Welcome to Day 1!!!

Schedule

- 9:00am - 4:30pm
  - You should have Slack Presence during this time.
  - Treat your time at Tech Elevator like a job
- Lecture usually runs from 9:00am to around Noon.
  - Please arrive a few minutes early so we can start on time.
  - You are considered late if you arrive after Daily Pulse Survey (more on this soon).
  - If you are going to need to be late, miss class, or leave early, please let me know.
    - Slack is best way to do that in a timely fashion.
- My availability after-hours:
  - Please only reach out after hours or on weekends if you have exhausted all other avenues, including asking your classmates for help. In general, if I am able, I will do my best to respond to Slack messages. Obviously that isn't always possible, but I'll do my best to respond if I can. If something requires deeper digging in or you want to review concepts, I will address the issue on the next class day, unless it is something I think constitutes an emergency. Putting your homework off to the last minute may seem like an emergency to you, but it's unlikely to seem that way to me. 😉
- Please be respectful of everyone's time!

"Typical" Day

- Morning
  - Daily Pulse Survey
  - Lecture/New Material

- Breaks
  - Around 10:15
  - Around 11:15
  - Whenever possible, we will stick to that schedule closely

- Afternoon
  - Work on exercises
  - Pathway Events
  - Meet 1:1 with instructors/fellow
  - Work on practice Projects

Resources

- Daily Pulse Survey
  - Socrative Room: WLM1DAILYPULSE
  - Your ID is your email address in ALL CAPS

- Classroom Resources Site
  - https://sites.google.com/techelevator.com/wlm1java/home
  - Daily Lecture Recordings
  - Daily Lecture Slides
  - Other useful links/reference material/resources

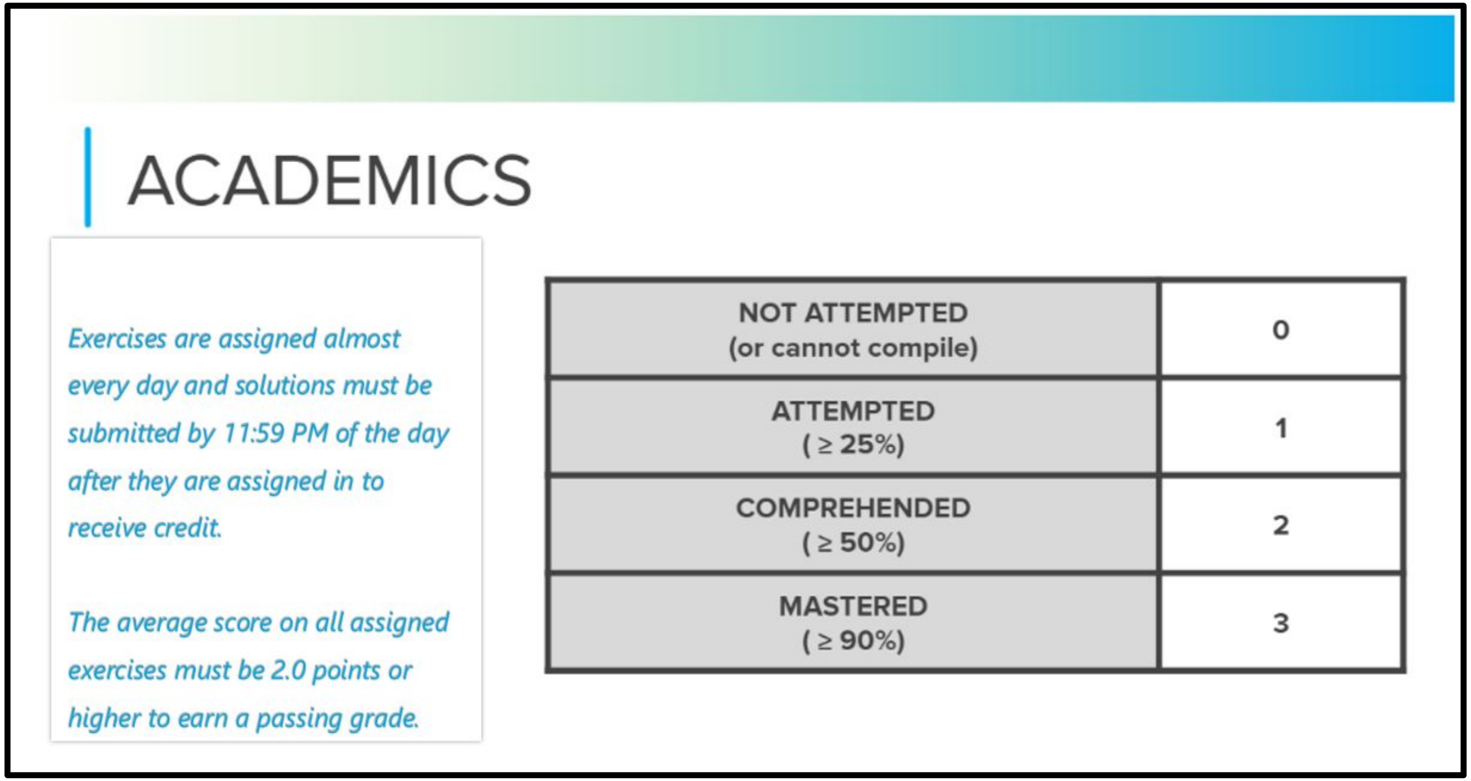
Quizzes

- Most reading topics have a follow-up quiz in BootcampOS
- Scores are used to help you and your instructor gauge how well you understand the reading material and what might need some extra focus in class.
- You will receive a score of 0 if you miss a quiz. Missing several quizzes in a row may earn you a talk with our Campus Director
- Please note that Quizzes for the current day should be complete by 7:00am.
- Daily Pulse Survey will open by 8:45am.

Exercise Due Date & Late Policy

- All exercises are due at 11:59:59pm (23:59:59) the next class day after assigned.
- Late exercises ***<u>WILL NOT</u>*** be accepted without prior instructor approval.
- Instructor approval must be obtained prior to the close of business on the day the assignment is due.
    - Needing an extension after-hours is unlikely to seem like an emergency to me, so make sure to ***<u>let me know DURING business hours</u>*** if you think you may wind up needing some extra time so we can discuss it.

Academic Policy



ACADEMICS

Exercises are assigned almost every day and solutions must be submitted by 11:59 PM of the day after they are assigned in to receive credit.

The average score on all assigned exercises must be 2.0 points or higher to earn a passing grade.

NOT ATTEMPTED (or cannot compile)	0
ATTEMPTED (≥ 25%)	1
COMPREHENDED (≥ 50%)	2
MASTERED (≥ 90%)	3

BECOMING PROFICIENT

- We focus on becoming proficient at key concepts
- Feedback is provided so you and your instructor know where you need to improve
- We expect your average to remain at or above 2.0
- Any work submitted must be your own. We may ask you to explain your code to us and you will, at times, ***be asked to write code during class*** to evaluate your understanding. Working with others is helpful, ***to a point.*** If you are working with others but are unable to write the same code on your own, you are unlikely to be able to do so on the job. Your on the job work will not be done by committee, so ***you are not doing yourself any favors if you get your assignments in but can't actually do the work on your own***.
- Please ask an instructor, fellow, or classmate if you need help! We move fast so ***if you have been banging on something for a while, it's time to seek help***.
- Make use of the Study Guide as you learn new material

Remember...

- This is your JOB for the next 14 weeks, so treat it that way
- The goal is for you to **UNDERSTAND** what you are doing, not just complete the required work
- We encourage working together to solve exercises. Collaboration will help you learn and is an important skill to have when you head into your career. Helping others will help solidify **your understanding**
- If you just focus on completing exercises, rather than **understanding** them, you are going to have a lot less of a chance of succeeding in that new career you are here to start
- Topics build on each other. Lack of understanding of previous topics makes moving forward even more difficult so make sure to reach out right away if you are struggling with anything - we move FAST!
- You are not in a race or a competition with your classmates. Everyone learns differently so work WITH your classmates...
- Your goal is not to finish fastest, but to make sure you are prepared for your new career when you graduate.

Ready?

Let's Go!!!!

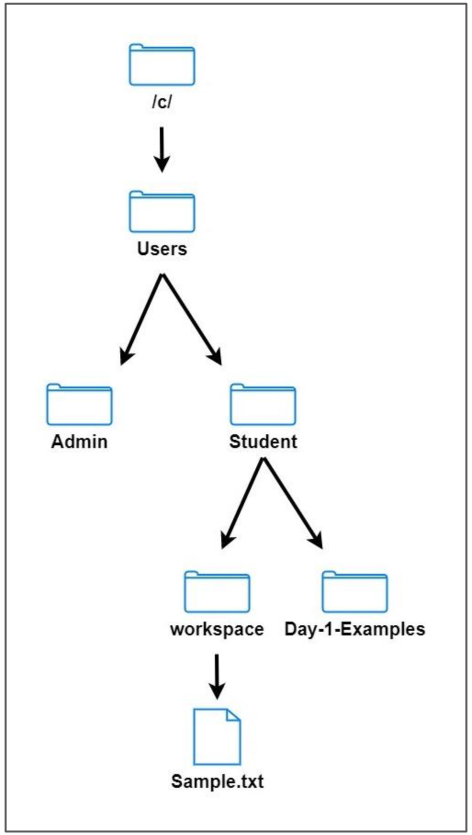
Command Line Shell & Version Control

Today's Objectives

- Navigating files using the UI (Windows Explorer).
- Finding and opening a command line application (Windows Terminal).
- Pulling your upstream repository (Lecture Code, Exercises, and Solutions).
- Opening and using the Visual Studio Code text editor.
- Using the command line.
- Pathing and Hierarchical Structures (Parent/Child folder and file structures).
- Basic Bash commands: **`cd, ls, pwd`**
- What is version control.
- Working with Git and the workflow used in class.

What Is a File System?

- Files are the parts of the file system that contain the data we want.
- Folders hold other folders and files. ALL files exist in some folder of the File System.
- Folders and files BOTH have metadata used to describe them. Metadata includes information such as modified date, names, and permissions that are attached to the files and folders as part of the File System.



What is a Command Line Shell?

- A shell is the means by which the user interacts with the computer.
    - Shells can be in the form of a graphical user interface (i.e. Windows, MacOS)
    - Shells can also be in the form of a command line interface (CLI), or Command Line shells, in which users type in commands followed by parameters.
- Information Technology professionals should be familiar with command line shells.
- In this class we will be using the Windows Terminal Bash Command Line Shell, which allows for UNIX commands from a Windows workstation.

Command Line Commands: Moving Around

- Data in your workstation are organized into files and folders.
- The main command to move around folder is `cd`. There are several variations of these:
  - cd ~ : Returns you to your home directory.
  - cd <directory name> : Takes you to a specified directory i.e. cd workspace takes you to a folder called workspace
  - cd .. : Takes you one level up.
- You can always see what directory you're in by typing pwd.
- The ls command lists all the files in the current directory.

Moving Around: Absolute Path

When you use the **pwd** command in your home directory, you should see something like this:

```
Student@Dell-V4-Multi MINGW64 ~
$ pwd
/c/Users/Student
```

pwd outputs the current directory but the path displayed is the full path of the directory from the "root" of the drive.

Moving Around: Absolute Path

When you use the **pwd** command in your home directory, you should see something like this:

```
Student@Dell-V4-Multi MINGW64 ~
$ pwd
/c/Users/Student
```

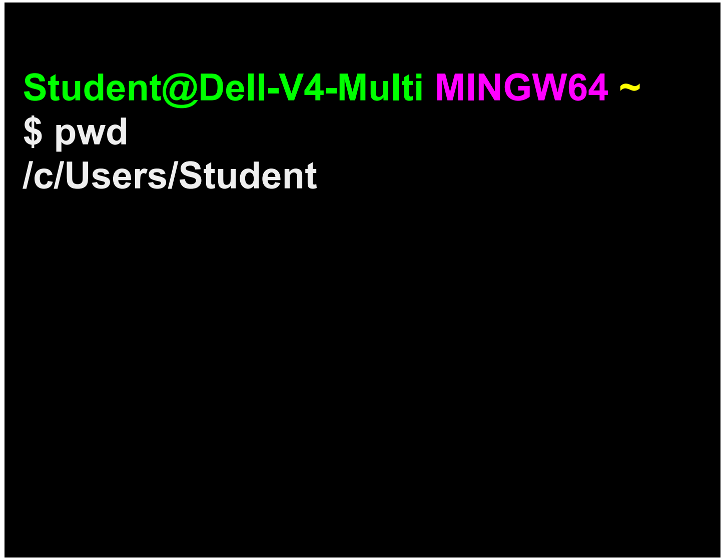
pwd outputs the current directory but the path displayed is the full path of the directory from the "root" of the drive.

The **/** at the beginning of the path indicates that the path you are looking at begins at the root rather than at the current directory. This is known as the **absolute path**.

Moving Around: Relative Path

When you are in a directory, you can **cd** into subdirectories of the directory you are in.

If you use the **pwd** command in the `Student` directory, you will see that the absolute path represents the same location you specified using a relative path.



```
Student@Dell-V4-Multi MINGW64 ~
$ pwd
/c/Users/Student
```

Moving Around: Relative Path

When you are in a directory, you can **cd** into subdirectories of the directory you are in.

If you use the **pwd** command in the `Student` directory, you will see that the absolute path represents the same location you specified using a relative path.

A path that does not start with / starts at the current directory. Since the path is specified in relation to the current directory, this is known as the **relative path**.

```
Student@Dell-V4-Multi MINGW64 ~
$ pwd
/c/Users/Student
```

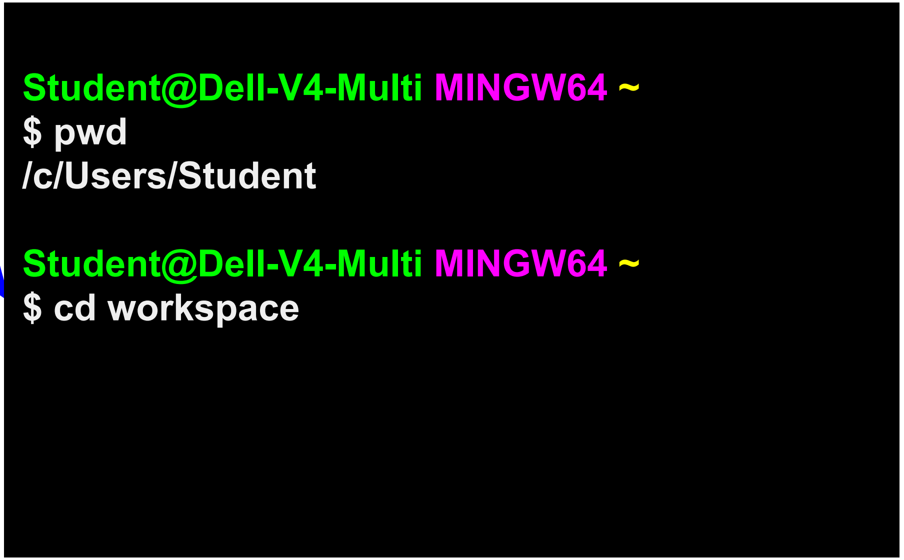
Moving Around: Relative Path

When you are in a directory, you can **cd** into subdirectories of the directory you are in.

If you use the **pwd** command in the `Student` directory, you will see that the absolute path represents the same location you specified using a relative path.

A path that does not start with **/** starts at the current directory. Since the path is specified in relation to the current directory, this is known as the **relative path**.

The path in **cd workspace** does not start with a **/** so it changes to the `workspace` directory contained in the current directory

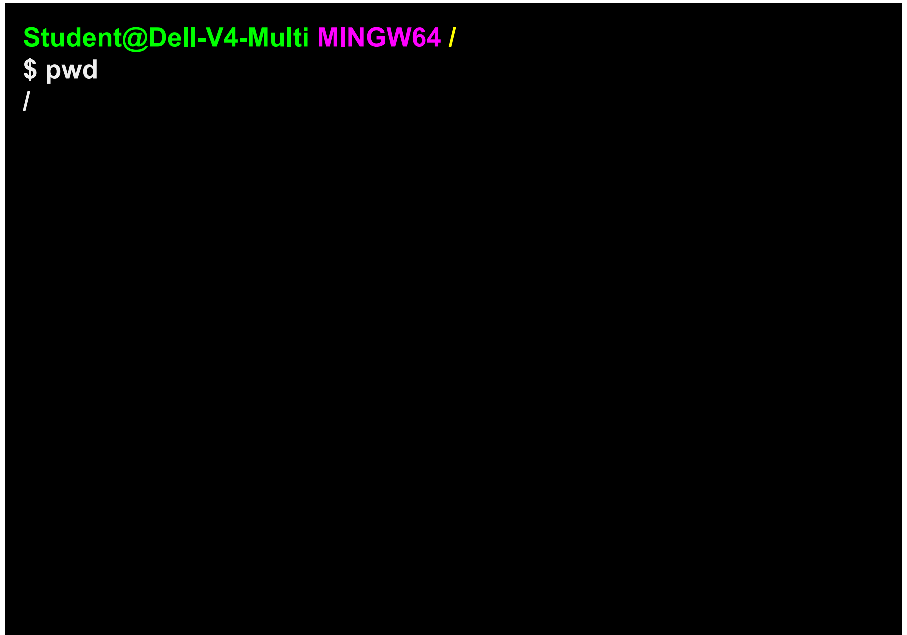


```
Student@Dell-V4-Multi MINGW64 ~
$ pwd
/c/Users/Student

Student@Dell-V4-Multi MINGW64 ~
$ cd workspace
```

Moving Around: The Tilde(~)

The tilde (~) is a special symbol used to denote the home directory. For all of your workstations this has been set to: /c/Users/Student

A path that starts with ~ starts at the user's home directory.

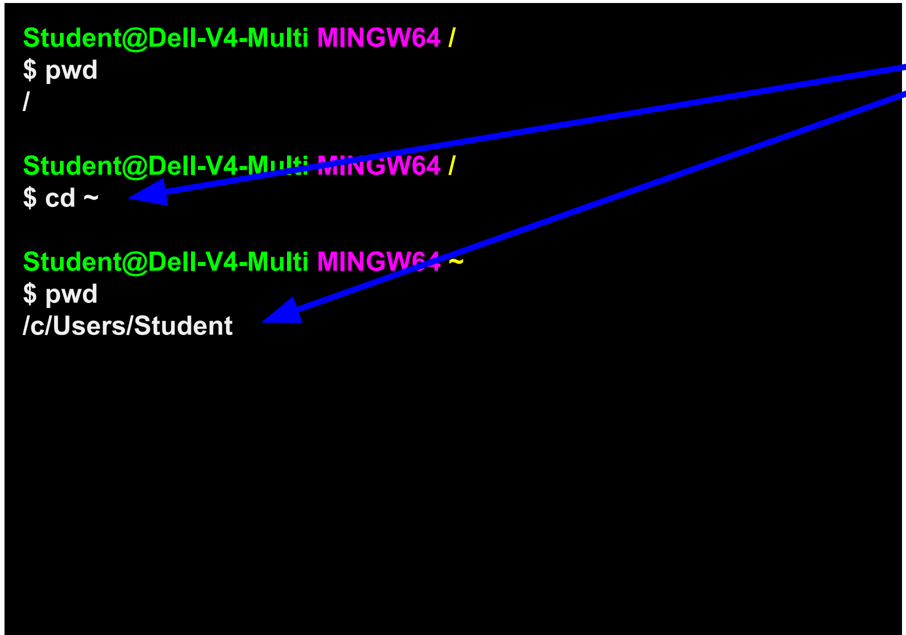


```
Student@Dell-V4-Multi MINGW64 /
$ pwd
/
```

Moving Around: The Tilde(~)

The tilde (~) is a special symbol used to denote the home directory. For all of your workstations this has been set to: /c/Users/Student

A path that starts with ~ starts at the user's home directory.



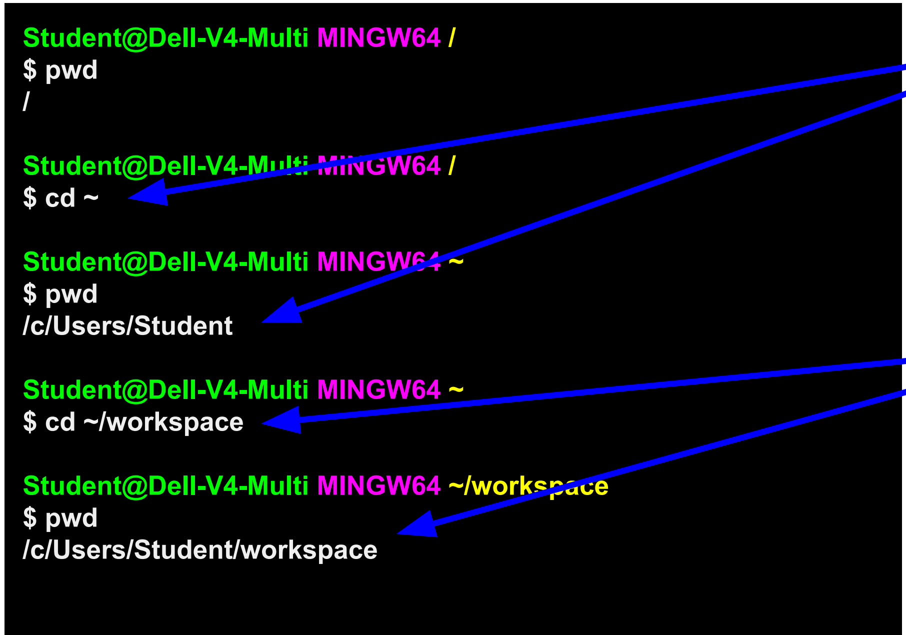
```
Student@Dell-V4-Multi MINGW64 /
$ pwd
/

Student@Dell-V4-Multi MINGW64 /
$ cd ~

Student@Dell-V4-Multi MINGW64 ~
$ pwd
/c/Users/Student
```

cd ~ will always return you to your home directory.

Moving Around: The Tilde(~)

The tilde (~) is a special symbol used to denote the home directory. For all of your workstations this has been set to: /c/Users/Student

A path that starts with ~ starts at the user's home directory.



```
Student@Dell-V4-Multi MINGW64 /
$ pwd
/

Student@Dell-V4-Multi MINGW64 /
$ cd ~

Student@Dell-V4-Multi MINGW64 ~
$ pwd
/c/Users/Student

Student@Dell-V4-Multi MINGW64 ~
$ cd ~/workspace

Student@Dell-V4-Multi MINGW64 ~/workspace
$ pwd
/c/Users/Student/workspace
```

cd ~ will always return you to your home directory.

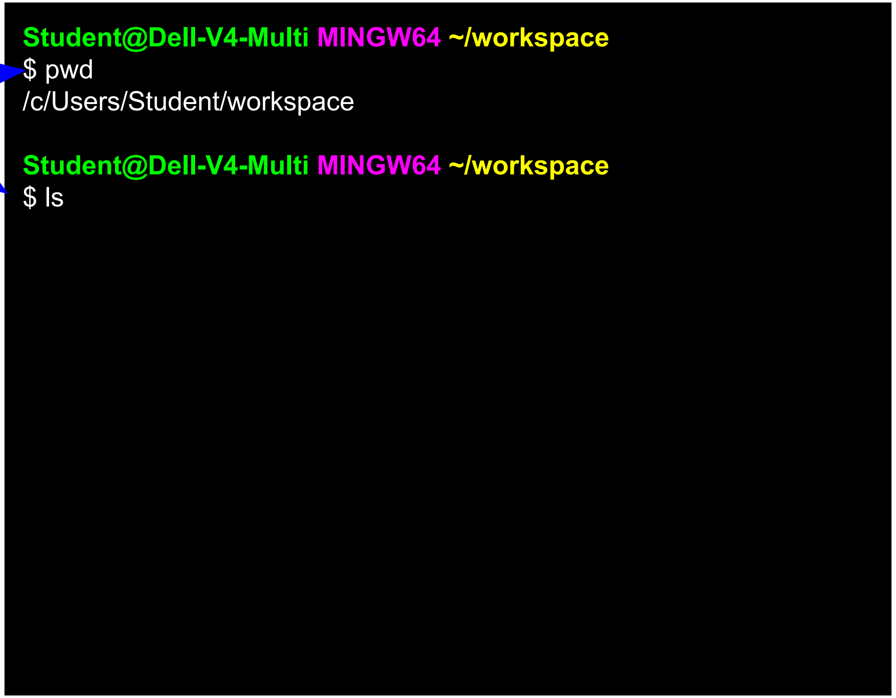
Note that we were able to use cd and specify a path relative to the home directory using ~ and that it put us at the same location we would have arrived at using the absolute path to the user's home directory.

Making Directories & Copying/Moving Files

Command Line Commands: Making Directories

To create a directory we use the **mkdir <directory name>** command.

If we use the **ls** command in the current directory, there are no subdirectories.



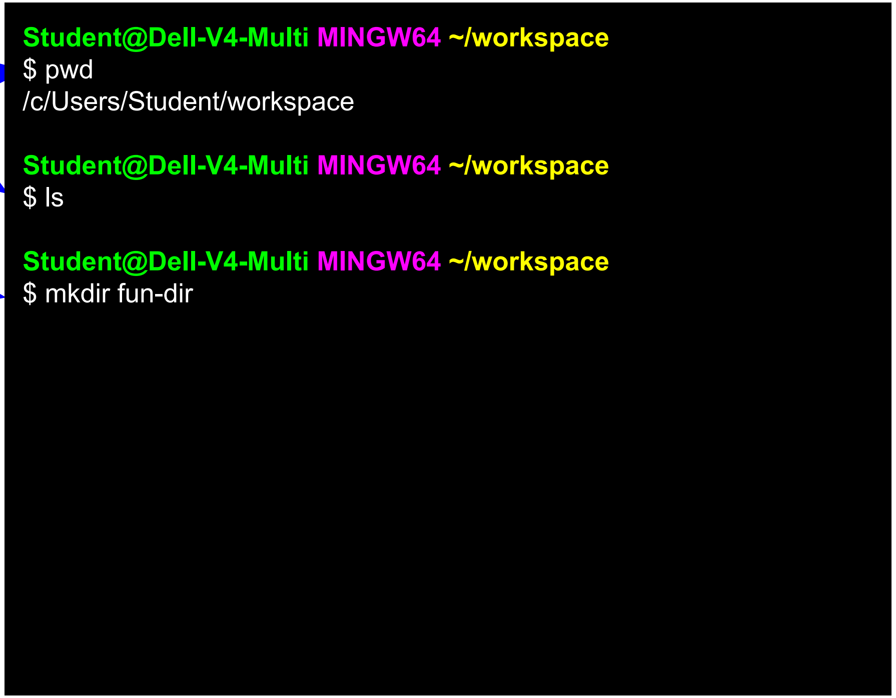
```
Student@Dell-V4-Multi MINGW64 ~/workspace
$ pwd
/c/Users/Student/workspace

Student@Dell-V4-Multi MINGW64 ~/workspace
$ ls
```

Command Line Commands: Making Directories

To create a directory we use the **mkdir <directory name>** command.

If we use the **ls** command in the current directory, there are no subdirectories.

We execute the **mkdir fun-dir** command to make the **`fun-dir`** directory.



```
Student@Dell-V4-Multi MINGW64 ~/workspace
$ pwd
/c/Users/Student/workspace

Student@Dell-V4-Multi MINGW64 ~/workspace
$ ls

Student@Dell-V4-Multi MINGW64 ~/workspace
$ mkdir fun-dir
```

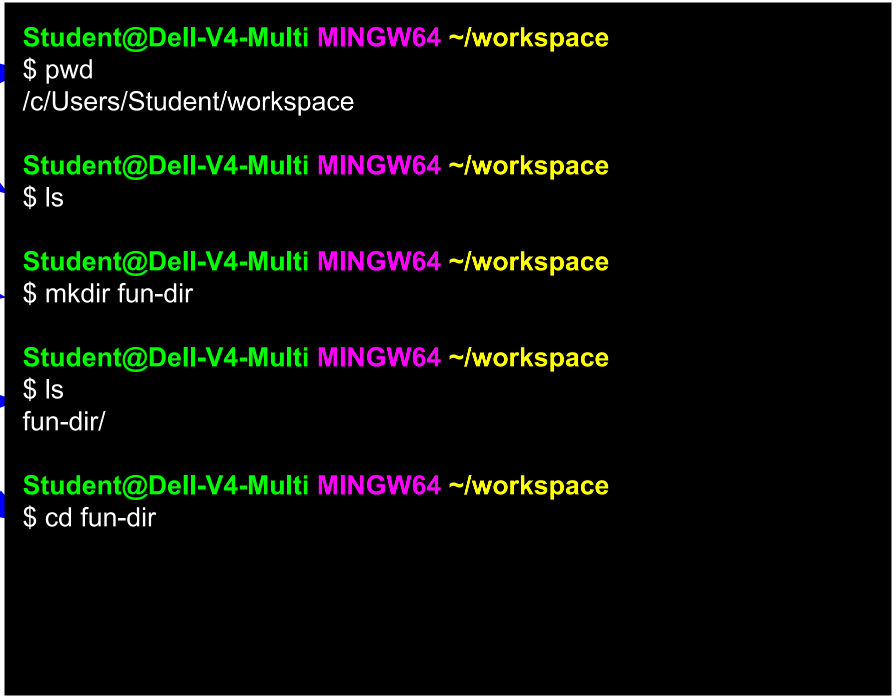
Command Line Commands: Making Directories

To create a directory we use the **mkdir <directory name>** command.

If we use the **ls** command in the current directory, there are no subdirectories.

We execute the **mkdir fun-dir** command to make the `fun-dir` directory.

Now **ls** lists the `fun-dir` directory and we can **cd** into it



```
Student@Dell-V4-Multi MINGW64 ~/workspace
$ pwd
/c/Users/Student/workspace

Student@Dell-V4-Multi MINGW64 ~/workspace
$ ls

Student@Dell-V4-Multi MINGW64 ~/workspace
$ mkdir fun-dir

Student@Dell-V4-Multi MINGW64 ~/workspace
$ ls
fun-dir/

Student@Dell-V4-Multi MINGW64 ~/workspace
$ cd fun-dir
```

Command Line Commands: Making Directories

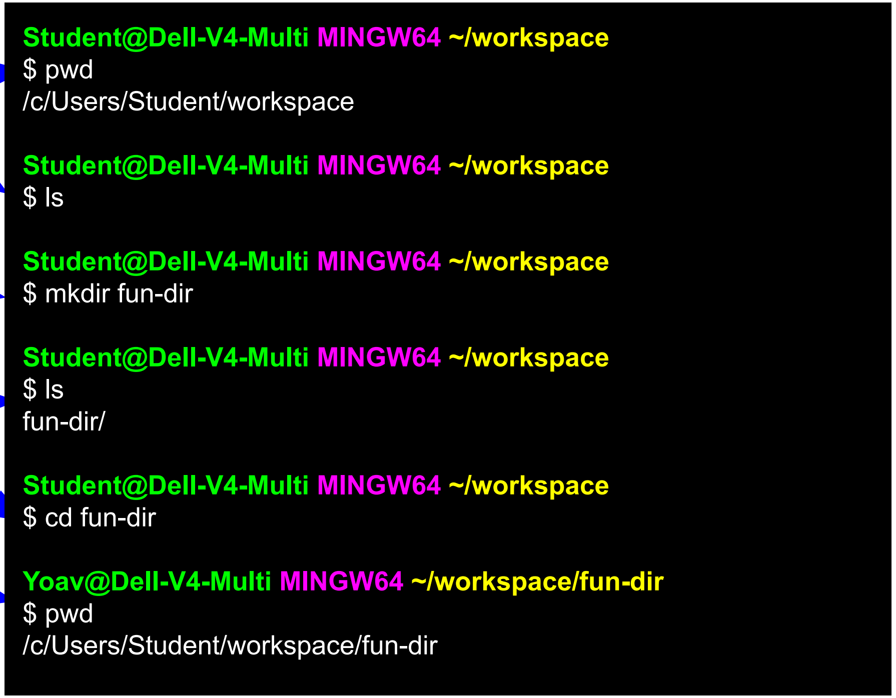
To create a directory we use the **mkdir <directory name>** command.

If we use the **ls** command in the current directory, there are no subdirectories.

We execute the **mkdir fun-dir** command to make the `fun-dir` directory.

Now **ls** lists the `fun-dir` directory and we can **cd** into it.

Executing **pwd** will show the `fun-dir` directory has been created as a subdirectory of the original directory..

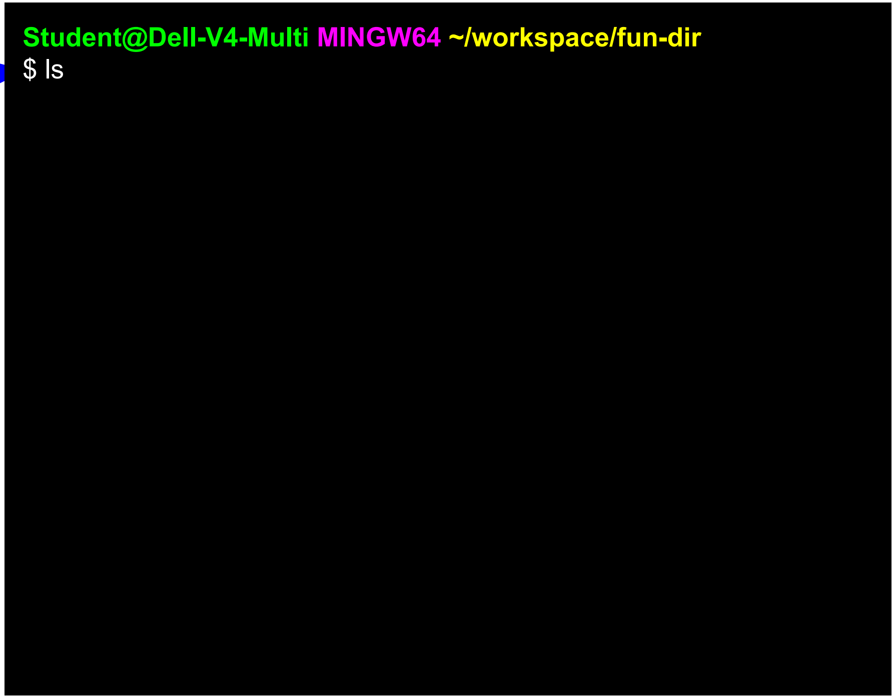


```
Student@Dell-V4-Multi MINGW64 ~/workspace
$ pwd
/c/Users/Student/workspace

Student@Dell-V4-Multi MINGW64 ~/workspace
$ ls

Student@Dell-V4-Multi MINGW64 ~/workspace
$ mkdir fun-dir

Student@Dell-V4-Multi MINGW64 ~/workspace
$ ls
fun-dir/

Student@Dell-V4-Multi MINGW64 ~/workspace
$ cd fun-dir

Yoav@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ pwd
/c/Users/Student/workspace/fun-dir
```

Command Line Commands: View File Contents

Let's create a file and view its contents...

If we use the **ls** command in the current directory, we see there are no files in it.



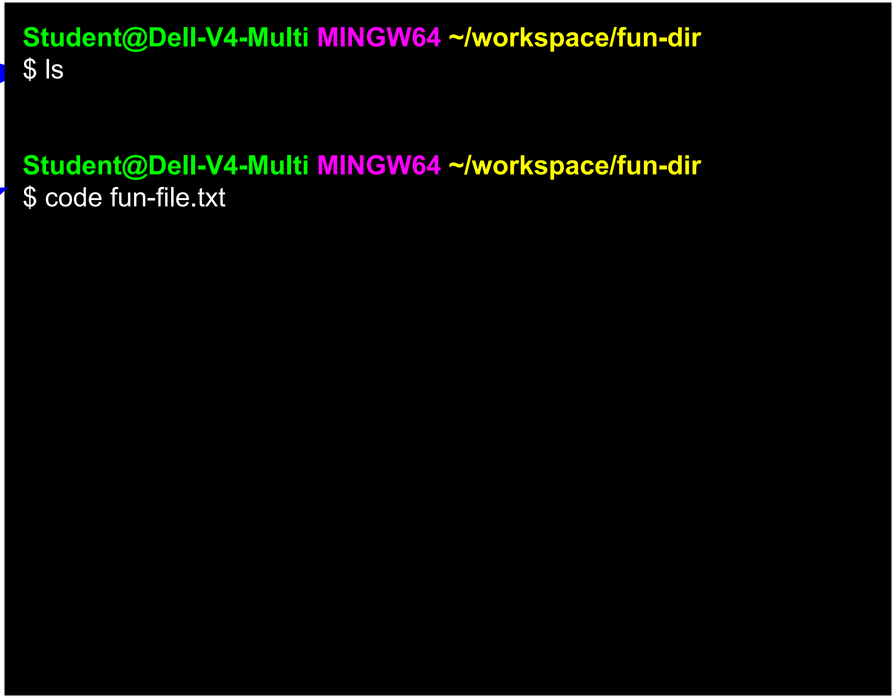
```
Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ ls
```

Command Line Commands: View File Contents

Let's create a file and view its contents...

If we use the **ls** command in the current directory, we see there are no files in it.

We can create a file using VIsual Studio Code by using the **code** command followed by the name of the file we want to create. This will create a new file and open it with VS Code



```
Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ ls

Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ code fun-file.txt
```

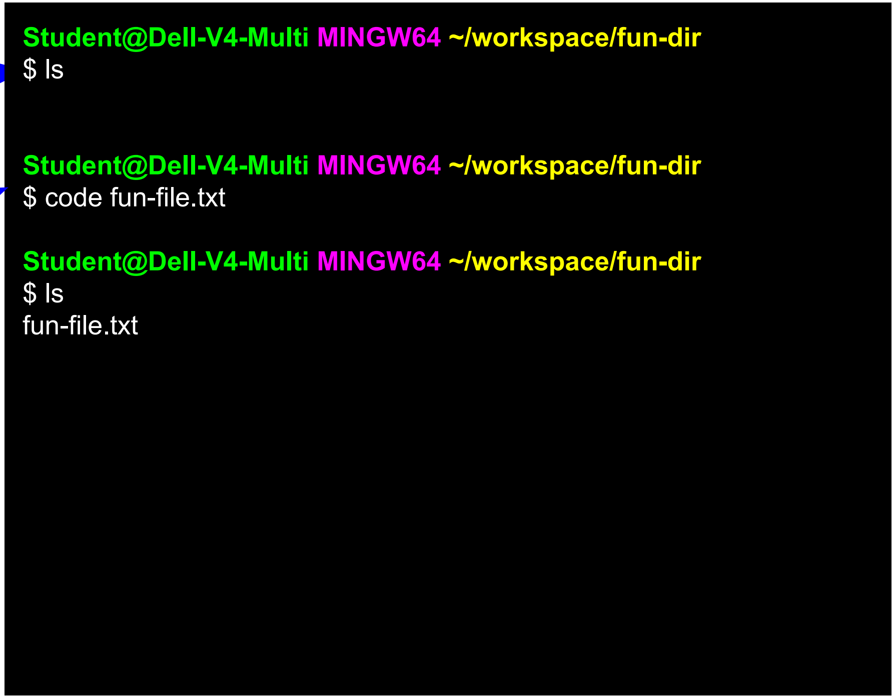
Command Line Commands: View File Contents

Let's create a file and view its contents...

If we use the **ls** command in the current directory, we see there are no files in it.

We can create a file using VIsual Studio Code by using the **code** command followed by the name of the file we want to create. This will create a new file and open it with VS Code

Now when we use the **ls** command, we see our new file.



```
Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ ls


Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ code fun-file.txt

Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ ls
fun-file.txt
```

Command Line Commands: View File Contents

Let's create a file and view its contents...

If we use the **ls** command in the current directory, we see there are no files in it.

We can create a file using VIsual Studio Code by using the **code** command followed by the name of the file we want to create. This will create a new file and open it with VS Code

Now when we use the **ls** command, we see our new file.

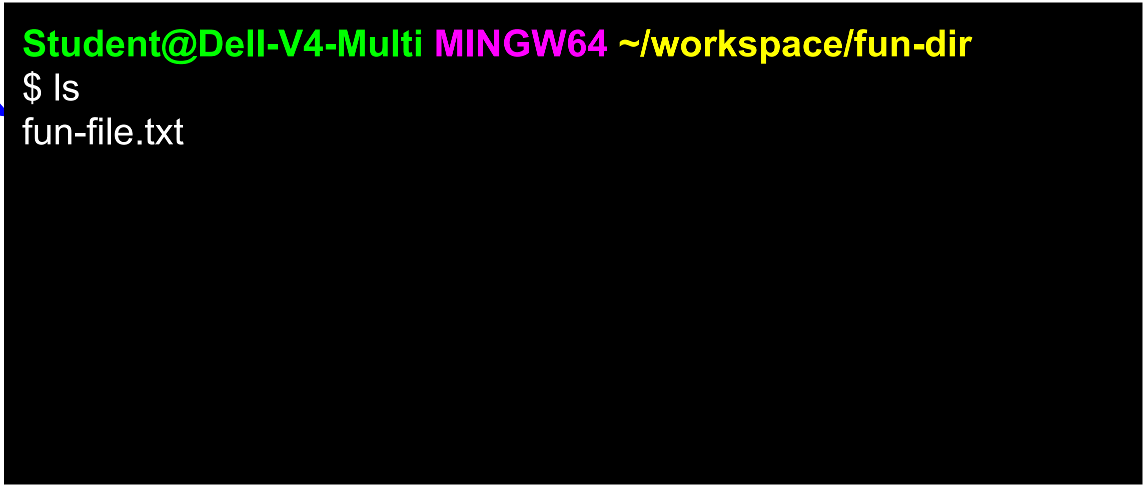
We can output the content of the file we created using the **cat** command.

```
Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ ls


Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ code fun-file.txt

Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ ls
fun-file.txt

Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ cat fun-file.txt
Hello, world!
```

Command Line Commands: Copying

To copy a file from one directory to another: **cp <source> <destination>**.`

If we use the **ls** command in the current directory, we see only the **`fun-file.txt`** file.



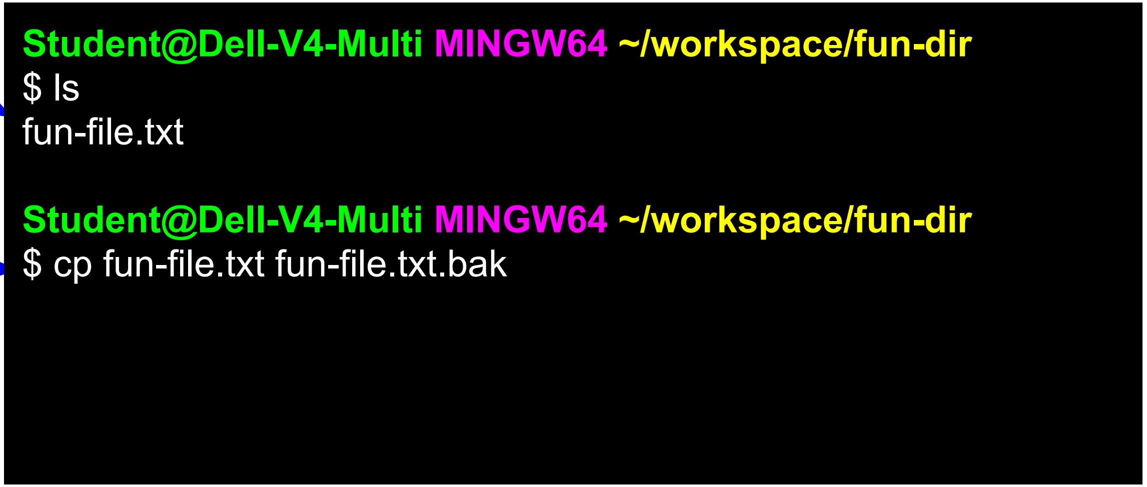
```
Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ ls
fun-file.txt
```

Command Line Commands: Copying

To copy a file from one directory to another: **cp <source> <destination>**.`

If we use the **ls** command in the current directory, we see only the **`fun-file.txt`** file.

We execute **cp fun-file.txt fun-file.txt.bak** to make a copy of the file with a new name.



```
Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ ls
fun-file.txt

Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ cp fun-file.txt fun-file.txt.bak
```

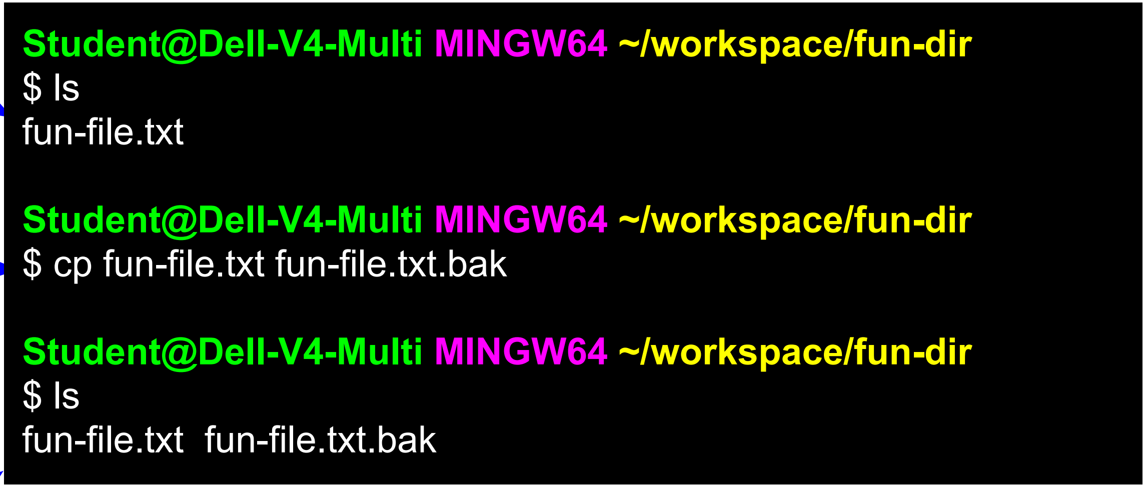
Command Line Commands: Copying

To copy a file from one directory to another: **cp <source> <destination>**.`

If we use the **ls** command in the current directory, we see only the **`fun-file.txt`** file.

We execute **cp fun-file.txt fun-file.txt.bak** to make a copy of the file with a new name.

Now if we use the **ls** command in the current directory, we see **`fun-file.txt`** along with the copy we made of it (**`fun-file.txt.bak`**).

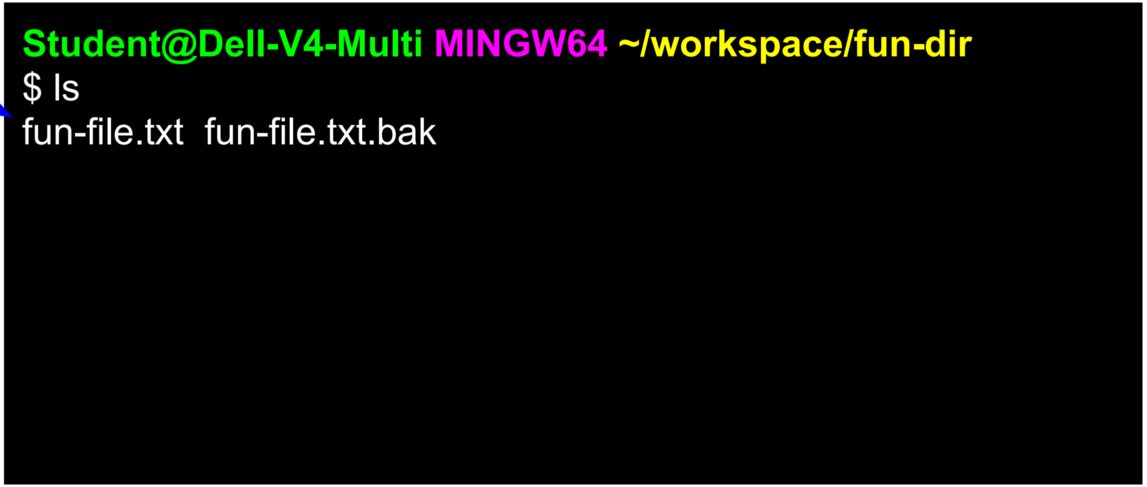


```
Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ ls
fun-file.txt

Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ cp fun-file.txt fun-file.txt.bak

Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ ls
fun-file.txt  fun-file.txt.bak
```

Command Line Commands: Moving

To move a file from one directory to another: **mv <source> <destination>**.`

If we use the **ls** command in the current directory, we see that the **`fun-file.txt.bak`** file exists.



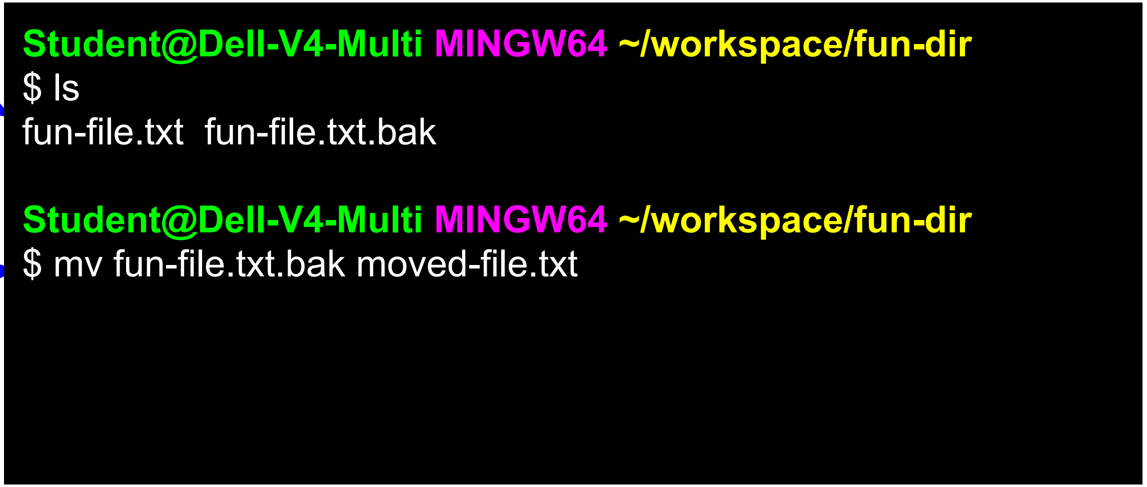
```
Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ ls
fun-file.txt  fun-file.txt.bak
```

Command Line Commands: Moving

To move a file from one directory to another: **mv <source> <destination>**.`

If we use the **ls** command in the current directory, we see that the **`fun-file.txt.bak`** file exists.

We execute
mv fun-file.txt.bak moved-file.txt
to make move the file to a new path.



```
Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ ls
fun-file.txt  fun-file.txt.bak

Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ mv fun-file.txt.bak moved-file.txt
```

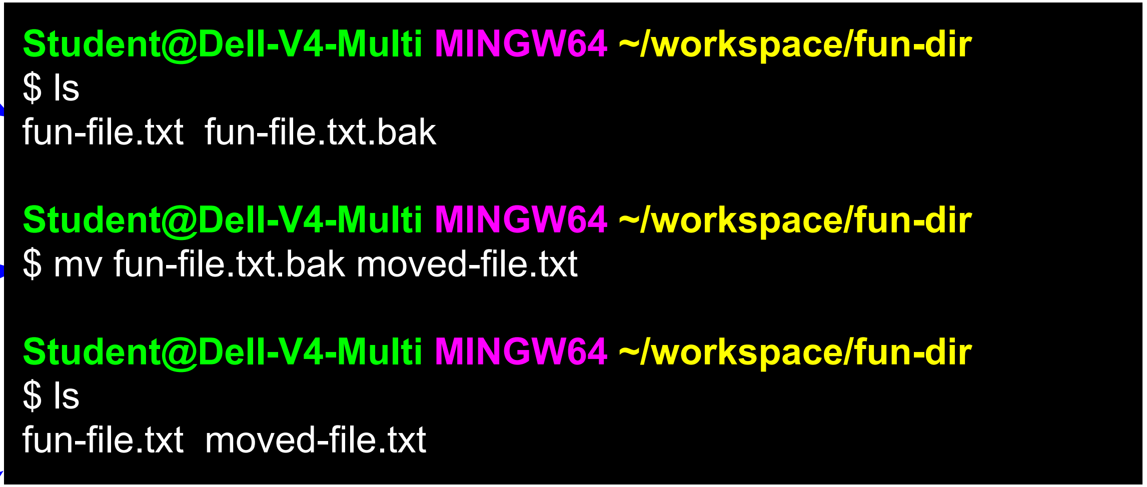
Command Line Commands: Moving

To move a file from one directory to another: **mv <source> <destination>**.`

If we use the **ls** command in the current directory, we see that the **`fun-file.txt.bak`** file exists.

We execute
mv fun-file.txt.bak moved-file.txt
to make move the file to a new path.

Now if we use the **ls** command in the current directory, we see **`fun-file.txt.bak`** file has been renamed to **`moved-file.txt`**.and the original file is no longer present.



```
Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ ls
fun-file.txt  fun-file.txt.bak

Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ mv fun-file.txt.bak moved-file.txt

Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ ls
fun-file.txt  moved-file.txt
```

Command Line Commands: Moving

To move a file from one directory to another: **mv <source> <destination>**.`

If we use the **ls** command in the current directory, we see that the **`fun-file.txt.bak`** file exists.

```
Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ ls
fun-file.txt  fun-file.txt.bak

Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ mv fun-file.txt.bak moved-file.txt

Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ ls
fun-file.txt  moved-file.txt
```

We execute
mv fun-file.txt.bak moved-file.txt
to make move the file to a new path.

Now if we use the **ls** command in the current directory, we see **`fun-file.txt.bak`** file has been renamed to **`moved-file.txt`**.and the original file is no longer present.

In this case, the path we specified was in the same directory so the file was just renamed but we could use the command with a different path to move it elsewhere with the same name (i.e. **mv fun-file.txt.bak ~/workspace/fun-file.txt.bak**) or to move it elsewhere AND rename it (i.e.**mv fun-file.txt.bak ~/workspace/moved-file.txt**).

Command Line Commands: cp/mv Shorthand

If you are copying or moving a file to another location but want to leave the name as is, you can omit the filename in the destination.

This will move the file **`moved-file.txt`** to the **`~/workspace/`** directory. The file will no longer exist in this directory and will now exist as **`~/workspace/moved-file.txt`**

```
Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ mv moved-file.txt ~/workspace/
```

Command Line Commands: cp/mv Shorthand

If you are copying or moving a file to another location but want to leave the name as is, you can omit the filename in the destination.

This will move the file **`moved-file.txt`** to the **`~/workspace/`** directory. The file will no longer exist in this directory and will now exist as **`~/workspace/moved-file.txt`**

This will make a copy if **`fun-file.txt`** in the **`~/workspace/`** directory. The file will still exist in this directory but will also exist as **`~/workspace/fun-file.txt`**

```
Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ mv moved-file.txt ~/workspace/

Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ cp fun-file.txt ~/workspace/
```

Command Line Commands: cp/mv Shorthand

If you are copying or moving a file to another location but want to leave the name as is, you can omit the filename in the destination.

```
Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ mv moved-file.txt ~/workspace/

Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ cp fun-file.txt ~/workspace/

Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ mv ~/workspace/moved-file.txt .
```

This will move the file `moved-file.txt` to the `~/workspace/` directory. The file will no longer exist in this directory and will now exist as `~/workspace/moved-file.txt`

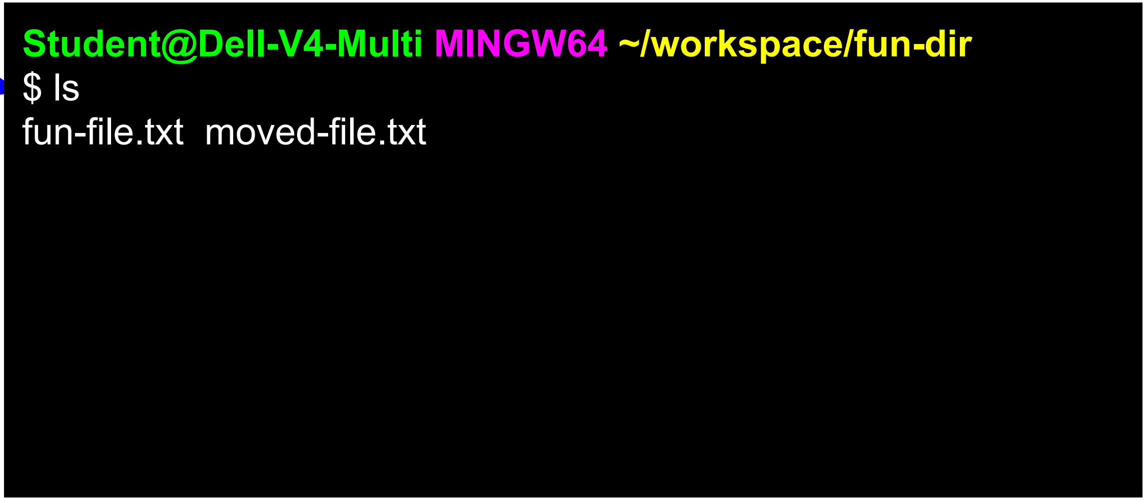
This will make a copy if `fun-file.txt` in the `~/workspace/` directory. The file will still exist in this directory but will also exist as `~/workspace/fun-file.txt`

Similarly, you can move a file to or make a copy of it with the same name in the current directory by specifying `.` (the "current directory" indicator) as the destination

Command Line Commands: Removing Files

We can remove a file using the **rm <filename>** command.

If we execute the **ls** command, we see two files



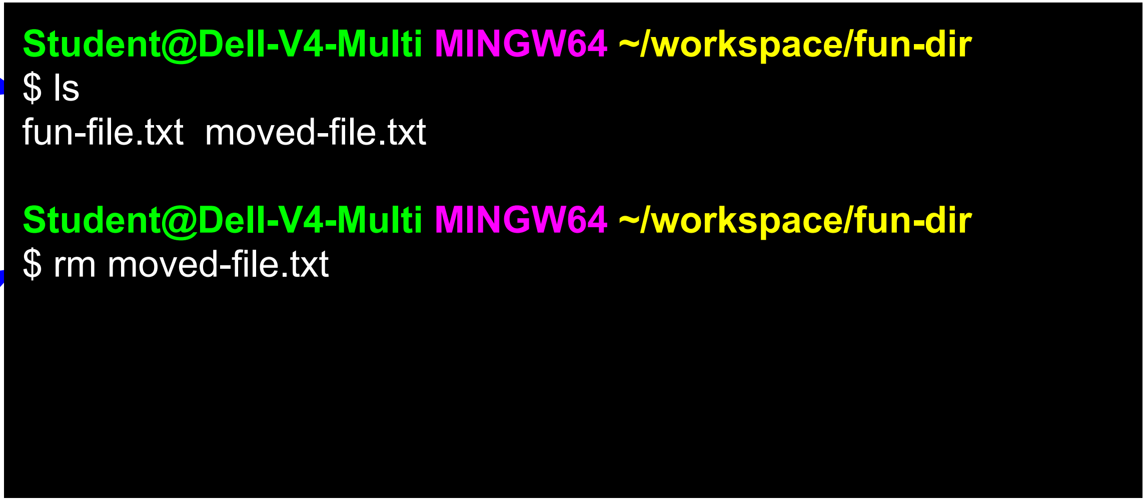
```
Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ ls
fun-file.txt  moved-file.txt
```

Command Line Commands: Removing Files

We can remove a file using the **rm <filename>** command.

If we execute the **ls** command, we see two files

We can use the **rm** command with a filename to remove a file.



```
Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ ls
fun-file.txt  moved-file.txt

Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ rm moved-file.txt
```

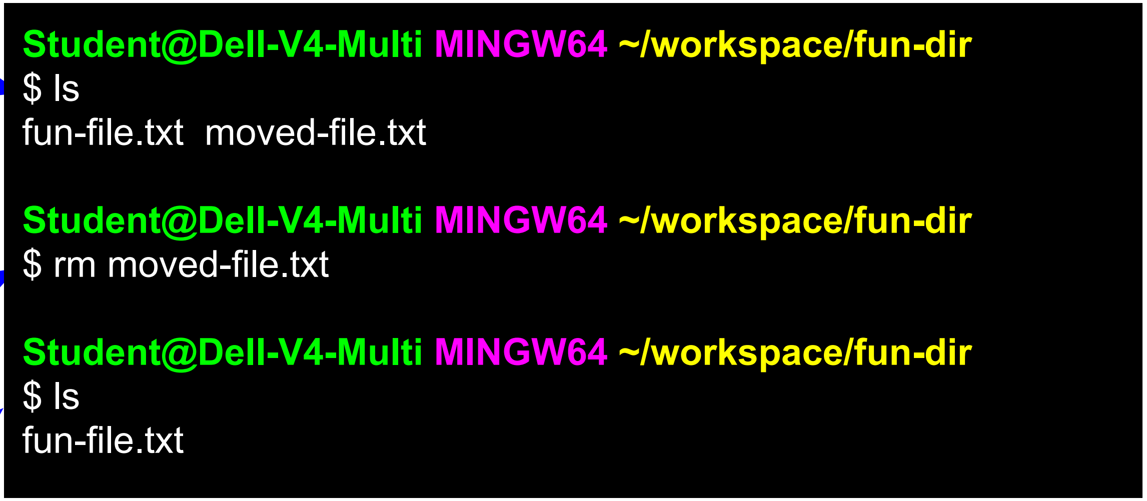
Command Line Commands: Removing Files

We can remove a file using the **rm <filename>** command.

If we execute the **ls** command, we see two files

We can use the **rm** command with a filename to remove a file.

Now if we execute an **ls**, **`the moved-file.txt`** file is no longer in the directory.



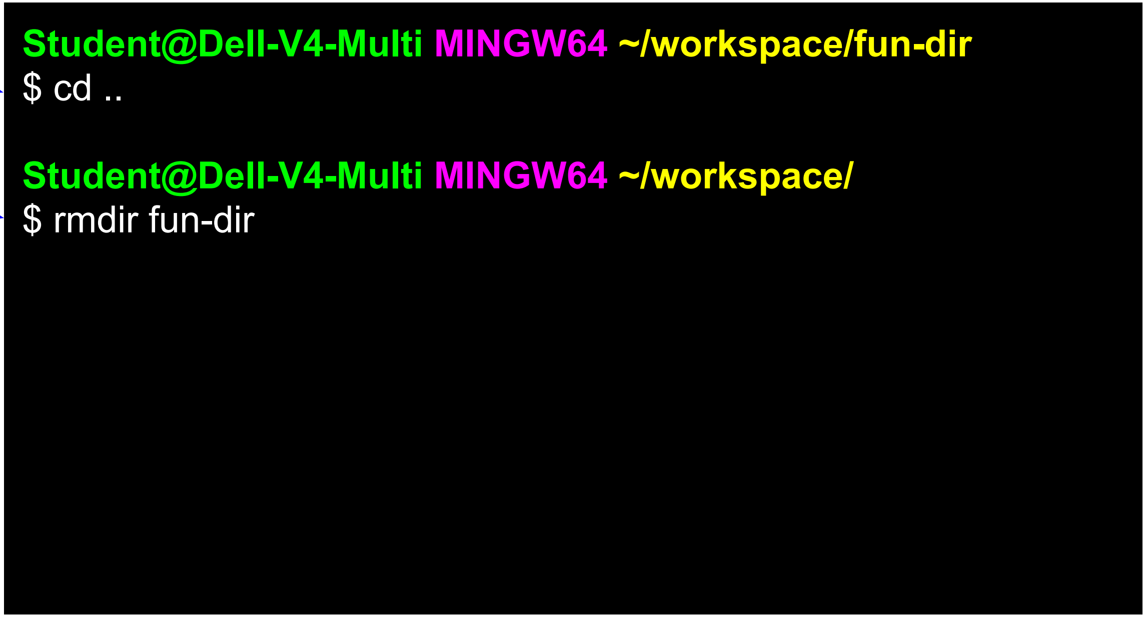
```
Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ ls
fun-file.txt  moved-file.txt

Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ rm moved-file.txt

Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ ls
fun-file.txt
```

Command Line Commands: Removing Directories

We can remove a directory using the **rmdir <directory name>** command.

If we **cd** back to the **`~/workplace`** directory...

We can use the **rmdir** command with a directory name to remove a directory.



```
Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ cd ..

Student@Dell-V4-Multi MINGW64 ~/workspace/
$ rmdir fun-dir
```

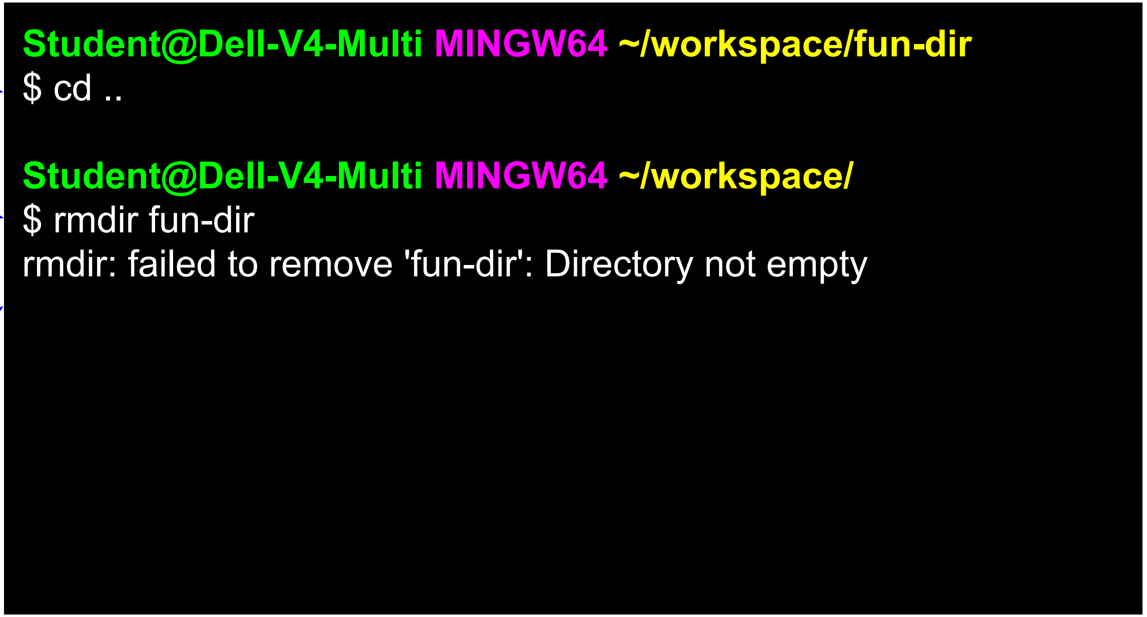
Command Line Commands: Removing Directories

We can remove a directory using the **rmdir <directory name>** command.

If we **cd** back to the **`~/workplace`** directory...

We can use the **rmdir** command with a directory name to remove a directory.

What happened? Directories must be empty before they can be removed.



```
Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ cd ..

Student@Dell-V4-Multi MINGW64 ~/workspace/
$ rmdir fun-dir
rmdir: failed to remove 'fun-dir': Directory not empty
```

Command Line Commands: Removing Directories

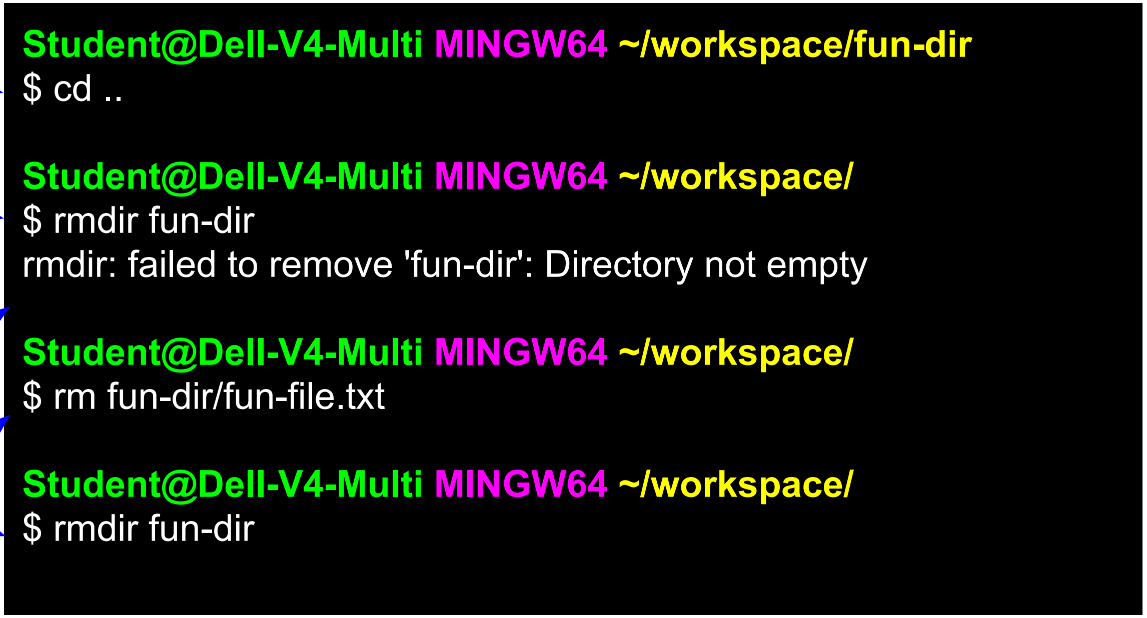
We can remove a directory using the **rmdir <directory name>** command.

If we **cd** back to the ~/workplace directory...

We can use the **rmdir** command with a directory name to remove a directory.

What happened? Directories must be empty before they can be removed.

We need to remove `fun-file.txt` from the fun-dir directory in order to remove the directory.



```
Student@Dell-V4-Multi MINGW64 ~/workspace/fun-dir
$ cd ..

Student@Dell-V4-Multi MINGW64 ~/workspace/
$ rmdir fun-dir
rmdir: failed to remove 'fun-dir': Directory not empty

Student@Dell-V4-Multi MINGW64 ~/workspace/
$ rm fun-dir/fun-file.txt

Student@Dell-V4-Multi MINGW64 ~/workspace/
$ rmdir fun-dir
```

Let's Try it Together!

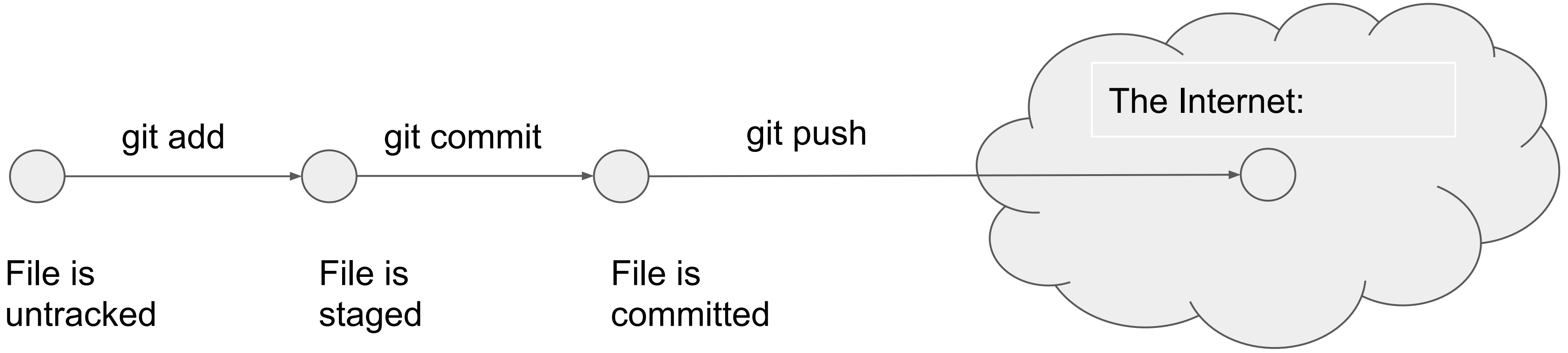
Introducing: Version Control

Version Control : What it is

- Version control systems record changes to a file or sets of files so that previous versions can be recalled at a later point in time.
- In this class, we will be using git with GitLab.
- Git is an example of a distributed version control system, where a repository exists locally on your own workstation and on a central network location.

Version Control : Git Flow (Checking In Changes)

- **git status**: See the current status of your files.
- **git add -A**: Stage any files you have changed.
- **git add <path>**: Stage specific file(s).
- **git commit -m "Commit message"**: Save files to your local repository.
- **git push origin main**: Push committed changes to network repository.



Version Control : Git Flow (Checking In Changes)

- **git clone**: Pulls the entire repository (including all previous commits) to your local workstation.
- **git pull upstream main**: Pulls latest changes from the remote repository.
- In this class we make a distinction between “upstream main” and “origin main”. Always pull from upstream main and push to origin main! There are some circumstances where this will change - the instructor will let you know.

Final Notes

- You want to pull often:
    - Pull when your instructors ask you to.
    - Pull first thing in the morning when you get to class.
    - Pull when you get back from lunch
    - Pull before you plan to push an assignment.
- Instructors will only grade what has been pushed to the GitLab git repository.
    - You can always check the web version of the repository to do a spot check to make sure what you pushed is actually there:

      https://git.techelevator.com/campuses/wlm/jan-2022/java/student-code/<your-name>-student-code