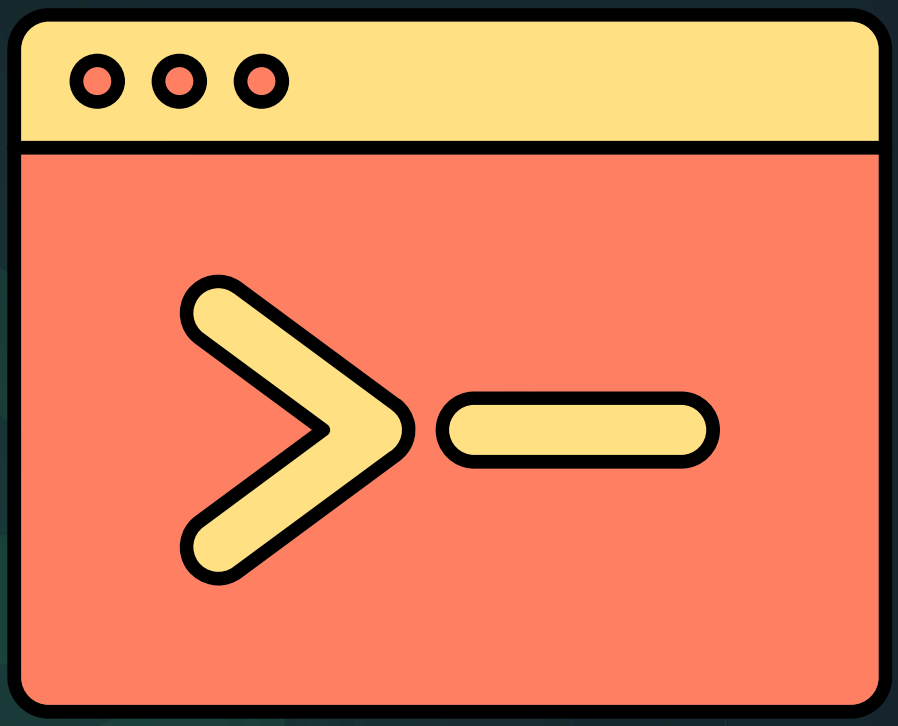
Command Line



Version Control

Let's say you are working on a web development project with a team of developers. The project involves creating a website for a client. Each team member is responsible for different sections of the website, such as the homepage, product pages, and contact form.

Without version control:

- Developer A starts working on the homepage and makes significant changes to the HTML and CSS files.
- Meanwhile, Developer B is working on the product pages and modifies some shared JavaScript files to implement certain functionalities.

- Developer C is responsible for the contact form and makes changes to the HTML and PHP files.

Now, imagine that after a few days of working independently, the team gathers to integrate their changes into the final version of the website. They realize that there are conflicts and inconsistencies in the code. Some changes made by Developer A conflict with the modifications made by Developer B and Developer C. It becomes challenging to determine which changes should be kept and how to merge them correctly.

With version control (e.g., Git):

- Each developer creates a separate branch for their assigned task.

- Developer A works on the homepage in their branch, Developer B on the product pages in another branch, and Developer C on the contact form in their own branch.

- As they progress, they commit their changes to their respective branches and push them to the version control system.

- Git keeps track of the individual changes made by each developer.

So far, we have learned why we use version control. Now, let's study what a version control system is.

What is version control.

Version control, also known as source control, is the practice of tracking and managing changes to software code. Version control systems are software tools that help software teams manage changes to source code over time.

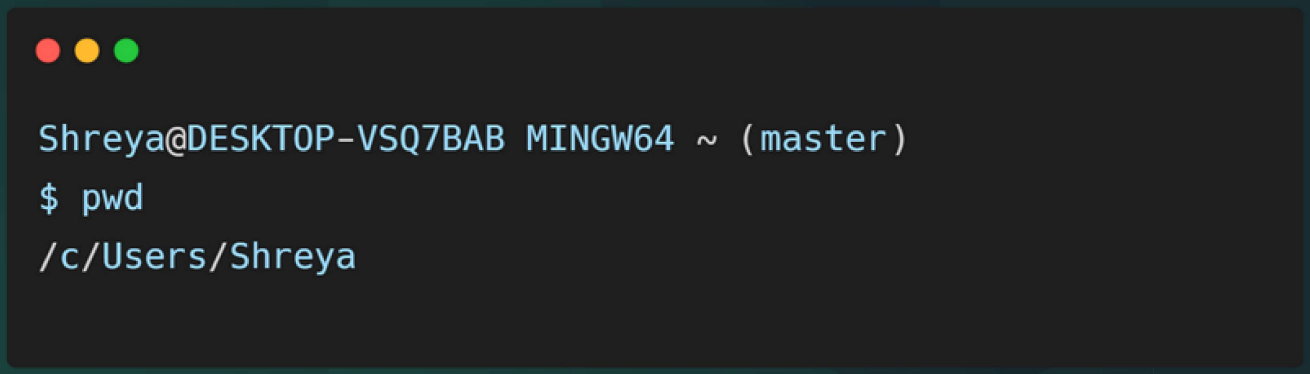
Basic Command Lines

Your computer's text interface is called the command line. It is a programme that receives instructions and sends them to the computer's operating system for execution. From the command line, you can navigate through files and folders on your computer, just as you would with Windows Explorer on Windows or Finder on Mac OS.

To master the fundamentals of the Unix command, we will utilise git bash in this lesson.

Current working directory

The directory that the user is presently working in is known as the current working directory. You are operating inside a directory each time you use your command prompt. Your home directory is configured as your current working directory by default when you log into a system



Navigating directory

You can switch folders with the cd command. You are in your home directory when you launch a terminal. You will use cd to navigate the file system.

- To navigate to your home directory, use "cd"
- To navigate up one directory level, use "cd .."

```
Shreya@DESKTOP-VSQ7BAB MINGW64 ~ (master)
$ cd desktop
```

Use the 'pwd' command to see where you are.

```
Shreya@DESKTOP-VSQ7BAB MINGW64 ~/desktop (master)
$ pwd
/c/Users/Shreya/desktop
```

Making Directory

The command that allows you to create directories is 'mkdir'.

```
Shreya@DESKTOP-VSQ7BAB MINGW64 ~/desktop (master)
$ mkdir Git-course-Geekster
```

Now let's go to the Git-course-Geekster folder using 'cd' command.

```
Shreya@DESKTOP-VSQ7BAB MINGW64 ~/desktop (master)
$ cd Git-course-Geekster
Shreya@DESKTOP-VSQ7BAB MINGW64 ~/desktop/Git-course-
Geekster (master)
$ pwd
/c/Users/Shreya/desktop/Git-course-Geekster
```

Note: Remember that when you do not mention a more than one-word directory name, it will create two directories.

List files and directories

The 'ls' command is used to list files or directories. Just like you navigate in your File explorer or Finder with a GUI, the 'ls' command allows you to list all files or directories in the current directory by default, and further interact with them via the command line.

```
Shreya@DESKTOP-VSQ7BAB MINGW64 ~/desktop/Git-course-
Geekster (master)
$ ls
```

Note: It is worth noting that 'ls' won't show the hidden folders. You need to use 'ls -a' for the same.

Git-course-Geekster folder didn't include any files or directories because we hadn't yet created them. so, Let's build a few directories.

```
Shreya@DESKTOP-VSQ7BAB MINGW64 ~/desktop/Git-course-
Geekster (master)
$ mkdir Day-01
Shreya@DESKTOP-VSQ7BAB MINGW64 ~/desktop/Git-course-
Geekster (master)
$ ls
Day-01/
```

You can create directories one by one with 'mkdir', but this can be time-consuming. To avoid that, you can run a single mkdir command to create multiple directories at once.

```
Shreya@DESKTOP-VSQ7BAB MINGW64 ~/desktop/Git-course-
Geekster (master)
$ mkdir day2 day3 day4 day5

Shreya@DESKTOP-VSQ7BAB MINGW64 ~/desktop/Git-course-
Geekster (master)
$ ls
Day-01/  day2/  day3/  day4/  day5/
```

Detail List

Let's use the 'ls -la' command to see a detailed list of the directories.

```
Shreya@DESKTOP-VSQ7BAB MINGW64 ~/desktop/Git-course-
Geekster (master)
$ ls -la
total 16
drwxr-xr-x 1 Shreya 197121 0 Aug 24 12:19 ./
drwxr-xr-x 1 Shreya 197121 0 Aug 24 12:01 ../
drwxr-xr-x 1 Shreya 197121 0 Aug 24 12:11 Day-01/
drwxr-xr-x 1 Shreya 197121 0 Aug 24 12:19 day2/
drwxr-xr-x 1 Shreya 197121 0 Aug 24 12:19 day3/
drwxr-xr-x 1 Shreya 197121 0 Aug 24 12:19 day4/
drwxr-xr-x 1 Shreya 197121 0 Aug 24 12:19 day5/
```

Using the above command, we can see the detailed view of a directory or a file

Creating file

Its is a very simple to create file in git bash at first write touch then file name with extension.

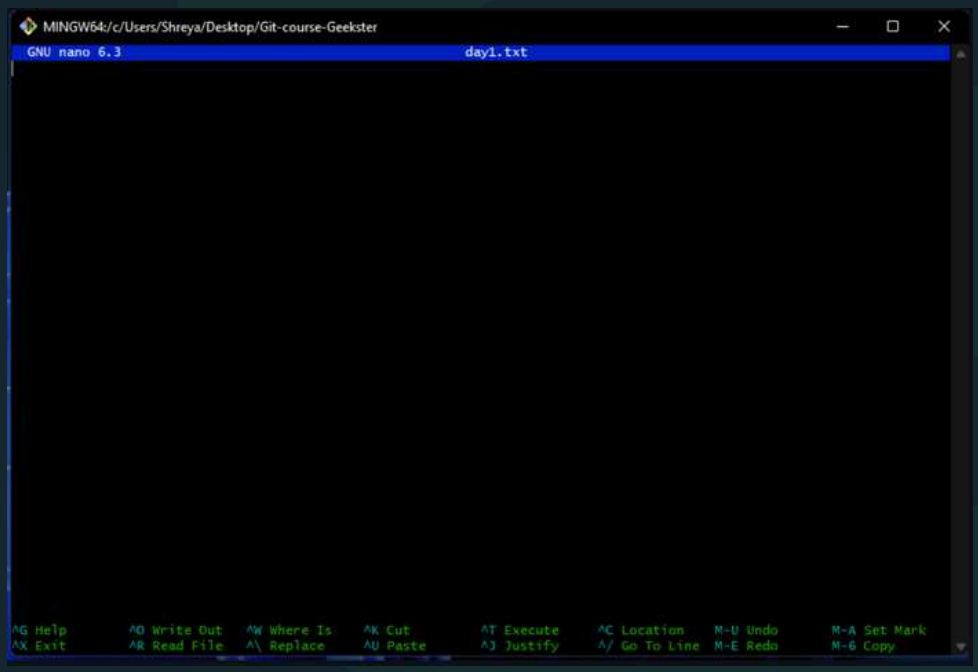
```
Shreya@DESKTOP-VSQ7BAB MINGW64 ~/Desktop/Git-course-
Geekster (master)
$ touch day1.txt

Shreya@DESKTOP-VSQ7BAB MINGW64 ~/Desktop/Git-course-
Geekster (master)
$ ls
Day-01/  day1.txt  day2/  day3/  day4/  day5/
```

As you can see, there is day1.txt in the list. That means we have created the day1.txt file inside the Git-course-Geekster folder. You can also you ls -la command to use the detailed view of the folders and file.

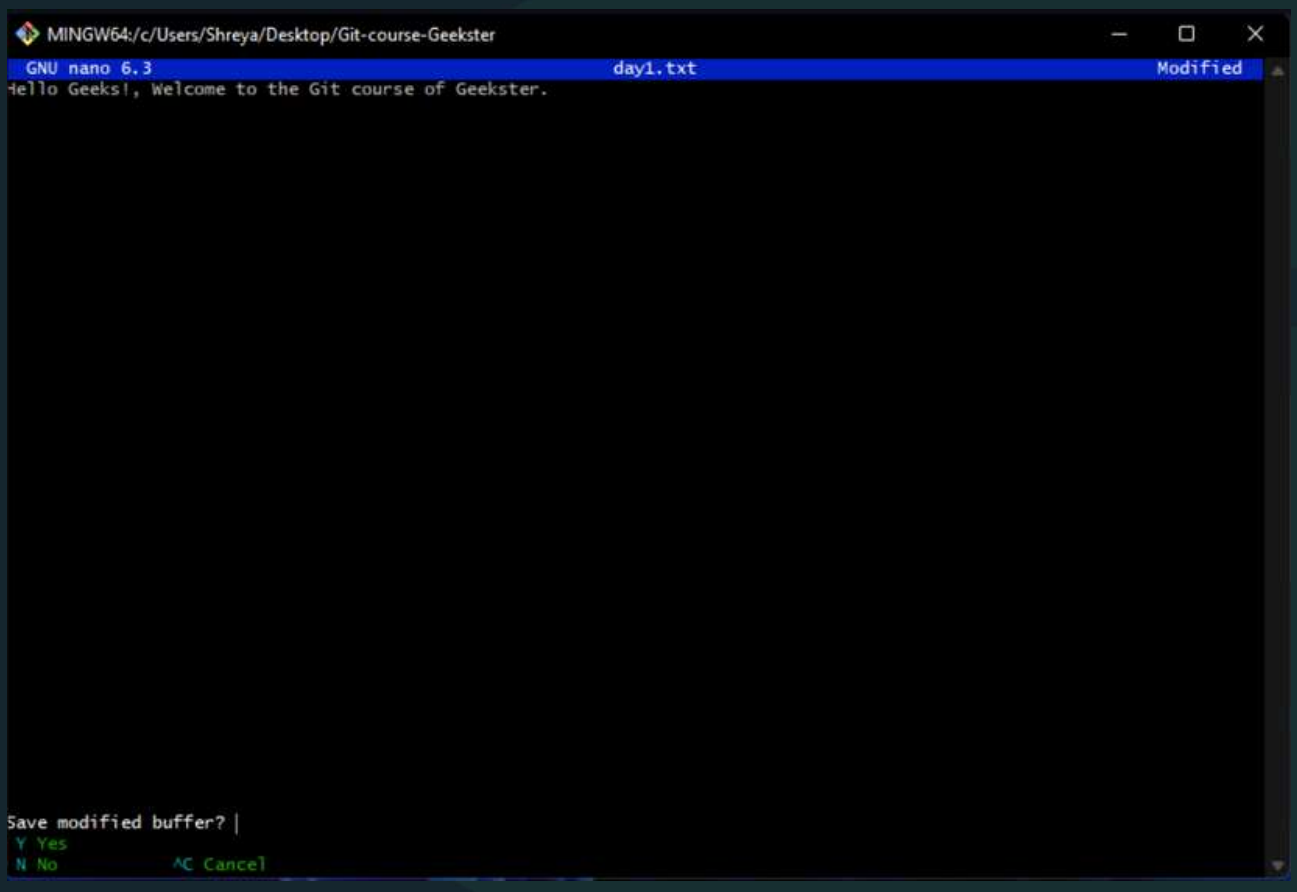
Opening and writing on file

Let's open day1.txt and start adding text to it. In order to open and write, we need the 'nano' command.

```
Shreya@DESKTOP-VSQ7BAB MINGW64 ~/Desktop/Git-course-
Geekster (master)
$ nano day1.txt
```



As you can see from the above figure, the cursor is active and you can write on the pad. You can only use arrow **keys to move the cursor left, right, up, and down. Let's write some text on the opened pad. There are instructors at the bottom that tells how to exit. For instance ctrl + x is to exit. When you exit either you save or cancel which comes when you click ctrl + x.**



Now you can save the modified file by writing Y or you can cancel it by clicking ctrl + c.
After you write Y then click enter.

Opening file to read

The 'cat' command can only be used to read files.

```
Shreya@DESKTOP-VSQ7BAB MINGW64 ~/Desktop/Git-course-
Geekster (master)
$ cat day1.txt
Hello Geeks!, Welcome to the Git course of Geekster.
```

Copy file

The command cp followed by the name of the file you want to copy and the name of the directory to where you want to copy the file (e.g. cp filename directory-name)
Let's have day1-backup.txt from day1.txt by copying using the cp command

```
Shreya@DESKTOP-VSQ7BAB MINGW64 ~/Desktop/Git-course-
Geekster (master)
$ cp day1.txt day1-backup.txt

Shreya@DESKTOP-VSQ7BAB MINGW64 ~/Desktop/Git-course-
Geekster (master)
$ ls
Day-01/  day1-backup.txt  day1.txt  day2/  day3/  day4/
 day5/
```

Rename file

We use the git mv command in git to rename and move files. We only use the command for convenience. It does not rename or move the actual file, but rather deletes the existing file and creates a new file with a new name or in another folder.

```
git mv options oldFilename newFilename
```

oldFilename: The name of the file that we rename.
newFilename: The new name of the file.

let's rename the day1.txt to day-one.txt using the mv command.

```
Shreya@DESKTOP-VSQ7BAB MINGW64 ~/Desktop/Git-course-
Geekster (master)
$ mv day1.txt day-one.txt

Shreya@DESKTOP-VSQ7BAB MINGW64 ~/Desktop/Git-course-
Geekster (master)
$ ls
Day-01/  day-one.txt  day1-backup.txt  day2/  day3/  day4/
 day5/
```

Moving file and directory

The mv and cp commands can be used to put files into a directory. The cp moves the copy of the file or the folder to another folder, however, mv just move it without copying. Let's move the day-one.txt Day-01 folder.

```
Shreya@DESKTOP-VSQ7BAB MINGW64 ~/Desktop/Git-course-
Geekster (master)
$ mv day-one.txt Day-01

Shreya@DESKTOP-VSQ7BAB MINGW64 ~/Desktop/Git-course-
Geekster (master)
$ ls
Day-01/  day1-backup.txt  day2/  day3/  day4/  day5/

Shreya@DESKTOP-VSQ7BAB MINGW64 ~/Desktop/Git-course-
Geekster (master)
$ cd Day-01

Shreya@DESKTOP-VSQ7BAB MINGW64 ~/Desktop/Git-course-
Geekster/Day-01 (master)
$ ls
day-one.txt
```

Delete file and directory

To delete (i.e. remove) a directory and all the sub-directories and files that it contains, navigate to its parent directory, and then use the command rm -r followed by the name of the directory you want to delete (e.g. rm -r directory-name). Let's remove the day-one.txt file from the Day-01 folder.

```
Shreya@DESKTOP-VSQ7BAB MINGW64 ~/Desktop/Git-course-
Geekster (master)
$ cd Day-01

Shreya@DESKTOP-VSQ7BAB MINGW64 ~/Desktop/Git-course-
Geekster/Day-01 (master)
$ ls
day-one.txt

Shreya@DESKTOP-VSQ7BAB MINGW64 ~/Desktop/Git-course-
Geekster/Day-01 (master)
$ touch day-two.txt
```

```
Shreya@DESKTOP-VSQ7BAB MINGW64 ~/Desktop/Git-course-
Geekster/Day-01 (master)
$ ls
day-one.txt  day-two.txt

Shreya@DESKTOP-VSQ7BAB MINGW64 ~/Desktop/Git-course-
Geekster/Day-01 (master)
$ rm day-one.txt

Shreya@DESKTOP-VSQ7BAB MINGW64 ~/Desktop/Git-course-
Geekster/Day-01 (master)
$ ls
day-two.txt
```

The rmdir delete a folder.Let's delete the day5 folder.

```
Shreya@DESKTOP-VSQ7BAB MINGW64 ~/Desktop/Git-course-
Geekster (master)
$ ls
Day-01/  day1-backup.txt  day2/  day3/  day4/  day5/

Shreya@DESKTOP-VSQ7BAB MINGW64 ~/Desktop/Git-course-
Geekster (master)
$ rmdir day5

Shreya@DESKTOP-VSQ7BAB MINGW64 ~/Desktop/Git-course-
Geekster (master)
$ ls
Day-01/  day1-backup.txt  day2/  day3/  day4/
```

1. rm-r removes the file "file.txt" from the repository's staging area, and it will be deleted in the next commit.
2. To remove a directory (and its contents) from the repository:

This command removes the file "file.txt" from the repository's staging area, and it will be deleted in the next commit.

Git and GitHub

Git: During the course of software development, changes to source code are monitored using the Git distributed version control system. Although it is intended for programmers, it may be used to keep track of changes to any set of files. Speed, data integrity, and support for dispersed, non-linear workflows are among its objectives.

GitHub: GitHub is a web-based service for hosting Git repositories. It provides all of Git's distributed revision control and source code management (SCM) functions in addition to a few extras.

Before using Git we should know why we need it

- Git makes it easy to contribute to open-source projects
- Documentation.
- Integration Option.
- Track changes in your code across versions
- Deployment.

1. Install Git

First, you need to install the version control software, Git.

2. Checking status of the repository

The git status command displays the state of the working directory and the staging area. It lets you see which changes have been staged, which haven't, and which files aren't being tracked by Git.

```
git status
```

3. Configure your name and your email

The global git username and email address are associated with commits on all repositories on your system that don't have repository-specific values.

To set your global commit name and email address run the git config command with the --global option.

```
git config --global user.name "Your Name"
git config --global user.email "youremail@yourdomain.com"
```

4. Create a local git repository

You will create a folder (directory) for your project in this stage. A project is only a folder where all the files relevant to a particular project are kept. A project or folder on your PC is referred to as a local repository.

```
mkdir project_name
cd project_name
```