6.9 Find

It's often the case that you wish to find a specific file, but don't exactly remember where you had saved it. Most modern machines have a method of finding something on your computer such as the search bar in the start menu of a windows machine or the spotlight search of a mac. Linux command line lets us do something very similar with the `find` command. There are many ways to use `find`, but a common form is:

```
find startingLocation -name someName
```

Find allows for you to search by direct filename, or by using file expansions.

Direct Searches

To search for a file directly, you can use its filename alone:

```
find . -name file
```

> find: './2750Materials/module4/ch6/unreadableDirectory': Permission denied
>
> find: './2750Materials/module4/ch6/unexecutableDirectory/subdirectory': Permission denied
>
> find: './2750Materials/module4/ch6/unexecutableDirectory/readableScript.sh': Permission denied
>
> ./2750Materials/module4/ch6/file
>
> ./test/file

If find doesn't have permissions to view a file or its subdirectories, it will be sure to let you know in its output as well.

Using Expansions

It's unlikely that you'll find yourself remembering the exact name of a file that you wish to find. To find a file that matches a specific pattern, you can use expansions:

```
find . -name *fire*
```

> ./testDirectory/fire ./processExpansions/directory_2/fire1 ./processExpansions/directory_2/fire2

Executing Commands on Found Files

Find is a powerful tool with many additional flags to use, but one of the most powerful flags at our disposal is the `-exec` flag. Suppose you wished to find all of your shell scripts:

```
find . -name '*.sh'
```

> ./takeforever.sh
>
> ./takesArguments.sh
>
> ./fifteenBlocks.sh
>
> ./badif.sh
>
> ./simpleWriteToFile.sh
>
> ./misc/bashScript.sh
>
> ./2750Materials/module1/ch3/takeforever.sh
>
> ./2750Materials/module1/ch3/failFast.sh
>
> ...
>
> ./basicScript.sh
>
> ./testFile.sh

Often times, you may wish to find a certain kind of file, but want to do something a bit more complex. We could write a shell script and expand the results of the `find` to loop over all of the file paths returned, so that we could then interact with the files. An easier method is using the built in `exec` flag for `find`.

```
find . -name '*.sh' -exec fgrep '#!/bin/bash' {} \;
```

> `#!/bin/bash`
>
> `#!/bin/bash`
>
> ...
>
> `#!/bin/bash`
>
> `#!/bin/bash`

The above syntax is relatively curious looking. We are executing the `fgrep` command in order to search every file that matches the `.sh` file extension for the `#!/bin/bash` header. The `{}` are needed for find to expand with its contents so that the `fgrep` can have some material to work on. The `\;` signifies the end of the statement. However, it's not always the case that you'll want the material of the executed command to be the only thing that prints. You can also use the `+` to append the output to the name of the file:

```
find . -name '*.sh' -exec fgrep '#!/bin/bash' {} +
```

> ./takeforever.sh:#!/bin/bash ./takesArguments.sh:#!/bin/bash ./badif.sh:#!/bin/bash ... ./failforever.sh:#!/bin/bash ./basicScript.sh:#!/bin/bash

You could additionally use the `-q` mode with `fgrep` and the `-print` flag with find:

```
find . -name '*.sh' -exec fgrep -q '#!/bin/bash' {} \; -print
```

Note that inside of the `exec` with `fgrep` the `-q` flag is an option for `fgrep` and not for find!

Searching by Mode

You can additionally search by `mode`:

```
 find . -perm 600
```

> ./.bash_history
>
> ./inputOutputDirectory/.someFile.swp
>
> ./watchMe
>
> ./.swp
>
> ./.viminfo

There are infinitely more options for the find command. Please refer to the man pages to read more about it!

6.10 Locate

The locate command is another command for finding specific files. Locate turns by searching from a database as opposed to attempting to search via directories.

6.11 Compressing and Distributing Files

Often times, you'll need to combine and compress a number of files before transferring them. A simple way to do this is by creating a collection via the `tar` command. `tar` combines all of the files into a single "file" format that can be unpacked into the standard directory format that we're used to. `tar` also allows for users to specify a type of compression by using a specific flag. Typically speaking, `tar` is run with the flags `c` / `x`, `v`, and `f` for `compress` / `extract`, `verbose` and `file`:

```
tar cvf touchedFiles.tar touch1 touch2 touch3
```

> touch1
>
> touch2
>
> touch3

The output of tar is the files combined to make it. If you wish to compress the files so they don't necessarily take up as much room on your disk/email/etc, you can use any number of the folloiwng flags:

- `z`: gzip compression (tar file's extensions is `tgz`)

  ```
  tar zcvf touchedFiles.tgz touch1 touch2 touch3
  ```

- `j`: bzip2 compression (tar file's extensions is `tbz`)

  ```
  tar jcvf touchedFiles.tbz touch1 touch2 touch3
  ```

- `J`: xz compression (tar file's extensions is `txz`)

  ```
  tar Jcvf touchedFiles.txz touch1 touch2 touch3
  ```

To untar a file, you need to have the proper compression flag, and substitute the `c` compress flag for the `x` extract flag:

```
tar xvf touchedFiles.tar
```

```
tar zxvf touchedFiles.tgz
```

```
tar jxvf touchedFiles.tbz
```

```
tar Jxvf touchedFiles.txz
```