Introduction to tools

Welcome to Tech Elevator. This is the first of many exercises you'll complete throughout the cohort. The purpose of the exercises is to reinforce the concepts you learn each day and to provide you with the opportunity to practice your new skills.

This exercise covers Git and several Unix commands typically used alongside Git. This exercise helps you gain familiarity and confidence in working with the command line.

There are two parts to this exercise. The first part asks you to navigate and modify the file system through the command line.

In part two, you'll need to provide the correct Unix command for eleven questions. You'll complete this part of the exercise in a program called **Visual Studio Code**, a popular code editor.

Learning objectives

After completing this exercise, you'll understand:

- How to navigate a file system directory structure.
- How to identify your current working directory.
- How to display files within a directory.
- How to rename, copy, and move files.
- Common techniques used when working with Git.
- How to push work using Git for review.
- How to use the README file to complete exercises.

Evaluation criteria and functional requirements

- Directories and files that have been modified, added, removed, moved, or renamed reflect the work that was completed during the exercise.
- You answered all questions not marked as optional. You must do this for all exercises you work on throughout the cohort.
- The `verify-part-1.sh` script prints `22/22 tests pass` to the screen when you run `sh verify-part-1.sh` from the command line.
- The `verify-part-2.sh` script prints `10/10 tests pass` to the screen when you run `sh verify-part-2.sh` from the command line.
- You added the appropriate commits to Git.

Tips and tricks

- When working through exercises at Tech Elevator, refer to the `README.md` files found at the root of each exercise folder for clarification on what's expected for each exercise, the work that you must complete, and information related to the concepts you'll learn in each exercise.
- There's also a section that includes helpful tips, tricks, and additional links related to some of the concepts discussed in each exercise. *Be sure to use the README file as you work through each exercise.*

- Reference Finder (Mac) or Explorer (PC) for a visual view of directory structure created by your commands.
    - While you can create folders and files using the Finder or Explorer, it's recommended that you do these exercises in the command line to reinforce what you learned today.
- If the script reports a folder or file not existing after you created it, make sure the folder or filename doesn't have a typo.

Instructions

Part one

Step One: Prepare your workspace

1. Open a Terminal (Mac) or Git Bash (PC) window and navigate to your home directory by using the command `cd ~`.

2. Open a second Terminal (Mac) or Git Bash (PC) window and navigate to the folder containing today's exercise.

3. In the second window, run the `verify-part-1.sh` script by using the command `sh verify-part-1.sh`. This command runs a script that verifies that you completed each step successfully. Initially, you'll see the following output:

```
---- VERIFYING  ----

❗ 1. ~/playground does not exist
❗ 2. ~/playground/usa does not exist
❗ 3. ~/playground/canada does not exist
❗ 4. ~/playground/usa/ohio does not exist
❗ 5. ~/playground/usa/pennsylvania does not exist
❗ 6. ~/playground/usa/michigan does not exist
❗ 7. ~/playground/canada/quebec does not exist
❗ 8. ~/playground/canada/british-columbia does not exist
❗ 9. ~/playground/usa/ohio/cuyahoga does not exist
❗ 10. ~/playground/usa/ohio/hamilton does not exist
❗ 11. ~/playground/usa/ohio/franklin does not exist
❗ 12. ~/playground/usa/pennsylvania/allegheny does not exist
❗ 13. ~/playground/usa/michigan/wayne does not exist
❗ 14. ~/playground/usa/ohio/cuyahoga/cleveland.txt does not exist
❗ 15. ~/playground/usa/ohio/hamilton/cincinnati.txt does not exist
❗ 16. ~/playground/usa/ohio/franklin/columbus.txt does not exist
❗ 17. ~/playground/usa/pennsylvania/allegheny/pittsburgh.txt does not exist
❗ 18. ~/playground/usa/michigan/wayne/detroit.txt does not exist
❗ 19. ~/playground/canada/quebec/montreal.txt does not exist
❗ 20. ~/playground/canada/quebec/quebec-city.txt does not exist
❗ 21. ~/playground/canada/british-columbia/vancouver.txt does not exist
❗ 22. ~/playground/canada/british-columbia/prince-george.txt does not exist

0/22 tests pass
```

Step Two: Apply the Unix commands you learned

Following the below steps, type the appropriate Unix commands in the first window to build a directory structure. If you want to check your progress, re-run `verify-part-1.sh` in the second window.

1. Starting from your home directory, create a new directory called `playground`.
2. Create a new directory at the path `~/playground/usa`.
3. Create a new directory at the path `~/playground/canada`.
4. Create a new directory at the path `~/playground/usa/ohio`.
5. Create a new directory at the path `~/playground/usa/pennsylvania`.
6. Create a new directory at the path `~/playground/usa/michigan`.
7. Create a new directory at the path `~/playground/canada/quebec`.
8. Create a new directory at the path `~/playground/canada/british-columbia`.
9. Create a new directory at the path `~/playground/usa/ohio/cuyahoga`.
10. Create a new directory at the path `~/playground/usa/ohio/hamilton`.
11. Create a new directory at the path `~/playground/usa/ohio/franklin`.
12. Create a new directory at the path `~/playground/usa/pennsylvania/allegheny`.
13. Create a new directory at the path `~/playground/usa/michigan/wayne`.
14. Create a new file at the path `~/playground/usa/ohio/cuyahoga/cleveland.txt`.
15. Create a new file at the path `~/playground/usa/ohio/cuyahoga/cincinnati.txt` and move it to the directory at `~/playground/usa/ohio/hamilton`.
16. Copy the file from `~/playground/usa/ohio/cuyahoga/cleveland.txt` and place it into the directory at `~/playground/usa/ohio/franklin`. Change the name of the file to `columbus.txt`.
17. Create a new file at the path `~/playground/usa/pennsylvania/allegheny/pittsburgh.txt`.
18. Create a new file at the path `~/playground/usa/michigan/wayne/detroit.txt`.
19. Create a new file at the path `~/playground/canada/quebec/montreal.txt`.
20. Create a new file at the path `~/playground/canada/quebec/quebec-city.txt`.
21. Create a new file at the path `~/playground/canada/british-columbia/vancouver.txt`.
22. Create a new file at the path `~/playground/canada/british-columbia/prince-george.txt`.

When you complete the exercise, you'll receive the following feedback from `verify-part-1.sh`:

```
---- VERIFYING  ----

✅ 1. ~/playground exists
✅ 2. ~/playground/usa exists
✅ 3. ~/playground/canada exists
✅ 4. ~/playground/usa/ohio exists
✅ 5. ~/playground/usa/pennsylvania exists
✅ 6. ~/playground/usa/michigan exists
✅ 7. ~/playground/canada/quebec exists
✅ 8. ~/playground/canada/british-columbia exists
✅ 9. ~/playground/usa/ohio/cuyahoga exists
✅ 10. ~/playground/usa/ohio/hamilton exists
✅ 11. ~/playground/usa/ohio/franklin exists
✅ 12. ~/playground/usa/pennsylvania/allegheny exists
✅ 13. ~/playground/usa/michigan/wayne exists
✅ 14. ~/playground/usa/ohio/cuyahoga/cleveland.txt exists
✅ 15. ~/playground/usa/ohio/hamilton/cincinnati.txt exists
```

```
✅ 16. ~/playground/usa/ohio/franklin/columbus.txt exists
✅ 17. ~/playground/usa/pennsylvania/allegheny/pittsburgh.txt exists
✅ 18. ~/playground/usa/michigan/wayne/detroit.txt exists
✅ 19. ~/playground/canada/quebec/montreal.txt exists
✅ 20. ~/playground/canada/quebec/quebec-city.txt exists
✅ 21. ~/playground/canada/british-columbia/vancouver.txt exists
✅ 22. ~/playground/canada/british-columbia/prince-george.txt exists

22/22 tests pass

Congratulations! All tests are passing.
Continue on to Part 2 in the README.
```

Your folder and file structure looks like this in Finder (Mac) or Explorer (Windows) once you complete part one:

```
├──canada
│   ├──british-columbia
│   │       prince-george.txt
│   │       vancouver.txt
│   │
│   └──quebec
│           montreal.txt
│           quebec-city.txt
│
└──usa
    ├──michigan
    │   └──wayne
    │           detroit.txt
    │
    ├──ohio
    │   ├──cuyahoga
    │   │       cleveland.txt
    │   │
    │   ├──franklin
    │   │       columbus.txt
    │   │
    │   └──hamilton
    │           cincinnati.txt
    │
    └──pennsylvania
        └──allegheny
                pittsburgh.txt
```

Part two

Step One: Prepare your workspace

You'll use the second terminal window for this part. Make sure your current working directory is `exercise-student`. If it isn't, change it before continuing with this step.

Next, type `code commands/` in the terminal window. This command opens all of the files in the `commands` directory in Visual Studio Code.

Step Two: Complete the command files

There are eleven files in the `command` directory. You can open any file by clicking on it in the file list on the left side of the window. Each file is numbered in suggested order of completion, but you can do them in any order you wish.

Each file has the task in the form of a question, such as:

```
# How do you create a new directory called "notes" in your current location?


```

Below that line, you'll type the command that performs that task.

You can verify your work as you progress by running the `verify-part-2.sh` script by using the command `sh verify-part-2.sh`. Initially, you'll see the following output:

```
---- VERIFYING  ----

❗ 1. Create a new directory called "notes" in your current directory
❗ 2. Create a new directory called "work" in your home directory
❗ 3. Change your current working directory to a subfolder named "projects"
❗ 4. Change your current working directory to the parent directory
❗ 5. Change your current working directory to a directory at the root of the file
system named "usr"
❗ 6. Remove a directory called "temp" in your current directory
❗ 7. List the contents of the current directory
❗ 8. List the contents of a subfolder named "notes"
❗ 9. Create a copy of "foo.txt" named "baz.txt"
❗ 10. Move a file named "lorem.txt" into a subfolder named "ipsum"
❗ 11. Rename the file "fizz.txt" to "buzz.txt"

0/11 tests pass
```

When you've successfully completed all tasks, you'll see all of the tests passing:

```
---- VERIFYING  ----

✅ 1. Create a new directory called "notes" in your current directory
✅ 2. Create a new directory called "work" in your home directory
✅ 3. Change your current working directory to a subfolder named "projects"
✅ 4. Change your current working directory to the parent directory
✅ 5. Change your current working directory to a directory at the root of the
file system named "usr"
✅ 6. Remove a directory called "temp" in your current directory
```

```
✅ 7. List the contents of the current directory
✅ 8. List the contents of a subfolder named "notes"
✅ 9. Create a copy of "foo.txt" named "baz.txt"
✅ 10. Move a file named "lorem.txt" into a subfolder named "ipsum"
✅ 11. Rename the file "fizz.txt" to "buzz.txt"

11/11 tests pass

Congratulations! All tests are passing.
Continue on to the next step in the README to submit your exercise.
```

IMPORTANT: Make sure to save your changes before running `sh verify-part-2.sh`. In Visual Studio Code, you can go to the `File` menu, then `Save`, or use the keyboard combination `⌘+S` on Mac or `Ctrl+S` on Windows.

When you get a passing result for one of the tasks, you can commit your changes in Git. Committing your changes often is a good habit to get into, especially when you complete a particular task or component.

For example, to commit `01_create_directory.txt`, type the following commands:

```
$ git add commands/01_create_directory.txt
$ git commit -m "Submitting create directory task"
$ git push origin main
```

> Note: The earlier example uses the `01_create_directory` task. Make sure to replace the filename and comment based on the file or files that you're committing.

Step Three: Submit your exercise using Git commands

After you verified your progress by running `sh verify-part-1.sh` and `sh verify-part-2.sh`, you'll need to submit all your work.

You can type the command `git status` to see what files have been changed and still need to be committed.

You can add each individual file like the previous example, or type `git add -A` to add all changed files.

After you've run `git add` to add the files, commit and push your changes by typing the following Git commands to submit your work:

```
$ git commit -m "Submitting Week 1 Day 1 exercise"
$ git push origin main
```