Getting Started With Git
Today’s Lecture:

1. Introduction to Unix/Linux/Bash
   Fundamentals of the command line

2. Introduction to Git
   Fundamentals of Git, version control, basic commands, ssh-keys

3. Demo/example
   Manage Your Own Repo!
Unix/Linux/Bash

Fundamentals of the command line
Linux Command Line
Basics of Command Line – 1

- **ls**: Lists the contents of the current or target directory.
- **cd**: Moves into the target directory.
- **pwd**: Shows the path to your current directory.
viewing the contents of the git_class directory
pwd

viewing the path of the current git_class directory

$ git_examples pwd
/Users/sanjay/Desktop/classes/git_examples
cd moving into the Test_Repo directory

```sh
$ cd
/Users/sanjay/Desktop/classes/git_class
$ cd Test_Repo
$ Test_Repo git:(master) cd ..
```

cd .. moves me back out of the Test_Repo directory
Basics of Command Line – 2

**cp**
Copies a specific file to a target directory.

**mv**
Moves a specific file/directory to a target directory. This command is also used for renaming.

**rm**
Removes a specified file. Add the `-r` flag to recursively delete a directory.
copying the random.txt into Test_Repo directory

$ git_class ls
Test_Repo  random.txt
$ git_class cp random.txt Test_Repo
$ git_class ls Test_Repo
random.txt
$ git_class ls
mv

renaming random.txt

```
Test_Repo  random.txt
[→] git_examples  mv random.txt test.txt
[→] git_examples  ls
Test_Repo  test.txt
```

moving gitiscool.txt to Test_Repo

```
[→] git_examples  mv test.txt.txt Test_Repo
[→] git_examples  ls Test_Repo
random.txt  test.txt
```
deleting random.txt

```
$ rm
$ ls
random.txt test.txt
$ rm random.txt
$ ls

```  

deleting Random_Repo, along with all its contents recursively

```
$ git ls
Random_Repo Test_Repo
$ rm -r Random_Repo
$ ls

```
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>cat</code></td>
<td>Displays the contents of a file. Also performs file creation and concatenation.</td>
</tr>
<tr>
<td><code>less</code></td>
<td>A dedicated file reader that displays the contents of a file one screen at a time.</td>
</tr>
<tr>
<td><code>mkdir</code></td>
<td>Makes a new directory at the specified target. If not target is provided, it assumes the current directory.</td>
</tr>
</tbody>
</table>
viewing the content of gitiscool.txt

```
$ Test_Repo git:(main) × ls
  test.txt
$ Test_Repo git:(main) × cat test.txt
  git is actually cool. Trust me!
```

making a new file

```
$ Test_Repo git:(main) × cat > random.txt
  test!!

^C
$ Test_Repo git:(main) × cat random.txt
  test!!
```
viewing the content of gitiscool.txt

→ Test_Repo git:(main) ✗ less random.txt

remember to press “q” to leave the editor

test!!

random.txt (END)
mkdir

make the Test_Repo directory

```
[→] git_class ls
[→] git_class mkdir Test_Repo
[→] git_class ls
Test_Repo
[→] git_class
```
Basics of Command Line – 4

**echo**

writes any of its parameters to standard output.

**sudo**

Run commands as a different user with possibly different security privileges.

**man**

Gives the user information regarding a specific command.
‘echoing’ CMSC388T in the shell
using sudo to run the command “echo CMSC388T” with root privileges
get more information about “sudo”
(press “q” to quit)

NAME
sudo, sudoedit - execute a command as another user

SYNOPSIS
sudo [-h | -k | -K | -y | -v [-A allnS] [-g group] [-h host] [-p prompt] [-u user]
sudo [-l [-A allnS] [-g group] [-h host] [-p prompt] [-u user] [command]
sudo [-A allnS] [-g group] [-h host] [-p prompt] [-T timeout] [-u user] [VAR=value] [-i | -s] [command]
sudoedit [-A allnS] [-g group] [-h host] [-p prompt] [-T timeout] [-u user] file ...

DESCRIPTION
sudo allows a permitted user to execute a command as the superuser or another user, as specified by the security policy. The invoking user's real (not effective) user-ID is used to determine the user name with which to query the security policy.

sudo supports a plugin architecture for security policies and input/output logging. Third parties can develop and distribute their own policy and I/O logging plugins to work seamlessly with the sudo front end. The default security policy is sudoers, which is configured via the file /private/etc/sudoers, or via LDAP. See the Plugins section for more information.

The security policy determines what privileges, if any, a user has to run sudo. The policy may require that users authenticate themselves with a password or another authentication mechanism. If authentication is required, sudo will exit if the user's password is not entered within a configurable time limit. This limit is policy-specific; the default password prompt timeout for the sudoers security policy is unlimited.
Basics of Command Line – 5

- **alias**: Define your own commands.
- **unalias**: Get rid of a specific alias.
- **touch**: Creates a new file.
alias

alias "echo hello" with the string "hi"
unalias

remove this alias

[→ Test_Repo git:(main) × unalias hi
[→ Test_Repo git:(main) × hi
zsh: command not found: hi
touch

make test.txt

[→ git_examples  ls
Test_Repo
[→ git_examples  touch test.txt
[→ git_examples  ls
Test_Repo  test.txt
Basics of Command Line– 6

*Piping input and output*

<

Input for a file or command.

>

Output of a file or command.

**Examples:**

- `ls > temp` *(Pipes output of `ls` into a new file called `temp`)*
- `./a.out < temp` *(Pipes input of `temp` to an executable file called `a.out`)*
- `./a.out < temp`
- `./a.out > hi` *(Pipes input of `temp` to an executable file called `a.out` and then pipes the output to a file called `hi`)*
Text Editors

In Console:
- Nano
- Vim
- Emacs

External:
- Visual Studio Code
- Sublime Text
- Atom