Learning Objectives

Learners will be able to...
* Use the following file operations:
- mv
- cp
- mkdir
- rm
- rmdir
- tree
- touch

Review

In the last assignment we learned a few key commands:
Command	Description
———	———-
`ls`	Lists the contents of a directory
`pwd`	Print working directory
`cd` *directoryname*	Change directory to *directoryname*
`.`	Indicates current directory
`..`	Indicates parent directory
`~`	Indicates home directory
`cat` *filename*	Lists the contents of *filename*
`man` *commandname*	Will provide the manual for most commands

info

Notice the command prompt has changed. Your username is the same but the unique identifier for your Codio server has changed. Each assignment has its own server address.

Creating directories and files

Type `ls -a` in the terminal window. You should see something similar to the image below - the unique Ubuntu server name will be different.

the ls -a command on the first line of a terminal resulting in . .. and .guides being printed on the following line

We have already discussed the `.` - current directory and `..` - parent directory. The `.guides` folder is where the text, images and other materials for this assignment reside. We recommend that you do not change anything in the `.guides` directory as it can interfere with the lesson as well as your scores.

We'll start out by creating some files and directories.

info

File and directory names

You can name the files and directories you create anything, you don't need to use our suggestions.

Using `mkdir` to make directories

Type the `mkdir` command in the terminal window followed by the directory name as shown below:

```
mkdir test
```

Now if you type `ls` or `ls -a` you will see the directory you created:

the ls -a command on the first line of a terminal resulting in . .. .guides and test being printed on the following line

For the next step we'll create a few files in the new directory.

Switch to the new directory using `cd test`.

Using `touch` to make files

The `touch` command creates new, empty files.

Create a new file by typing the following into the terminal window:

```
touch file1.txt
```

Notice that you can have multiple file names on the same command line.

```
touch file2.txt file3.txt
```

The same is true for `mkdir`, you can make multiple directories in the current directory at the same time.

If you type `ls` or `ls -a` you will see the new files you created.

```
codio@dependbread-judgemadrid:~/workspace/test$ ls
file1.txt  file2.txt  file3.txt
```

Return to your parent directory using `cd ..`

Create a file in a directory without switching to that directory using the following:

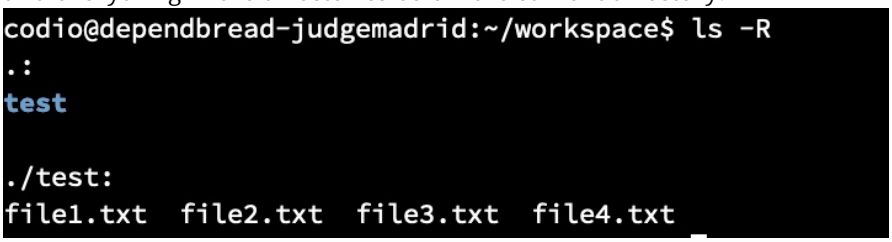
```
touch test/file4.txt
```

We used a **relative path** to create `file4.txt`. Specifically, we told it to create the file in the directory `test` below the *current directory*.

Print the contents of the test folder without switching into it:

```
ls -R
```

The `-R` means recursive so it will list out everything in the current directory and everything in the directories below the current directory:



Deleting files and directories

There are two main commands for deleting files and directories.

Using `rmdir` to remove empty directories

Confirm there are files in the directory you made on the last page using `ls`.

Try to delete the directory:

```
rmdir test
```

You should see the error below:

rmdir test command in terminal followed by an error stating failed to remove test: Directory not empty

Let's make a new directory and try that again.

In the terminal window type:

```
mkdir test2
ls
rmdir test2
ls
```

In the two printed contents from the ls commands, you should be able to see the appearance of test2 and then the disappearance of test2.

Using `rm` to remove directories and files

Similar to above, let's start by trying to delete our directory.

Try to delete the directory:

```
rm test
```

Your result should look like this:

```
codio@dependbread-judgemadrid:~/workspace$ rm test
rm: cannot remove 'test': Is a directory
```

Let's take a closer look at the usage information for the `rm` command.

Open the manual page for `rm`:

```
man rm
```

Read about the `-r` flag and the `-i` flag. Remember, you close the manual by pressing the `q` key.

info

The `rm` command and directories

The `rm` command does not remove directories by default. You need to use the `--recursive` (`-r` or `-R`) option to remove directories, along with all of their contents.

Let's try that again, we'll use the recursive option and also add the `-i` parameter to prompt us. When you do a recursive delete, you are in danger of deleting unexpected items.

Type:

```
rm -R -i test
```

The results should look as follows, a somewhat tedious, but safe, deletion of a directory and all its contents:

```
codio@dependbread-judgemadrid:~/workspace$ rm test -R -i
rm: descend into directory 'test'? y
rm: remove regular empty file 'test/file1.txt'? y
rm: remove regular empty file 'test/file2.txt'? y
rm: remove regular empty file 'test/file3.txt'? y
rm: remove regular empty file 'test/file4.txt'? y
rm: remove directory 'test'? y
```

Wildcards

We'll take a brief pause from our discussion about Bash file commands to talk about wildcards since they are useful in conjunction with file commands.

We'll cover the main three wildcards (see table below), but if you would like to learn more, you can take a look in the GNU/Linux Command-Line Tools Summary.

Wildcard	Description
`*`	represents any number of characters
`?`	represents any single character
`[ ]`	represents a range, can be `[1-3]` or `[1,2,3]`

Let's start out by creating a few directories in our workspace:

```
mkdir test
mkdir test1
mkdir test2
mkdir test3
mkdir testrandom
```

Using wildcards with the `ls` command

Try:

```
ls test?
```

You should see something like this:

```
codio@dependbread-judgemadrid:~/workspace$ ls test?
test1:

test2:

test3:
```

▼ **Why aren't the directories `test` or `testrandom` listed?**

You only get the files that start with `test` and are followed by a single character.

Try these variations:

- List everything except the `test` directory.

  ▼

  Solution

  Use the `?` wild card to ensure there is at least one character after test.

  ```
  ls test?*
  ```

- List the `test1` and `test2` directories.

  ▼

  Solution

  Use brackets to provide multiple values. Either of the solutions below will result in the same output.

  ```
  ls test[1,2]
  ls test[1-2]
  ```

- List the `test1` and `test3` directories:

  ▼

  Solution

  Since it's not a range, this one needs the two values you are looking for separated by a comma.

  ```
  ls test[1,3]
  ```

Use the following commands to create files that might be typical in a development environment:

```
touch main.c
touch utils.c
touch main.obj
touch utils.obj
touch main.exe
touch readme.txt
```

If you were viewing a directory for the first time, you might first check to see if there are any readme or text files.

Try looking for a readme or text file by:

```
ls readme*
ls *.txt
```

You might want to know what C files are in there:

```
ls *.c
```

Moving, copying and viewing

In addition to creating and deleting files and directories, you might need to move, rename, or copy them.

Let's start out by creating a directory for code files:

```
mkdir code
```

Using `mv` to move or rename files

Take a look at the manual pages for the `mv` command:

```
man mv
```

What needs to be specified for the `mv` command? Remember to close the manual with `q` when you are done.

Let's move all our `.c` files into the code directory:

```
mv *.c code
```

We'll use the `tree` command to see if everything is where we want it.

```
tree
```

As a last step, we'll clean up some of the clutter in this workspace directory:

```
rm *.obj
rm *.exe
tree
```

You should see something like the image below:

You also use the `mv` command to rename files.

Rename the readme file:

```
mv readme.txt README.txt
```

If the destination you supply is not a directory, the `mv` command will perform a rename.

Using `cp` to copy files and directories

The `cp` command is very powerful, especially if you use it with the `-R` parameter so that it is recursive.

Take a look at the manual pages for the `cp` command:

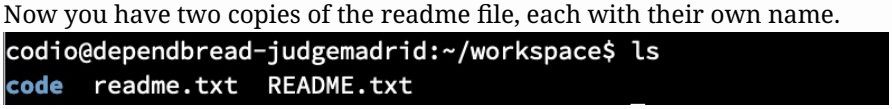
```
man cp
```

Let's see how the `cp` command differs from the `mv` command. Earlier we renamed the file `readme.txt` to `README.txt`.

We'll start with an `ls`to see what's in the directory right now and end with an `ls` to see what has changed.

```
ls
cp README.txt readme.txt
ls
```

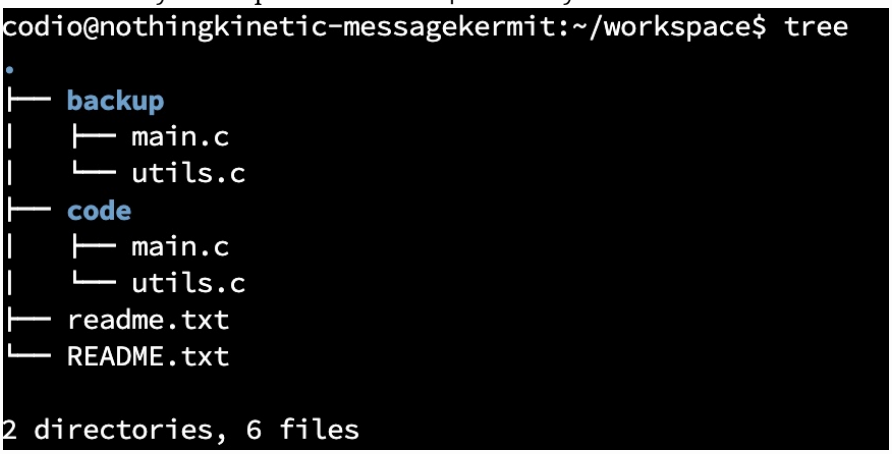
Now you have two copies of the readme file, each with their own name.



Try this:

```
cp -R code backup
tree
```

A lot happened! The `backup` directory was created, and the contents of the `code` directory was copied to the `backup` directory.



Summary

Commands we covered in this assignment:

Command	Description
`cp` *source destination*	Makes a copy of a file or directory
`mkdir` *name*	Creates a directory
`mv` *source destination*	Moves a file or directory
`rm`	Removes files and directories
`rmdir` *name*	Removes empty directories
`touch` *name*	Creates a new empty file named *name*
`tree`	Displays the contents of a directory in a tree structure