# Overcoming your Fear of the Command Line

10TH JUN 2015

Most tools that help you automate your workflow require the use of the command line. Hence, the first obstacle you have to overcome is getting comfortable with the command line.

But the command line is scary.

Playing with it feels like you're dismantling a bomb that could go off any moment. One wrong move and that'll mean the end of your life, and your computer.

I didn't dare to touch it when I first began to code. I felt that the command line was a tool that only experts could use.

However, as I got to know more about it, I began to realize that the command line isn't scary at all! It's incredibly safe, even for beginners, and anyone can use it to help improve their workflow.

In this article I'll show you why the command line isn't that scary, and how to start to get comfortable with it.

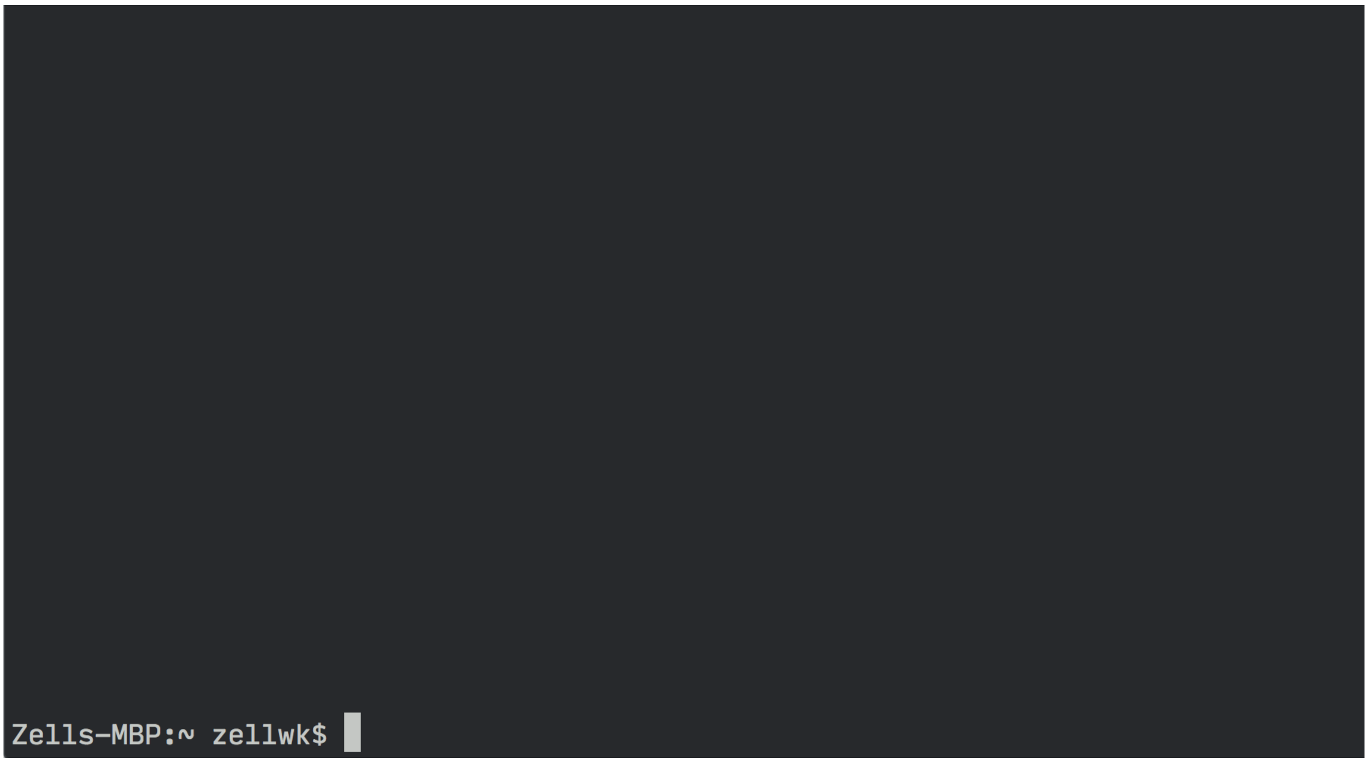
Ready to go? Let's begin!

## Starting Up the Command Line

The command line is a program that takes in written commands and performs them with your operating system.

Your operating system should have a program built in to run the command line. It's called the Terminal on the Mac, and the Command Prompt on Windows. Fire that up and you're already looking at the command line (Note: Windows users might want to use a command line emulator like Cmder instead).

Here's how it looks like on a Mac



```
Zells-MBP:~ zellwk$ 
```

You don't see anything you can do with it, there's no step by step instructions you can follow and everything you type in seems to return an error.

No wonder it's so scary!

Well, don't worry.

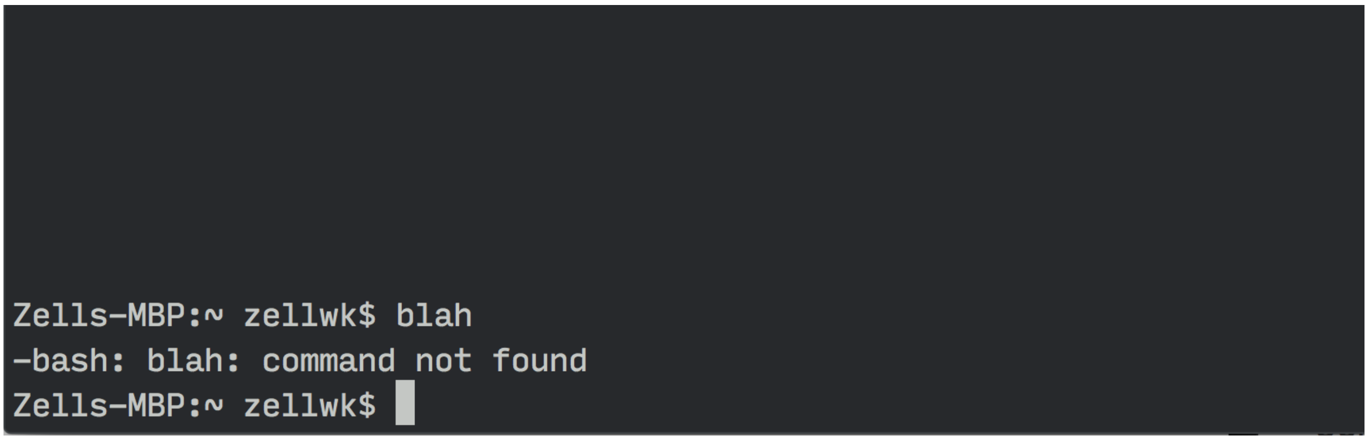
## Nothing you do will break your computer

Even if you entered multiple invalid commands.

When you enter an invalid command, all the command line does is to show you an error message, then do nothing.

Here's an example if what it does if you enter an invalid command:

```
$ blah
```



```
Zells-MBP:~ zellwk$ blah
-bash: blah: command not found
Zells-MBP:~ zellwk$ 
```

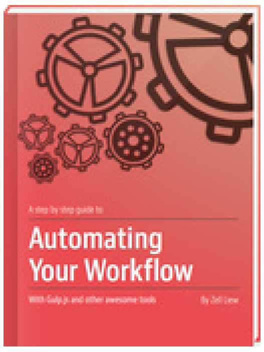
You'll get is a "command not found" error message, then nothing happens. You're still safe and your computer didn't explode.

The only command you need to be wary of is the `rm` command. This means to remove files permanently, which deletes your file and leaves it nowhere to be found, not even in your trash.

Next, you'll want to learn a few commands that you can use with the command line. It comes with a big list of commands, but you'll only need to know 6 of them.

(Before we move on, here's a quick signup form if you'd like to read similar articles from me every Wednesday).

## Save 2 hours a day on web development



Have you had days where it's suddenly 5pm, and you haven't done anything constructive yet? Me too. It's been like this forever and I'm sick of it.

One day, I found out that I'm spending too much time on repetitive tasks. I decided to learn how to automate them away. After doing so, I now have more time to write, design and learn new stuff every single day.

That's what I'm going to teach you in Automating Your Workflow. **Start reclaiming your free time with 10 free chapters**.

| First Name | Email Address |
|---|---|
| | |

Send me the chapters!

## The 6 commands you need to know

The 6 commands you need to know are:

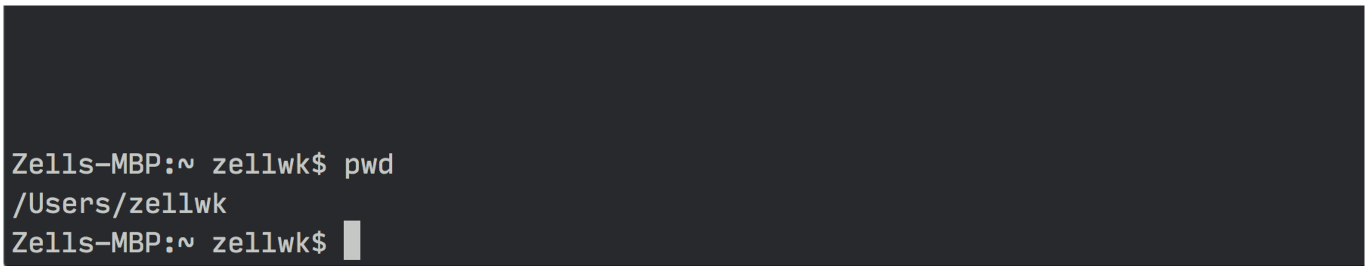
1. pwd
2. cd
3. ls
4. mkdir
5. touch
6. clear

Let's go through them one by one.

### PWD

`pwd` means print working directory. All it does is to let you see the location you are at in the terminal.

```
$ pwd
```

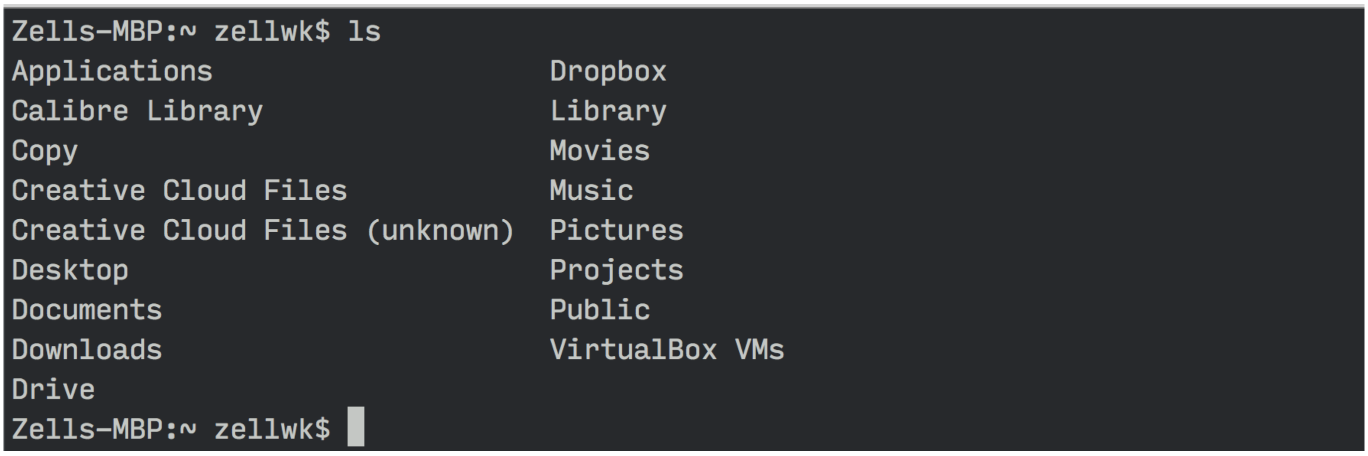


```
Zells-MBP:~ zellwk$ pwd
/Users/zellwk
Zells-MBP:~ zellwk$
```

## LS

`ls` means list files. If you enter this command you'll get a list of all the files and folders that is in your current location.

```
$ ls
```



```
Zells-MBP:~ zellwk$ ls
Applications                    Dropbox
Calibre Library                 Library
Copy                            Movies
Creative Cloud Files            Music
Creative Cloud Files (unknown)  Pictures
Desktop                         Projects
Documents                       Public
Downloads                       VirtualBox VMs
Drive
Zells-MBP:~ zellwk$
```

In this case, you'll see that I have folders such as Desktop, Music, Pictures and Library within my current working directory.

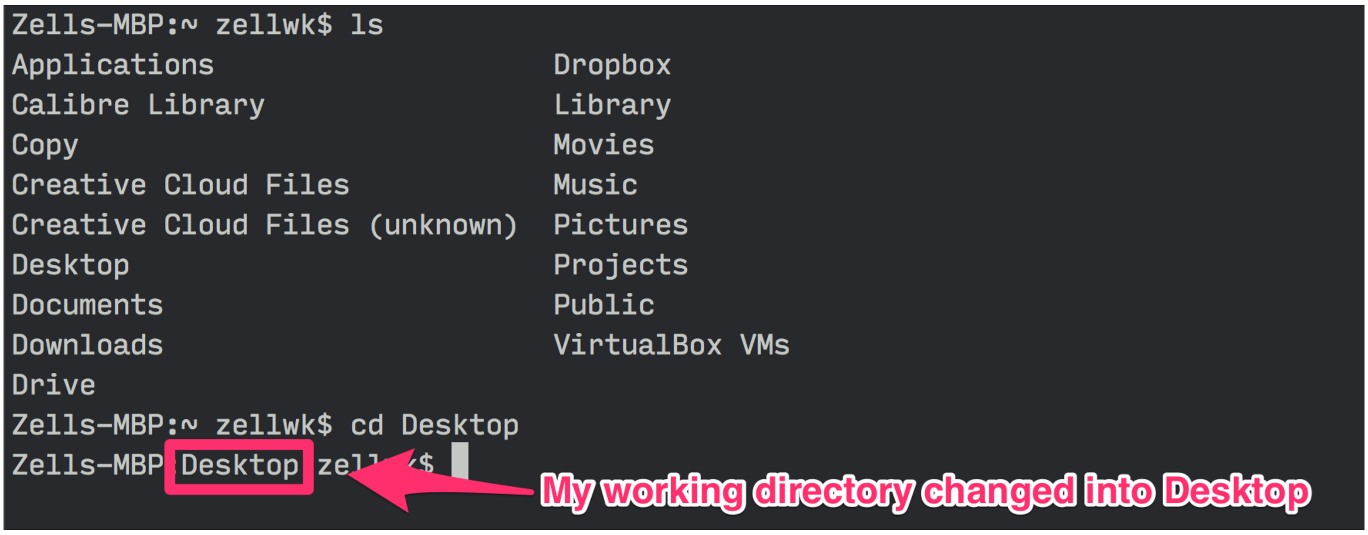
This information is good knowledge for the next command you'll use.

## CD

`cd` means change directory. It allows you to change the current directory to different folders. It is the most used command of all.

When combined with the `ls` command, you'll be able to see the folders you can navigate to. If I wanted to navigate to Desktop, all I have to do is to write this:

```
$ cd Desktop
```

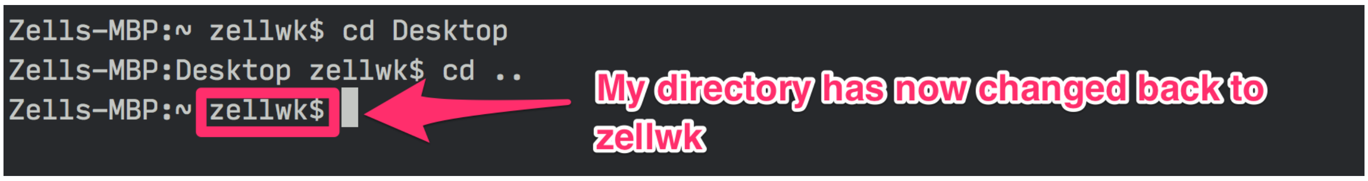


```
Zells-MBP:~ zellwk$ ls
Applications                     Dropbox
Calibre Library                  Library
Copy                             Movies
Creative Cloud Files             Music
Creative Cloud Files (unknown)   Pictures
Desktop                          Projects
Documents                        Public
Downloads                        VirtualBox VMs
Drive
Zells-MBP:~ zellwk$ cd Desktop
Zells-MBP:Desktop zellwk$
```

My working directory changed into Desktop

If you wanted to go back up a directory, you'll just have to type in `..` after `cd`.

```
$ cd ..
```

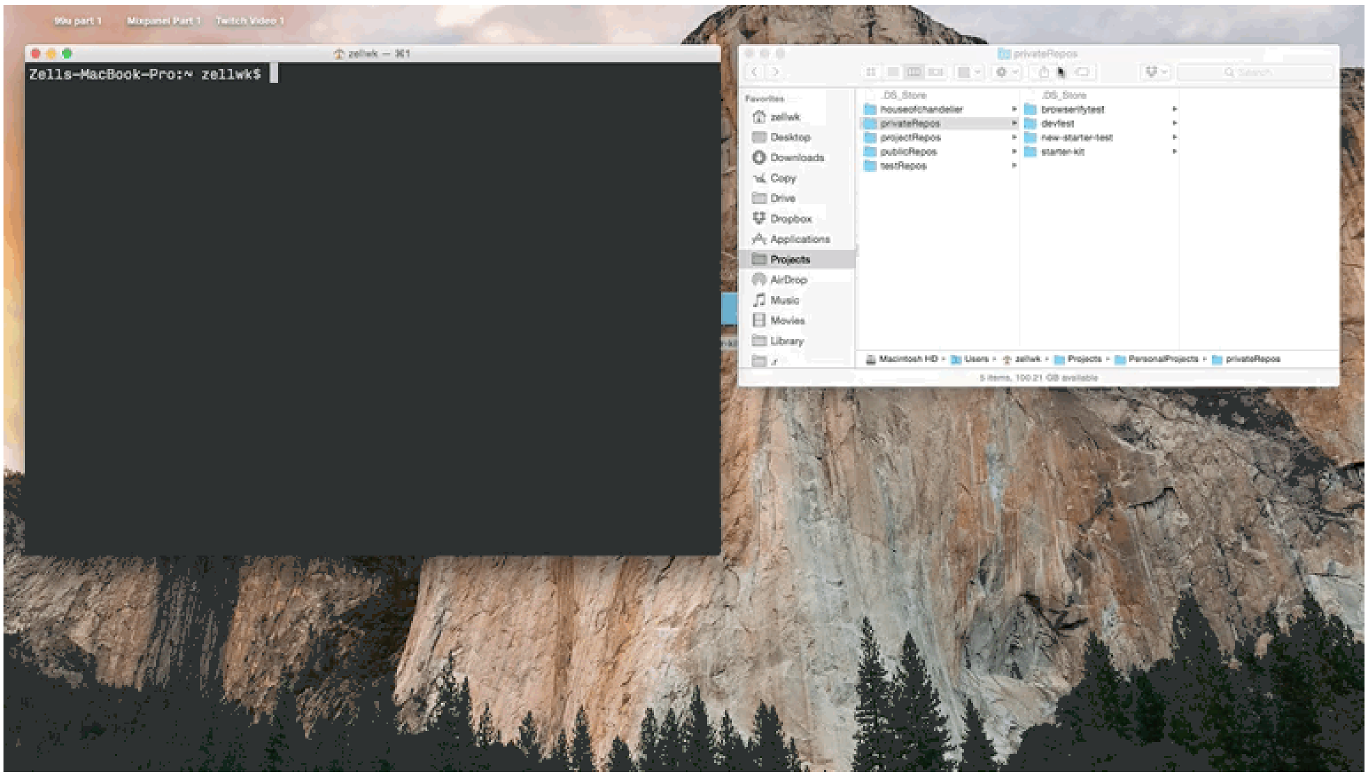


```
Zells-MBP:~ zellwk$ cd Desktop
Zells-MBP:Desktop zellwk$ cd ..
Zells-MBP:~ zellwk$
```

My directory has now changed back to zellwk

You can repeat this `ls`, `cd` combination to get to any folder you want to get to.

Here's a neat thing. You can type cd and drag any folder on the mac into the terminal, and it'll fill up the correct path for you.



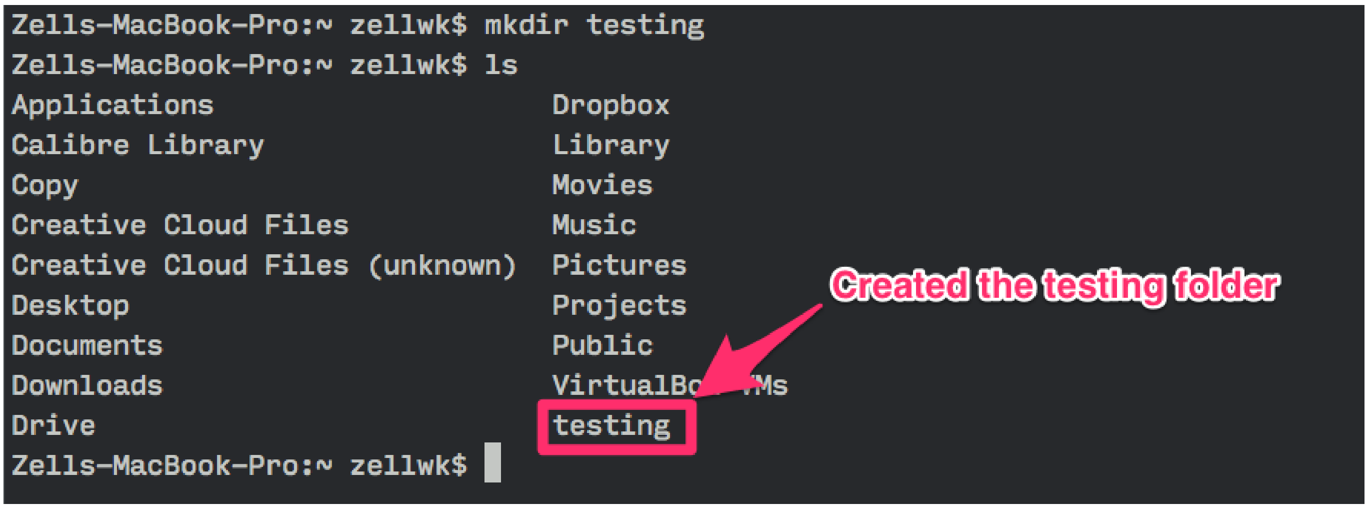
```
Zells-MacBook-Pro:~ zellwk$
```

## MKDIR

`mkdir` means make directory. It's the same as creating a new folder by right clicking with your mouse and selecting create new folder.

```
$ mkdir testing
```

This creates the testing directory.



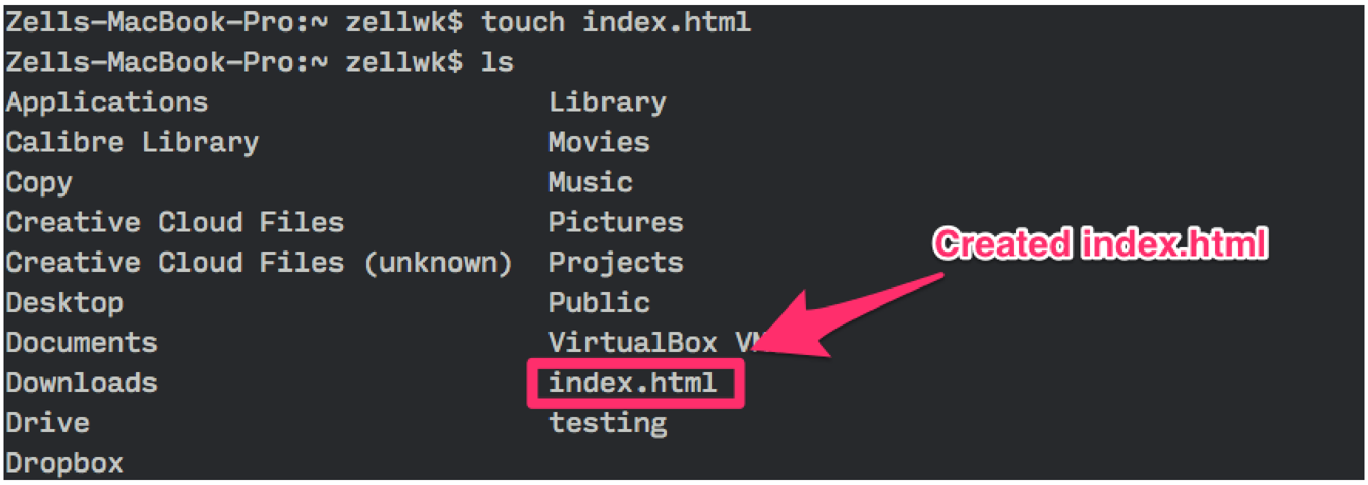
```
Zells-MacBook-Pro:~ zellwk$ mkdir testing
Zells-MacBook-Pro:~ zellwk$ ls
Applications                      Dropbox
Calibre Library                   Library
Copy                              Movies
Creative Cloud Files              Music
Creative Cloud Files (unknown)    Pictures
Desktop                           Projects
Documents                         Public
Downloads                         VirtualBox VMs
Drive                             testing
Zells-MacBook-Pro:~ zellwk$
```

## TOUCH

Touch is the command to create a file. You can create any kind of file with touch.

```
$ touch index.html
```

This means to create a file named `index` with the extension `html` in the current working directory.

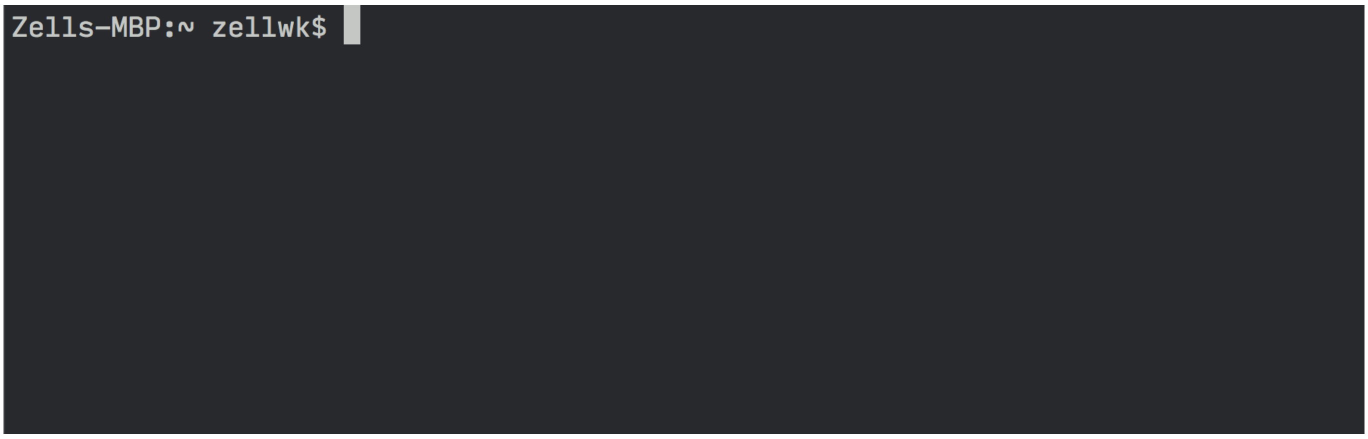


```
Zells-MacBook-Pro:~ zellwk$ touch index.html
Zells-MacBook-Pro:~ zellwk$ ls
Applications                      Library
Calibre Library                   Movies
Copy                              Music
Creative Cloud Files              Pictures
Creative Cloud Files (unknown)    Projects
Desktop                           Public
Documents                         VirtualBox VMs
Downloads                         index.html
Drive                             testing
Dropbox
```

## CLEAR

`clear` means to clear the terminal screen. It'll remove all the clutter you have on the screen and revert it back to the clean state you had when you open the terminal.

```
$ clear
```



```
Zells-MBP:~ zellwk$
```

There you go, the 6 commands you'll need to know to use the terminal effectively.

And this point you may be wondering about other commands like `git` , `bower` , `npm` , `gulp` and all other kinds of commands you see around the internet.

The thing is these commands are not native to the command line. You'll wont be able to use them unless you have installed their command line interfaces (CLIs).

This means you don't have to worry about these commands for now. You can pick them up one by one as you learn more about them.

But even if you make mistakes with these additional commands, you're still safe. Let's take a look at an example with the `git` command

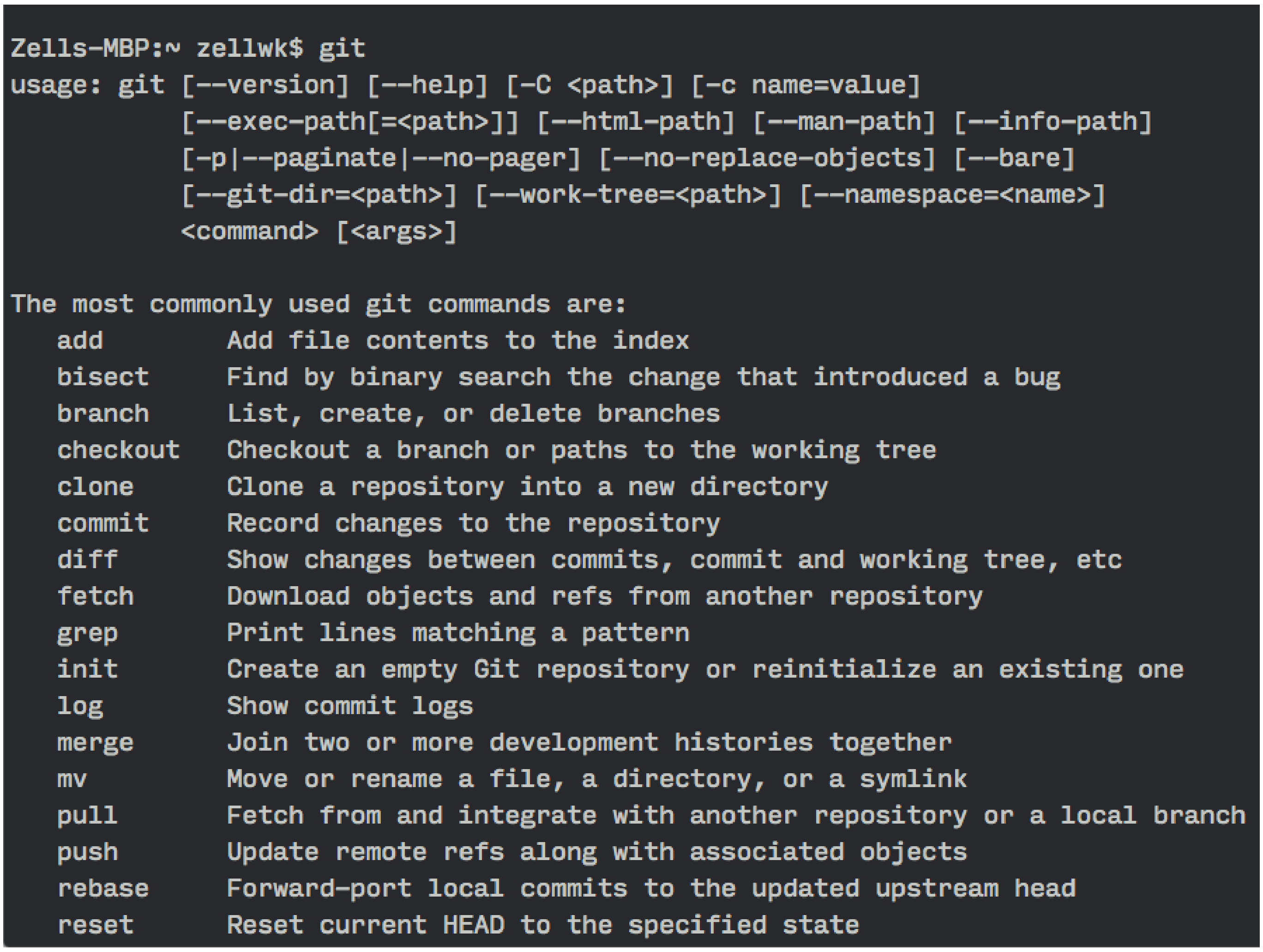
## Addtional commands from other CLIs

To use the `git` command, you'll need to make sure that Git is installed on your machine.

Git is already installed on the Mac, so Mac users can skip the installation step. Windows users will have to install Git though.

Once you have Git installed, you can type commands that Git provides. You'll see a scary bunch of text if you type in an invalid command:

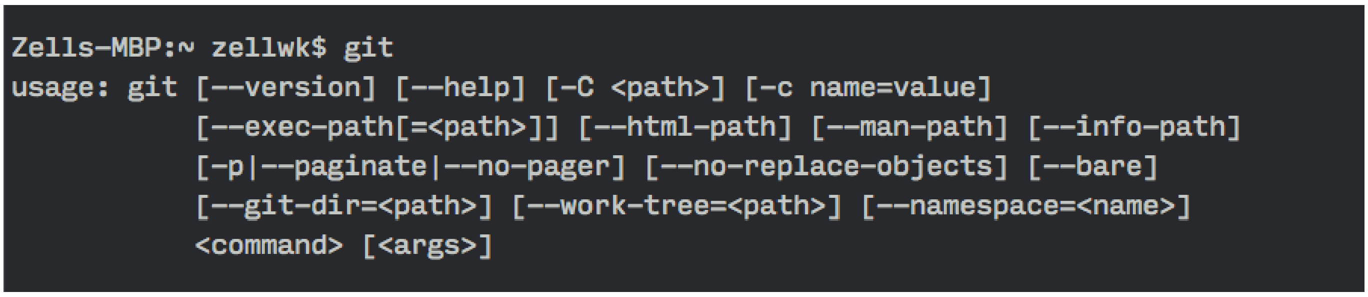
```
$ git
```



```
Zells-MBP:~ zellwk$ git
usage: git [--version] [--help] [-C <path>] [-c name=value]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p|--paginate|--no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

The most commonly used git commands are:
   add        Add file contents to the index
   bisect     Find by binary search the change that introduced a bug
   branch     List, create, or delete branches
   checkout   Checkout a branch or paths to the working tree
   clone      Clone a repository into a new directory
   commit     Record changes to the repository
   diff       Show changes between commits, commit and working tree, etc
   fetch      Download objects and refs from another repository
   grep       Print lines matching a pattern
   init       Create an empty Git repository or reinitialize an existing one
   log        Show commit logs
   merge      Join two or more development histories together
   mv         Move or rename a file, a directory, or a symlink
   pull       Fetch from and integrate with another repository or a local branch
   push       Update remote refs along with associated objects
   rebase     Forward-port local commits to the updated upstream head
   reset      Reset current HEAD to the specified state
```

Let’s look at it part by part so it’s not so overwhelming. Here’s the first part:



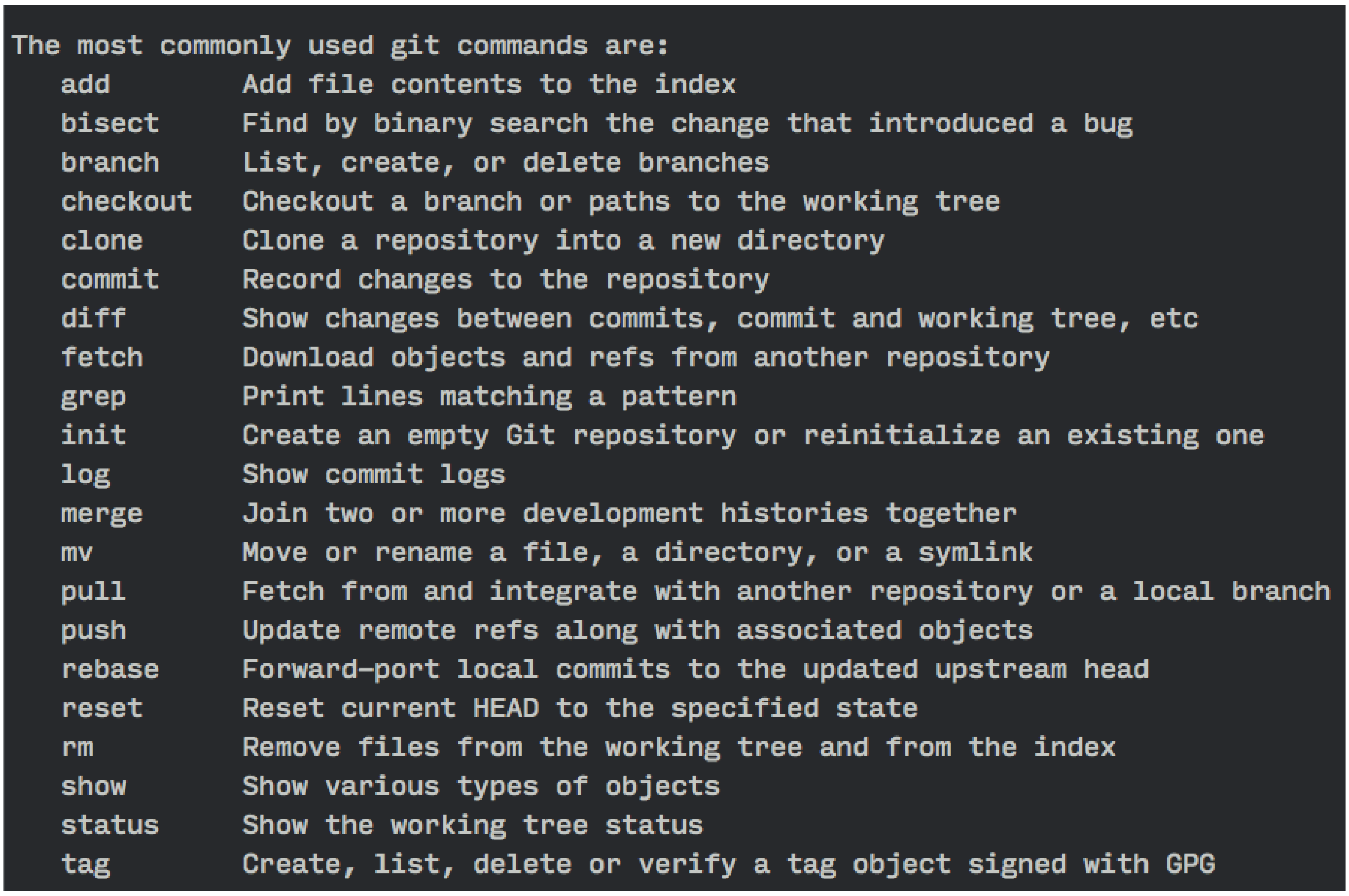
```
Zells-MBP:~ zellwk$ git
usage: git [--version] [--help] [-C <path>] [-c name=value]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p|--paginate|--no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]
```

Here we can see that the `git` command doesn’t work alone. You’ll have to accompany it with either an argument or a command.

So commands like these would work:

```
$ git --version # shows you the version of git installed
$ git --help # shows git help
```

The second part of this huge chunk tells you the common commands that people use with Git.

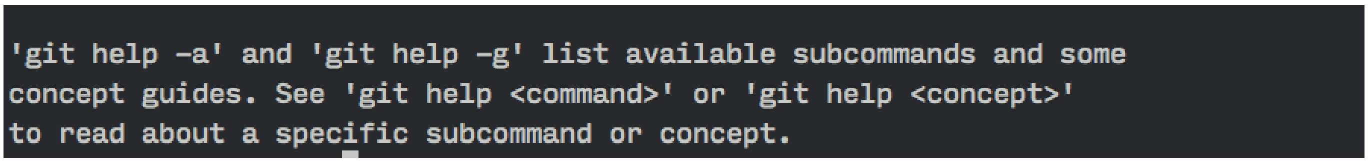


```
The most commonly used git commands are:
   add        Add file contents to the index
   bisect     Find by binary search the change that introduced a bug
   branch     List, create, or delete branches
   checkout   Checkout a branch or paths to the working tree
   clone      Clone a repository into a new directory
   commit     Record changes to the repository
   diff       Show changes between commits, commit and working tree, etc
   fetch      Download objects and refs from another repository
   grep       Print lines matching a pattern
   init       Create an empty Git repository or reinitialize an existing one
   log        Show commit logs
   merge      Join two or more development histories together
   mv         Move or rename a file, a directory, or a symlink
   pull       Fetch from and integrate with another repository or a local branch
   push       Update remote refs along with associated objects
   rebase     Forward-port local commits to the updated upstream head
   reset      Reset current HEAD to the specified state
   rm         Remove files from the working tree and from the index
   show       Show various types of objects
   status     Show the working tree status
   tag        Create, list, delete or verify a tag object signed with GPG
```

One of these commands is the `checkout` command, where you can check out a branch. Here's an example of how you can use it:

```
$ git checkout development # checks out the development branch
```

Finally, the last part simply tells you to enter the commands if you need more help:



```
'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
```

As you can see, **nothing broke even when you entered an invalid command**. The command line is smart enough to prompt you that something is wrong, and ask you to correct your commands.

## Wrapping Up

We've covered the basics of the command line and you've seen why there's nothing to be afraid of. We also took a look at other CLIs with `git` as an example. I hope this article has helped you overcome your fear with the command line.

Don't worry if you don't understand what the `git` commands do right now. Every CLI has its own set of commands and you'll have to learn them by reading through their documentation or viewing tutorials on the web.

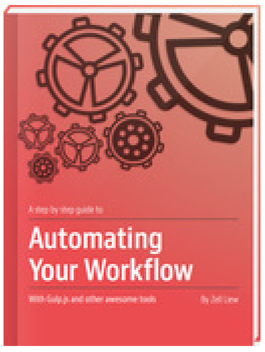
What's important right now is for you to get your feet wet with the command line and make sure you're comfortable typing in it.

One more thing. If you want to up your command line game, I highly suggest checking out the free command line video series by Wes Bos.

Try playing around with the 6 commands mentioned in the article – `pwd`, `ls`, `cd`, `mkdir`, `touch` and `clear` and let me know how you feel about the command line in the comments section below!

Thanks for reading. Did this article help you in any way? If you did, I hope you consider sharing it. You might help someone out. Thank you!

## Save 2 hours a day on web development



Have you had days where it's suddenly 5pm, and you haven't done anything constructive yet? Me too. It's been like this forever and I'm sick of it.

One day, I found out that I'm spending too much time on repetitive tasks. I decided to learn how to automate them away. After doing so, I now have more time to write, design and learn new stuff every single day.

That's what I'm going to teach you in Automating Your Workflow. **Start reclaiming your free time with 10 free chapters**.

First Name

Email Address

Send me the chapters!

**15 Comments** **Zell's Blog** 1 **Login**

Recommend Share Sort by Newest

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

**sammycat1024** • a year ago

wonderful articles.

Reply • Share ›

**Ben Bright** • a year ago

Thanks so much Zell for this. Really appreciate.

Reply • Share ›

**Dorothee Bächle** • 3 years ago

Ahhhhhh, Thank you Zell, finally someone told me the secrets of command lines. Whow ,and I wonder what ever says bower :-) after all, now I sense,thanx!!!

Reply • Share ›

**Zell Liew** Mod → Dorothee Bächle • 3 years ago

Here's whatever that says bower: http://www.zell-weekeat.com... :)

Reply • Share ›

**Dorothee Bächle** → Zell Liew • 3 years ago

thank you Zell

Reply • Share ›

**martinj** • 3 years ago

Thank you for this article Zell.

Reply • Share ›

**Zell Liew** Mod → martinj • 3 years ago

you're welcome

Reply • Share ›

**Paweł Grzybek** • 3 years ago

Instead of 'clear' command i prefer to use 'Cmd + K'.
'ls -a' seems to be very useful for me as well - displays all files including hidden.

Reply • Share ›

**Zell Liew** Mod ➦ Paweł Grzybek • 3 years ago
I recently got to know about cmd + k as well! Such a useful shortcut :)

ls -a. Point taken. Thanks :)
^ | v • Reply • Share ›

**Nicolás Danelón** • 3 years ago
nice article
^ | v • Reply • Share ›

**Zell Liew** Mod ➦ Nicolás Danelón • 3 years ago
Thanks. Glad you enjoyed it :)
1 ^ | v • Reply • Share ›

**Hunbei** • 3 years ago
A good article too!This article introduce The bacis of Workflow. I think you are intelligent and a good man. You are taking care of most of people which starts do develop projects and help them overcome Cammend line. I like this teaching method----Teach knowleage from zero.. But my operating system is Windows.
^ | v • Reply • Share ›

**Zell Liew** Mod ➦ Hunbei • 3 years ago
You can work with this on windows too :)
^ | v • Reply • Share ›

**Yahia M.** ➦ Hunbei • 3 years ago
Am using Windows too and it works just fine. But I'd recommend using different Terminal (Command Line) like ConEmu. Am using it and happy with it, because you can open as many new tabs as you want and that will come handy in the next stages of your development workflow and much more...

And thank you @Zell Liew for the post really great. Can't for your next ones.
^ | v • Reply • Share ›

**Zell Liew** Mod ➦ Yahia M. • 3 years ago
Thanks for chipping in Yahia :)
^ | v • Reply • Share ›

ALSO ON **ZELL'S BLOG**

**Useful VS Code keyboard shortcuts**
3 comments • 6 months ago
Giovanni Pires da Silva — If someone else looks for the same feature, Code's File > Open Workspace or FIle > Open Recent >

**How to build a calculator—part 3**
1 comment • 5 months ago
Nicolás Danelón — will this end in a electron app?

**Traversing the DOM with JavaScript**
14 comments • 5 months ago
Zell Liew — Well, with this, you don't need to

**Case study—a project from hell**
3 comments • 5 months ago
Gregor Arz — Hi ZellBut I don not get the

include the jQuery library. :)

point that the company has not paid the work you did even ask for a discount from your

## Need help with your projects?

Hit the button below and tell me more! I'd love to help :)

Hire Zell for my project