Final Notes (Linux Fundamentals 2025)

How to clone a GitHub repository

To clone a GitHub repository, you use the **`git clone`** command in the terminal followed by the URL of the repository you want to copy.

Formula: `git clone "repository_url"`

Example:

- `git clone https://github.com/yourusername/cis106`
    - This will download a copy of the repository to your computer.
    - A new folder (cis106) will be created in your current directory
    - The folder will contain all the files from the GitHub repository.

How to use the git commands

Git commands are used to track changes, save work, and upload file to Github.

To use the git commands, after you make changes or finished your work you will have to click on **file** and click on **terminal** in VS code.

Once in the terminal window you will enter the 4 most useful git commands one by one in order.

The Git commands are :

- **`Git pull`**
    - Downloads the most recents changes from the remote Github repository and updates your local copy.
    - Helps prevent conflicts before making changes.
- **`git add .`**
    - Stages files so they are ready to be committed.
- **`git commit -m "message goes here"`**
    - The message explains what changes were made.
    - Commits are stored locally until pushed to Github.
- **`git push`**
    - Make the changes visible on GitHub.

How to write a Markdown file that contains images and proper formatting

To write a Markdown file with images + proper formatting you will need to make sure the file has the following:

- Headings `#, ##, ###, ####`
    - Headings are for titles and sub-titles. The
        - There are four heading styles
            - `# - Heading 1`

        - `## - Heaidng 2`
        - `### - Heading 3`
        - `#### - Heading 4`

This is the paragraph formatting to use in Markdown

- Bold `**text**`
    - Bold is for bolding any word or code
    - To bold you have to put two `*` before and the after the word or code.
    - Example:
        - `**hello**` = **hello**
- Italic `*word*`
    - To Italicize a word, put a `*` before and after the text
    - Example:
        - `*text*` = *This is Italic*

Code Formatting

- Code block

```
!#/bin/bash
echo "hello world"
```

Images

To add an image in a Markdown file, the image file must be the same directory as the Markdown file, or in a known subdirectory (such as an images/ folder)

If the image is `Downloads` or `Pictures`, it must be copied or moved to the same folder as the markdown file before it can be displayed correctly.

Image Synntax in Markdown

- `![decription of the image](image_filename.png)`
    - The text inside `![]` - is alternative text which describes the image.
    - The text inside `()` - is the path to the image file.
- Example : Image in a Subfolder
- If the image is stored in an `images directory`
    - `![AWK command output](images/awk_example.png)`

How to convert a Markdown file to PDF

Once a Markdown file is complete, is can be converted into a PDF using a Markdown extension in a code editor such as a Visual Studio Code.

One commonly used extension is Markdown PDF by yzane.

Steps to Convert Using Markdown PDF (VS code)

1. Install the Markdown PDF extension by yzane in Visual Studio Code.
2. Open the Markdown file you want to convert (for example, finalNotes.md).
3. Right-click anywhere inside the editor window.
4. Select “Markdown PDF: Export (pdf)”.
5. The PDF file will be created in the same directory as the Markdown file.

How to compress (zip) a directory/folder in Debian

To compress(zip) a directory or folder in Debian, you use the `zip` command in the terminal. This creates a single compressed file that contains all files and subdirectories.

- Syntax
  - `zip -r output_filename.zip directory_name/`
    - `-r` - means recursive, which allows all subfolders to be included.
    - `output_filename.zip` - is the name of the compresses file.
    - `directory_name/` - is the folder you want to compress
- Steps
  - Open the terminal
  - Navigate to the directory that contain the folder you want you want to compress.
  - Runs the `zip` command with the `-r`
- Example
  - Compress the directory `final_exam_study_notes` into a zip file
    - `zip -r final_exam_study_notes.zip final_exam_study_notes/`
      - The command creates a file named `final_exam_study_notes.zip` thats contains the entire folder and its contents.
- Installing `zip` (If not Installed)
  - If the `zip` command is not available, install it using:
    - `sudo apt update`
    - `sudo apt install zip -y`

What are Absolute paths and Relative paths?

Absolute path

Defintion

An absolute path is the full path to a file or directory starting from the root directory `(/)` of the system.

- Can think of it as the complete address of a file.

Example:

- Creating a file using Absolute Path
  - `touch /home/student/final_exam_study_notes/absoulute_file.txt`
    - This command creates a file named `absolute_file.txt` in the `final_exam_study_notes` directory, no matter where you are currently located in the terminal.'

Relative path

Definition

A relative path is a path that is relative to your current working directory. It does not start with `/` Can also think of it as directions from where I am right now.

Example

- Creating a file using a relative path
  - If your current directory is `/home/student`:
    - `touch final_exam_study_notes/relative_file.txt
- `cd final_exam_study_notes`
  - `ls`
    - This changes into the directory and lists its content using relative paths

---

How to work with the manual pages (man command)

man

Definition

`man` display the manual (help documentation) for a command. Manual pages explain what a command does, its options and how to use it.

- ### Syntax

```
man command
```

- ### Example
  - Open the manual for `grep`
    - `man grep`

Navigating the manual page While view a manual page, you can use the following keys:

- `space` - Move to the next page
- `b` - Go back one page.
- `/word` - Seatch for a word
- `n` - Go the next match

- `q`- to quit the manual page

How to parse(search) for specific words in the manual page

Definition

Parsing (searching) a manual page means finding specific words or options inside the output of the `man` command. This helps you quickly locate information in long manual pages.

Method 1: Search inside the man page

1. Open the terminal
2. In the terminal, enter the `man + command`
    1. `man grep`
3. Press the `/` followed by the owrd you want to search for
    1. `/recursive`
4. Press enter to find the first match.
5. Use:
    1. `n` - go the next match
    2. `N` - go the previous match
    3. `q` - quit the manual page

Method 2: Parse the Manual Page Using Pipes

Can also send the manual page output to `grep` using a pipe `(|)`.

- Example: Search for the word `"recursive"` in the `grep` manual page
    - `man grep | grep -i "recursive`
        - This command displays only the lines in the manual page that contain the word `recursive`,ignoring case

How to redirect output (>, >>, and |)

Output redirection allows you to control where the result of a command goes. By default, commands display output on the screen, but redirection allows you to send that output to a file or to another command.

Redirect Output to a File (`>`)

How to append the output of a command to a file

How and when to redirect the output of a command to another (pipes)

How to use echo and output redirection to create a new file that contain some text

How to to use braxe expansion (For creating directory strucures in single command)

How to create simple "hello world" shell script

How to use variables in a shell script

---

awk

- Definition
  - `awk` is a text-processing scripting language. It reads input line by line and is mainly used to extract and print columns (fields) from files.
  - Can think of `awk` as "Take this file and show me specific columns"
- Formula:
  - `awk + options + {awk command} + file + file to save (optional)`
- Example
  - Print the first column of every line
    - `awk '{print $1}' ~/Documents/Csv/car.csv`
  - Print the first field of /etc/passwd
    - `awk -F:'{print $1}' /etc/passwd`
  - Print the last field
    - `awk -F: '{print $NF}' /etc/passwd`
  - Print first and last field
    - `awk -F: '{print $1, "=" , $NF}' /etc/passwd`

AWK variables

`$0` - Entire line `NR` - Line number `NF` - Number of Fields `FS` - Input field separator `OFS`- Output record separator `FILENAME` - Name of the file `IGNORECASE` - Ignore case

---

cat

- Usage
  - The **cat** command is used for displaying the content of a file.
- Formula
  - `cat + option + file(s) to display`
- Example
  - Display the content of a file located in ~/Documents/sample_files/
    - `cat ~/Documents/sample_files/Code/helloWorld.py`

- Display the content of a file with line numbers.
  - `cat -n ~/Documents/sample_files/Code/helloWorld.py`
- Display the content of a file including non printing characters and line endings
  - `cat -A ~/Documents/sample_files/Code/helloWorld.py`

---

cp

Definition: **cp** is used to **copy files or directories** from one place to another.

- It takes something from a **source**(the orignal location) and duplicates it in a **destination**(the new location).

Usage/Formula

- **copy a file:**
  - `cp [file_to_copy] [destination]`
- **Copy multiple files:**
  - `cp [file1] [file2] [destination_folder]`
- **Copy a directory (folder):**
  - `cp -r [directory_to_copy][destination]`

*The `-r` options means **recursive**, which is needed when copying folders because it includes all their contents (subfolders and files)*

Example

1. **Copying a single file to another folder:**
   - `cp Downloads/wallpapers.zip Pictures/`
2. **Copy multiple files into one folder:**
   - `cp list.txt notes.txt backup/`
3. **Copy a folder and everything inside it**
   - `cp -r wallpapers Pictures/`

cut

- Usage

The **cut** command is used to extract a specific section of each line of a file and display it to the screen.

- Formula

    - `cut + option + file(s)`

- Example

    - Display a list of all the users in yours system
        - `cut -d ':' -f1 /etc/passwd`
    - Display a list of all the users in your system with their login shell
        - `cut -d ':' -f1,7 /etc/passwd`

Remember: cut grabs columns

Fields start at 1 Must know what separates the fields(delimiter)

Options - `-d` - delimiter (what separates fields) `-f` - field number(s) `--output-delimiter` - change separator in output

More examples:

`cut -d': -f1 /etc/passwd` usersname only

`cut -d':' -f1,7 /etc/passwd` username and login shell

`cut -d' ' -f3,4 file.txt` fields 3 and 4 using space as delimiter

`cut -d':' -f1,7 --output-delimiter=' -> ' /etc/passwd` change how output looks

grep

- Definition

    - Grep is used to search text inside a file. It works line by line, meaning it checks each line separately and prints only the lines that match the search criteria

- Formula

    - `grep + option + 'search crieria' + files`
        - search criteria - the word or pattern you looking for
        - file(s) one or more files to search

- Example

    - Search for any line that contains the word `"dracula"`
        - `grep 'dracula' ~/Documents/dracula.txt`
    - Search for dracula regardless of uppercase or lowercase letters
        - `grep -i 'dracula' ~/Documents/Books/dracula.txt`
    - Search for "dracula" ignoring case and show line numbers
        - `grep -in 'dracula' ~/Documents/Books/dracula.txt`
    - Show all line that do not contain the word "war"
        - `grep -v 'war' ~/Documents/Books/war-and-peace.txt`

- Display only the matched word "pride"

    - `grep -o 'pride' ~/Documents/Books/war-and-peace.txt`
- Show information about the current user
    - `grep -i $USER /etc/passwd`

Common option `grep` `-i` = ignore case (uppercase/lowercase does not matter) `-n` = show line numbers for matching lines

`-E` = Use extended regular expressions

`-G` = Use basic regular expressions

`-v` = Invert search (show lines that do not match)

`-o` = Show only the matched part of the line

`-c` = Count how many times a match occurs

`-w` = Match the whole word only

`r`, `-R` = Search recursively in directories.

head

- Usage

The **head** command displays the top N number of lines of given file. By default, it prints the first 10 lines. If more than one file name is provided then data from each file is preceded by its file name

- Formula

`head + option + file(s)`

- Example
    - Display the first 10 lines of a file
        - `head ~/Documents/Book/dracula.txt`
    - Display the first 5 lines of a file
        - `head -5 ~/Documents/Book/dracula.txt`

LS

Definition

- The `ls` command is use when you want to **see what's inside a folder**.
- It shows all the files and folders that are in your current location.
- You can think of it like **opening a folder on your computer** to look inside but using the terminal instead of clicking.

Usage/Formula

- `ls[options][directory_to_list]`

Example

- Typing `ls` and pressing enter - shows everything in the current folder.
- Typing `ls -a` shows all files, even the hidden ones (tLike the ones that start with a dot like .blank.txt)
- Typing `ls -l` - shows a detailed view, including file sizes, permission, and dates.

`ls` mean **List**

---

mkdir

Definition - **mkdir** command is used for creating single directory (folder) or multiple directories (folders) in the terminal.

Usage/Formula: **Creating a folder: `mkdir + the name of the directory`**

- Creating a directory in the working directory: `mkdir wallpapers`
- Creating a folder in a different folder using a relative path - `mkdir wallpapers/ocean`
- Creating a folder in a different folder using absolute path - `mkdir ~/wallpapers/forest`
- Creating a directory with a space
  - 1. `mkdir wallpapers/new\ cars` OR
  - 2. `mkdir wallpaper/'cities usa'`
- Create multiple directories
  - `mkdir wallpapers/cars /wallpapers/cities wallpapers/forest`
- Create a directory with a parent directory at the same time
  - `mkdir -p wallpapers_other/movies`

Example: Step One: **Create a folder in your current working directory:** `mkdir wallpapers`

Step Two: **Create a folder inside another folder using a relative path:** `mkdir wallpapers/ocean`

Step Three: **Create a folder using an absolute path:** `mkdir ~/wallpapers/forest`

Step Four: `mkdir wallpapers/new\ cars` (Backslash before space)

```
      OR
```

`mkdir wallpapers/'cities usa`

Step 5. Create multiple folders at once

`mkdir wallpapers/cars wallpapers/cities wallpapers/forest`

- This creates three folders(cars, cities, and forest) inside wallpapers folder

mv

Definition: **mv** is used to **move or rename files and directories**.

- It can move items from one location to another or give them a new name.

Usage/Formula **Move a file or folder:** `mv [source] [destination]`

Rename a file or folder: `mv [old_name] [new_name]`

*Both **Source** and **destination** can be wrtitten as **absolute path** (the full address starting from `/`) or **relative paths** (based on your current folder)*

Example

1. **Move a file from one folder to another (relative path):** `mv Downloads/homework.pdf Documents/`

2. **Move a directory using absolute path:** `sudo mv ~/Downloads/theme /usr/share/themes`

3. **Move a file using both absolute and relative path:**

    - `mv Downloads/english_homeowrk.doc /media/student/flashdrive/`
        - moves the "english_homwork.doc" file from downloads folder to flash drive.

4. **Move multiple files or folders at once:**

- `mv games/ wallpapers/ rockmusic/ /media/student/flashdrive/`
    - Moves all three folders "games", "wallpapers", and "rockmusic" to flashdrive

Renaming Examples

1. Renaming a file: `mv homework.docx cis106homework.docx`

2. **Rename a file using absolute path:**

- `mv ~/Downloads/homework.docx ~/Downloads/cis106homework.docx`

3. **Move and rename at the same time**

`mv Downloads/cis106homework.docx Documents/new_cis106homework.docx`

---

tac

- Usage

The **tac** command is used for displaying the content of a file in reverse order.

Just like cat, tac concatenates files and displays the output of the concatenates

- Formula

```
tac + option + files(s) to display
```

- Example
  - Display the content of a file located in
    - `tac ~/Documents/sample_files/Code/helloworld.py`
  - Display the content of multiple files in reverse order
    - `tac ~/Documents/sample_files/Code/helloWorld.py ~/Documents/sample_files/Code/helloWorld.sh`

---

tail

- Usage

The **tail** command displays the last N number of lines of a given file. By default, it prints the last 10 lines.If more than one file name is provided then data from each file is preceded by its file name.

- Formula
  - `tail + option + file(s)`
- Example
  - Display the last 10 lines of a file
    - `tail ~/Documents/sample_files/`
  - Display the last 5 line of a file
    - `tail -5 ~/Documents/sample_files/`
  - Display the first 5 lines of multiple files
    - `tail -n 5 Txt/{dracula,war-and-peace}.txt`
  - Display the first line of multiple files using wildcards
    - `tail -n 1 Csv/*.csv Code/*.py`

---

touch

Definition: touch command is used for creating files.

- Its can make one or serveral empty files at once.
- If the file already exists,

Usage/Formula:

- **Creating a file:**
  - `touch[file_name]`
- **Creating multiple files:**
  - `touch [file1] [file2] [file3]`

Examples:

1. **Creating a file called list:**

   - `touch list`

2. **Create several files at once**

   - `touch list_of_cars.txt script.py names.csv`

3. **Create a file using an absolute path:**

   - `touch ~/Downloads/games.txt`

4. **Create a file using a relative path (assuming your home directory is your current location)**

   - `touch Downloads/games2.txt`

5. **Create a file with a space in its name:**

   - `touch "list of foods.txt"`

Finish later