# Help Topic: Augmenting Your Shell Path

**Spring 2022**

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

## Adding a Directory to Your Shell Path

The *shell path* is a list of locations your shell program knows about to look for programs to run. When you type:

```
$ pwd
```

or

```
$ du
```

The `pwd` command is located in the `/bin` directory and `du` is located in the `/usr/bin` directory. These locations are conventional locations most shells users have in their shell paths. But if you add your own code (as we will in this class) you'll need to know how to augment your shell path to tell your shell where the new programs reside.

Adding a directory to your shell path is a very common thing command line users need to do. Unfortunately the need often comes up for new users while they're still finding their feet with the command line, and before they know what a shell *environment* is, and sometimes before they even know how to use a text editor. Here we focus on the minimal steps for adding a new directory to the shell path, perhaps glossing over a few things for the sake of getting it done.

### Make Sure You Know How to Edit a Text File

The next step involves a simple edit to a text file. If you know how to do this proceed to the next step.

If you don't know what a text file is, vs. say a Microsoft Word document which just has text, then you may not know what a *text editor* is either. Wikipedia has a good entry on this topic. I recommend giving it a skim.

The good news is that you very likely already have a text editor on your machine. On the Mac it is an application called `TextEdit`, in the `/Applications/` directory. In GNU/Linux, it is a called `gedit` in the `/usr/bin` directory. And, if for some reason those directories are not in your shell path, don't despair. You can launch these editors by typing their full path name, or double clicking on them in a GUI like Finder on the Mac.

The bad news is that we really don't want to be using editors like `gedit` or `TextEdit` in our class.

And we certainly don't encourage people to launch a text editor from a GUI. We want to do this all from the command line, and want to use a more powerful text editor like `Emacs` or `vi`. But for now, do what you need to do, just so long as you can make simple edits to a text file so we can get through the next few steps below.

If you really do want to hit the pause button here and go get Emacs and learn just enough to make and save simple edits, then check out the first couple emacs help topics on the main help page.

The main help page:
https://oceanai.mit.edu/ivpman/help

The Emacs Editor help page for MacOS users:
https://oceanai.mit.edu/ivpman/help/emacs_get_for_osx

## Augmenting the Shell Path for `bash` Users

If you're a bash user, your configuration is done in a couple files in your home directory: `.bash_profile`, and `.bashrc`. Use `ls -a` to see them since they are hidden files. If you don't have these files, I recommend downloading the example files described earlier, or on the help page:

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

Otherwise, if you already have the `.bashrc` file, edit the file and put the following two lines at the end of the file.

```
PATH=$PATH:/my/full/directory/
export PATH
```

## Verifying Success

To verify success, pick one of the executables in the directory you added to the shell path and use the `which` tool:

```
$ which cmake
/opt/local/bin
```

Note the returned value may also be something like `/usr/local/bin/cmake`, or `/bin/cmake`, or `/usr/bin/cmake`, depending on if your system is MacOS or GNU/Linux and depending on the particular release of MacOS or GNU/Linux. The important measure of success is that the above command does just return without an answer.