Learning Objectives

Learners will be able to...
- Echo to the console and to a file
- Find files
- Get information about a file
- Concatenate files

Echoing to the console and to a file

In the last assignment we used the `touch` command to create an empty file. Now we'll learn an easy way to add content to a file.

Writing to the console using `echo`

The `echo` *bash shell* utility writes the result of the given inputs to the *standard output*.

Type:

```
echo "Hello World!"
```

You should see **Hello World!** printed out under where you typed the command since printing to the terminal is the standard output.

Writing to a file using `echo`

Echo lets you write to places other than standard output - you just have to specify where.

Try this command:

```
echo "Hello World!" > myfile.txt
```

The above sends the output of the echo command to `myfile.txt` rather than the console.

Breaking down the pieces of the above command:
1. The `echo` command gets a string argument *'Hello world!'*. This first argument is also called *standard input* (or *stdin* for short)
2. The `>` character or *stdout redirection operator* changes the *stdout* to be a file with the specified name rather than the console
3. The destination file will be created if it doesn't exist and overwritten if it does.

Type in the following two commands to see your file and list it out:

```
ls
cat myfile.txt
```

Concatenating files

On the last page we learned that the > character redirects the *stdout* to a file. It will create the file if it does not exist and overwrite it if it does.

Appending a file using the >> operator

The >> redirection operator does an append rather than an overwrite. It will create the file if it doesn't exist but if it does, it will append to it.

Let's try it out:

```
echo "I added this line!" >> myfile.txt
cat myfile.txt
```

cat myfile.txt command in terminal followed by output which says Hello World! on one line and I added this line! on the following line

We'll create another file with some text. We can still use the >> operator even though the file doesn't exist; it will create the file.

```
echo "First line in another file" >> anotherfile.txt
cat anotherfile.txt
```

Concatenating files using the >> operator

>> can also be used to combine two files:

1. The `cat` command lists out the contents of `anotherfile.txt`
2. The >> *redirection operator* appends it to the `myfile.txt` file, if the file exists.

Let's try it out:

```
cat anotherfile.txt >> myfile.txt
cat myfile.txt
```

```
codio@emotionmouse-gyrorose:~/workspace$ cat myfile.txt
Hello World!
I added this line!
First line in another file
```

The output of cat myfile.txt command printed out - Hello World! on one line, I added this line!on a second line, and First line in another file on the last line

The file `anotherfile.txt` is unchanged - you can list it out and check:

```
cat anotherfile.txt
```

Finding files

The `find` command looks for a file or directory based on the parameters you set.

The syntax of the find command is:

```
find [starting-point] [expression]
```

The `find` utility traverses the file system evaluating an *expression* or `argument` against each file and directory in the specified `source path`. In addition to searching for files and directories by name, you can search by permissions, users, groups, file types, date, and size.

First take a look at the `man` pages for the find command:

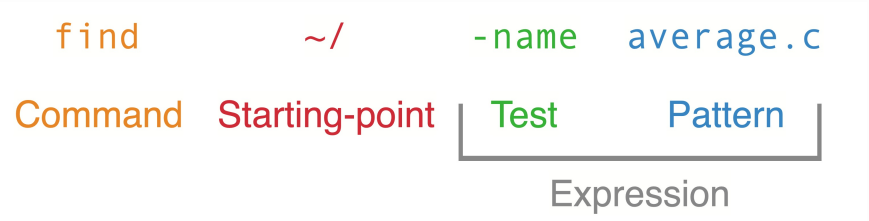
```
man find
```

It's a bit overwhelming, `find` is a versatile and powerful command! We'll show you a few examples so you can get a sense on how it would be used.

Type the following commands to get a sense of the versatility of the `find` command.

Finding a file by name

Take a look at this example of how to find a file by name:



1. The **starting-point** is `/home/codio/` - specified by `~/`
2. The **expression** is made up of two pieces: a *test* and a *pattern*
3. The *test* is indicated by `-name`, meaning we are checking file and directory names
4. The *pattern* we are looking for is `average.c`

Try finding the `average.c` file with the following:

```
find ~/ -name "average.c"
```

Finding files by other properties

Based on the manual page, try each of the following:

- Find files **or directories** with `average` in their name in the `/home` directory

  ▼

  Solution

  Instead of `-name` you will use the `-path` test.

  ```
  find /home -path *average*
  ```

  1. The starting-point is `/home`.
  2. The test is `-path`, it means that the file/directory name should match the pattern.
  3. Any file or directory containing the word `average` is what we are looking for. We use the `*` wildcard on either side to let there be any number of characters on either side of the word `average`.

- Find files in the home directory that are greater in size than 10 bytes.

  ▼

  Solution

  This time you'll use the `-size` test.

  ```
  find ~/ -size +10
  ```

  1. The starting-point is `/home/codio/`
  2. The test is `-size`.
  3. To specify **bigger than**, we need to put a `+` in front of the `10`

  Generally, you won't look for files based on bytes. You may specify `k` for kilobytes, `M` for megabytes next to the size (e.g. `find ~/ -size +10k`).

  If you are looking for a range, you can specify both a minimum and maximum file size. For example, to look for files between 10 and 50 bytes you would use `find ~/ -size +10 -size -50`

- Find files or directories in the `code` directory in your workspace that have been modified in the last year

  ▼

  Solution

  This time you'll use the `-mtime` test.

  ```
  find ~/workspace/code -mtime -365
  ```

  1. The starting-point is `/home/codio/workspace/code`
  2. The test is `-mtime`.
  3. Looking for any file modified between now and 365 days ago.

File information

There are two commands that are helpful for looking at file information (as opposed to contents).

Display file status using `stat`

The `stat` utility can be used to provide access information about a file or about the file system where it resides.

First take a look at the `man` pages for the `stat` command:

```
man stat
```

Notice the general syntax at the top is `stat FILE`.

Try the `stat` command:

```
stat code/src/average.c
```

challenge

What happens if you add the `-f` parameter, as in:

```
stat -f code/src/average.c
```

▼ The `-f` parameter

When you use the `-f` parameter you get system information rather than file information

Determine file type using `file`

The `file` utility can be used to get information about a file's type.

First take a look at the `man` pages for the `file` command:

```
man file
```

Similar to `stat`, the general syntax is `file FILE`.

Try the `file` command

```
file code/src/average.c
```

challenge

What other information do you get when you look at a binary file?

Try:

```
file code/bin/average
```

▼ Viewing information about a binary file

This file is an executable, you can see which system it is built for and information about the linker among other things.

Finding file content

The `grep` utility searches a file for a pattern or string and outputs all lines that match to stdout.

First take a look at the `man` pages for `grep` to get a sense on how it would be used:

```
man grep
```

The general syntax is pretty simple (`grep PATTERN`) but take a look at some of the helpful OPTIONS under Matching Control (e.g. -w and -i).

Try out the `grep` command:

```
grep -w printf code/src/*.c
```

grep command with the w option specifying the pattern as printf with the file/directory to be searched set to code/src/*.c

1. The `-w` option forces a whole word match.
2. The string `printf` is what we are searching for.
3. Searching in the relative path `code/src/` for all `.c` files

Try these variations

- Find all lines that do not contain `printf` in the c files in the `code/src/` directory.

  ▼

  Solution

  The `-v` option means do not contain.

  ```
  grep -v printf code/src/*.c
  ```

- Find all lines that contain `concat`, regardless of case, search in `code/docs`.

  ▼

  Solution

  The `-i` option means regardless of case.

  ```
  grep -i 'concat' code/docs/*
  ```

- Output the file names of files that contain `concat`, search in `code/docs`.

  ▼

  Solution

  The `-l` option means output file name.

  ```
  grep -l 'concat' code/docs/*
  ```