steps discussed earlier and here:

http://oceanai.mit.edu/ivpman/help/osx_terminal

An important part of using the command line involves the customization of your shell to make it support your individual work style. In this help topic we:

- Explain how you can tell which shell you are presently using
- Introduce shell configuration files
- Provide example bash and tcsh shell configuration files
- Introduce custom prompts

The bash and tcsh executables are typically found in `/bin/bash` and `/bin/tcsh`. At any time you can just launch a new shell from the command line:

```
$ /bin/bash
```

This has the effect of perhaps switching shell types if you were using another shell, and it clears the local shell history (another topic).

What shell am I using?

Before augmenting your shell path, you need know which shell you're running. In most cases, it is either bash or tcsh. One way to find out:

```
$ echo $0
bash                        (should return with either bash or tcsh)
```

Take note of the answer and proceed.

Configuration files

Regardless of which shell you use, custom configuration is done with one or two key configuration files stored in the user's home directory. When the shell is launched, the file is read, and customizations are in place. Typically each line in the configuration file uses the same format as if the user typed them all directly at the start of their session.

If you have N terminal or console windows open, then you have N separate shell sessions in operation. They all read the same configuration file(s) upon startup. If you do make a change to the shell environment from the command line, e.g., define an alias, change an environment variable, it will only be in effect in the shell session where you typed it in. If you want that effect to be carried over across all future shell sessions, this needs to be reflected in the shell configuration file(s).

Example shell configuration files for bash users

Bash typically is configured with the `.bashrc` and `.bash_profile` files in the user's home directory. For a discussion on why both files are used, see here:

http://www.joshstaiger.org/archives/2005/07/bash_profile_vs.html

Example `.bash_profile` and `.bashrc` files can be found by pointing your web browser to the below URLs and copying and pasting what you want into your own files.

http://oceanai.mit.edu/ivpman/docs/.bash_profile

http://oceanai.mit.edu/ivpman/docs/.bashrc

Or just use `wget` to download copies directly into your home directory:

```
$ cd                                                   // go to your home directory
$ wget http://oceanai.mit.edu/ivpman/docs/.bash_profile // get the file
$ wget http://oceanai.mit.edu/ivpman/docs/.bashrc       // get the file
```

Customize Your Shell Prompt

One customization I can't live without is the customization of my shell prompt. By convention when we indicate a command typed on the command line, we would show something like:

```
$ cd ../
```

The dollar sign character is the *shell prompt*. In practice, people commonly customize this with something like:

```
leonardo:Research/project$ cd ../
```

Everything to the left of the dollar sign is a customization. In this case `leonardo` is the name of the machine, and `Research/project` is the name of current directory. Actually the full name of the current directory, may be too long to want on your shell prompt, so only two levels are shown in the prompt. This is my personal favorite configuration, but you can choose one of your own. For me, I always want to know what machine I'm on because I'm frequently remotely logging in to other machines or robots. And I always want some indication of which directory I'm in and showing two levels is enough to do the trick.

Next we'll show how this kind of configuration can be done for both bash and tcsh. Regardless of how you configure yours, I definitely recommend taking this step. It's extremely useful in practice and will cut down common errors due to not knowing what directory or machine you're on when you type a command.

Customize Your Shell Prompt for bash Users

If you're using bash and you took the example `.bashrc` file provided above, then you already have customized your shell prompt. It was done with the line:

```
  PS1="\H:\W\$ "
```

The
`H` will expand to the machine name, and the
`W` will expand to the last component of the present working directory. The
`$` simply puts the dollar sign at the end of prompt. A convention is to use the dollar sign for bash, and the percent sign for tcsh.

There are many other ways to configure your prompt to your liking, such as including the time, day of the week, and so on. A google search on "customize bash prompt" will give you all you need.

Customize Your Shell Prompt for tcsh Users

If you're using tcsh and you took the example `.cshrc` file provided above, then you already have customized your shell prompt. It was done with the line:

```
  set  prompt="%m:%C2%% "
```

The `%m` will expand to the machine name, and the `%C2` foobar will expand to the last two components of the present working directory. The `%%` simply puts the percent sign at the end of prompt. A convention is to use the percent sign for tcsh and the dollar sign for bash.

There are many other ways to configure your prompt to your liking, such as including the time, day of the week, and so on. A google search on "customize tcsh prompt" will give you all you need.