Hack Astro Academy

Óscar A. Chávez Ortiz and Owen Chase

Class Objectives

1. **Recap Terminal Commands**
2. **Introduce Terminal Based Text Editors(vi/vim)**
3. **Outline Programmatic Thinking**

Recap: Terminal Commands

1. **What command make you switch folders/directories?**
2. **How do I go back two folders/directories?**
3. **What command displays the contents of a file?**
4. **What command displays what is in the folder you're currently in?**
5. **What command tells you where you are currently at?**
6. **What is the difference between relative and absolute paths?**

Recap: Terminal Commands

1. **cd**
2. **cd ../../ or cd ../..**
3. **cat filename.txt or head/tail filename.txt**
4. **ls, ls -a**
5. **pwd**
6. **Absolute path contains the full directory tree to a file/folder and always points to the same file. Relative path uses ../ and depending on where you are at can point to different files.**

Recap: Terminal Commands

ls:
list the content of the directory

cd:
moves you from one directory to another

mkdir:
Makes a directory

pwd:
shows the present working directory
(i.e. The directory you are currently at)

Recap: Terminal Commands

touch: makes a file and/or updates time of last used

cat: opens a file

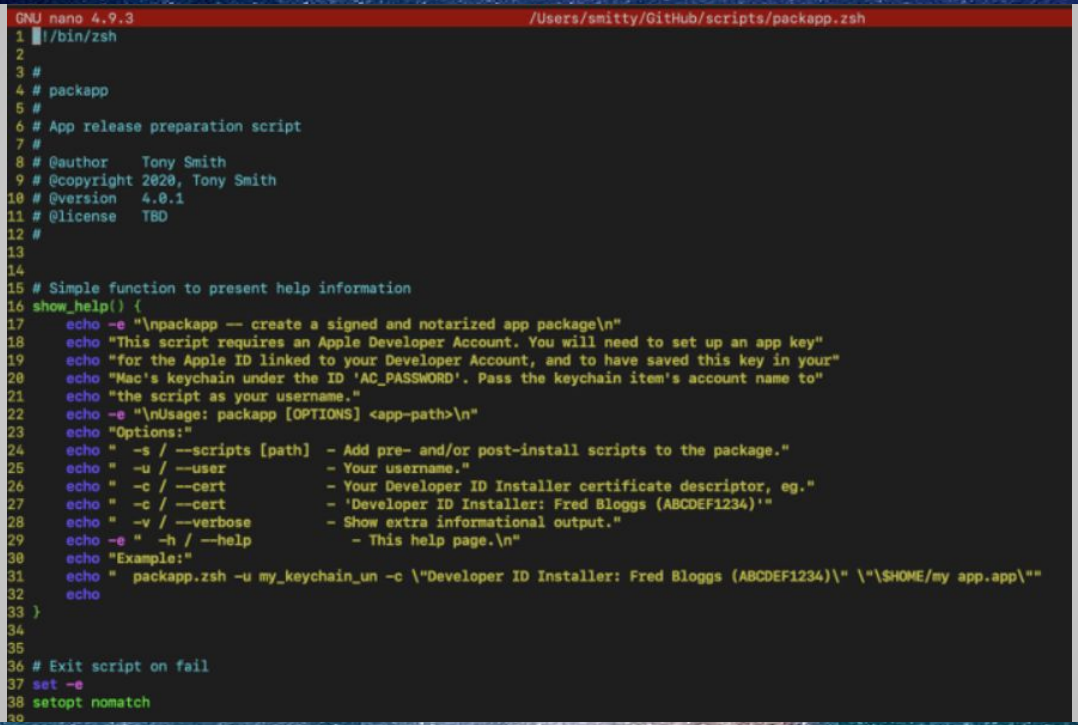
head/tail: shows the first/last 5 lines in a file

rm/rmdir: remove file/remove directory*

Terminal Commands

man: short for manual, lists the instruction for how to use the command that follows it. I.e. man ls will list the manual for the ls command

Editing Files Through Terminal



Introduction to vim/vi

What is Vim/Vi?

A powerful text editor commonly used in Unix/Linux environments.

Vim (Vi IMproved) extends the original Vi with additional features.

Key Features:

- Lightweight and efficient.
- Designed for editing text directly in the terminal.
- Supports syntax highlighting, search, and macros.
- Highly customizable and scriptable.

Introduction to vim/vi

Modes of Operation:

1. **Normal Mode:** Navigation and text manipulation.
2. **Insert Mode:** Text entry and editing.
3. **Command Mode:** Executes commands (e.g., save, exit).
4. **Visual Mode:** Selects and manipulates text.

Introduction to vim/vi

Basic Commands:

- Enter **Insert Mode**: Press i or a.
- Exit Insert Mode: Press Esc.
- Save and Exit: :wq.
- Quit without Saving: :q!.

Note: Mouse clicks do NOT work on terminal, navigating the text through vim is done through the arrows on your keyboard or through vi/vim shortcuts

Introduction to vim/vi

Useful Shortcuts:

- **W:** move to beginning of next word after a whitespace
- **B:** move to the beginning of the previous word before a white space
- **0:** move to the beginning of the line
- **$:** move to the end of the line

More vi/vim shortcuts can be found here:
https://gist.github.com/glnds/11027696

Thinking like a Programmer

Demonstration of vi

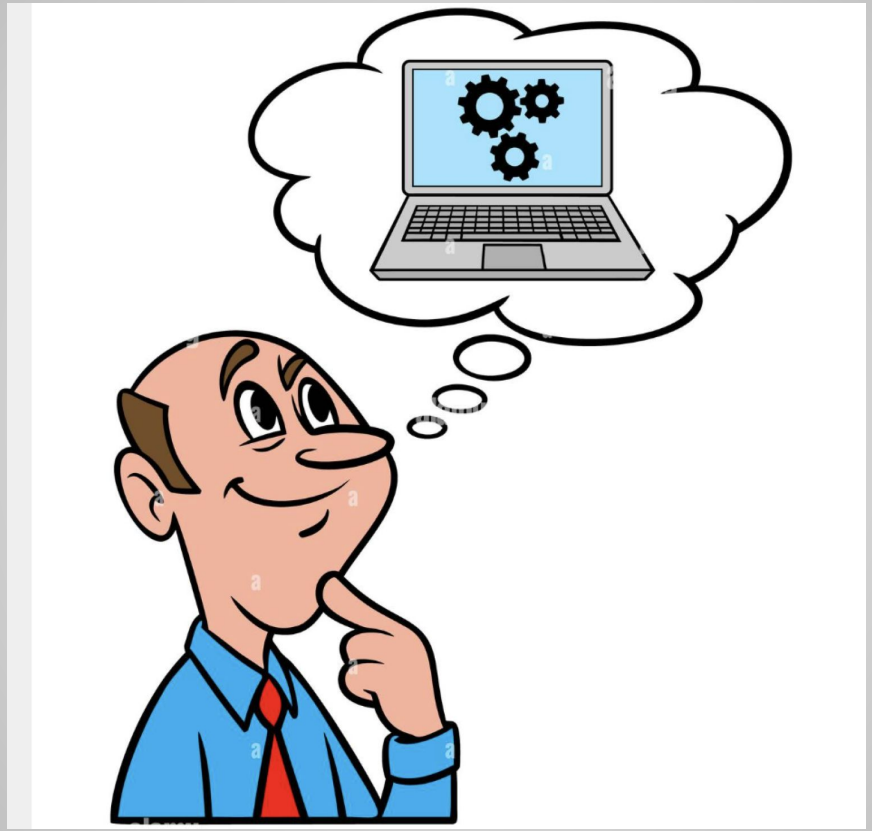
Your Turn (~5 minutes)

In the Week_02 folder there is a file labeled:

text_editor_assignment.txt

Read the contents of the file and perform the actions outlined in the file

Thinking like a Programmer



Overview on How Computers Work

- Question:
  - In the previous lecture we covered terminal commands. How did the computer know what to do when we gave it certain commands like ls, cd, cat?

Overview on how Computers Work

- The computer knew what these commands are because someone has coded up what a computer should do when it sees those commands.
- These commands are a set of instructions that get executed when the correct input is given. These set of instructions are what we call as algorithms

Introducing Algorithms

What is an Algorithm?

An algorithm is a step-by-step, ordered list of instructions designed to solve a specific problem or perform a task.

Key Characteristics:

1. **Clear and Unambiguous:** Every step is precise and easily understood.
2. **Well-Defined Inputs and Outputs:** Starts with given inputs and ends with specific outputs.
3. **Finiteness:** Completes after a finite number of steps.
4. **Effectiveness:** Each step can be performed in a reasonable amount of time

Why Learn Algorithms?

- They form the foundation of programming and problem-solving.
- Efficient algorithms save time and resources in software development.

An Analogy of Algorithm: Recipes

1. Grab some flour, eggs, and milk.
2. Mix everything together.
3. Heat up a pan and pour some batter in.
4. Flip it when it looks right.
5. Cook until done and eat.

An Analogy of Algorithm: Recipes

Ingredients:

- 1 cup all-purpose flour
- 1 tablespoon sugar
- 1 teaspoon baking powder
- 1/2 teaspoon baking soda
- 1/4 teaspoon salt
- 3/4 cup milk
- 1 egg
- 2 tablespoons melted butter or oil

Instructions:

1. **Prepare the Dry Ingredients**:
   In a large mixing bowl, combine 1 cup of flour, 1 tablespoon of sugar, 1 teaspoon of baking powder, 1/2 teaspoon of baking soda, and 1/4 teaspoon of salt. Stir to mix evenly.
2. **Prepare the Wet Ingredients**:
   In a separate bowl, whisk together 3/4 cup of milk, 1 egg, and 2 tablespoons of melted butter (or oil).
3. **Combine Ingredients**:
   Slowly pour the wet ingredients into the bowl with the dry ingredients. Stir gently until just combined. Small lumps in the batter are okay—don't overmix.
4. **Heat the Pan**: Place a non-stick pan or griddle over medium heat. Lightly grease the surface with a small amount of butter or oil.
5. **Cook the Pancakes**:
   Pour 1/4 cup of batter onto the pan for each pancake. Cook for 2-3 minutes, or until bubbles form on the surface and the edges look set.
6. **Flip and Finish Cooking**: Use a spatula to flip the pancake. Cook for an additional 1-2 minutes until the underside is golden brown.
7. **Serve**: Remove from the pan and serve warm with your favorite toppings like syrup, fruit, or whipped cream.

Exercise: (~5-10 minutes)

- Read the instructions provided in the **pbj_assignment.txt** file and use vi to write your instructions on making PB&J to the file named *pbj_instructions.txt*

Final Thoughts

- Computers are powerful machines and work by following algorithms
- To make a computer execute a task the algorithm needs to be exact, and clear
- One thing to keep in mind as we start building our own algorithms is that computers only do what we tell it to do.
    - If something did not execute as we expect we need to revisit the steps we provided the computer and see what was unclear