Introduction to Linux*

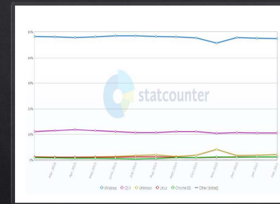
*And some Git, too!

Why Linux?

- First off… We actually call it *NIX (as in, UNIX-Like). But, for all intents and purposes, "Linux" will do.
  - The "Why" is complicated, and like many things, wrapped in legalities.
- When you think of a "computer" what comes to mind?
  - There's a lot of computers out there. Not all are intended to do things that we traditionally use them for.
    - Client / Server model
    - GUI

Look Familiar?

- Is it exactly this? Probably not.
  - But it is probably very close.
- It's extremely misleading
- It's a very small slice of "the pie"
- Credit: https://gs.statcounter.com/os-market-share/desktop/worldwide



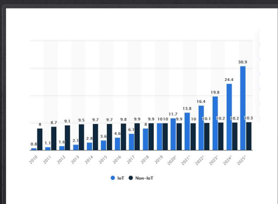
What's a "Computer" again?

- No, not the commercial from apple
  - Although, it was very on point.
  - It was also kind of universally hated.
    - Seriously… They took it off of YouTube
- How many "things" in your house do you think have a processor?
  - Do the terms "IoT" or "Smart" mean anything to you?

Embedded Systems

- You've probably heard of the "Internet of Things" or IoT
  - It's more of a concept than a specific kind of device.
    - Think: A giant web of connected devices that send data between each other
    - Observe: Fancy graph!

Credit: https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/



Embedded Systems

- Remember when we said the Desktop graph was a little misleading?
- I have some bad news
  - IoT graph is a little misleading, too.
  - IoT is primarily made of "Embedded Systems"

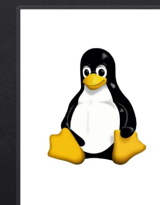
Okay…

- So what exactly are "Embedded Systems," then?
  - It's a type of computer system!
    - Has a processor
    - Has memory
    - Handles Input/Output (IO)
    - Usually does a *thing* or specific *things*
- Hopefully you are starting to think of a lot of things that fit that description
  - Because it is a lot of things..
    - Spoiler: It's most things.

THAT is why we are talking about Linux*

- Some 98% of computing devices are embedded systems. That's a lot.
- And what do these systems run? Is it Windows?
  - Spoiler: NO
  - They run good, old, trusty, Linux*

*And by Linux, we mean: *NIX or UNIX-Like



Therefore

- You need to be prepared and comfortable using the Linux environment
  - Headless Mode
    - Command Line Interface (CLI)
  - Graphical User Interface (GUI)
    - More like what you're used to.
      - Think: macOS (gasp), Ubuntu, PopOS, etc.
- You must be able to configure these systems. They run the world.
  - Websites, e-mail, social media, you name it.
  - If you write software: Version Control
    - More on this in a bit.

The Basics

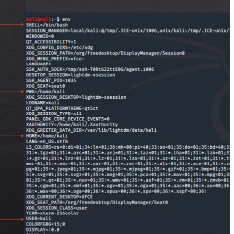
- The Command Line Interface (CLI)
  - Super-hacky, type stuff and do things.
    - In some cases, it does feel better. In other cases, the GUI was a much-needed change.
- Think back to last time: You type commands and things happen.
  - But how do you find what you can do?
  - How do you add new commands?
  - What's happening in the background when you do these things?
- There's a lot of circular problems here. You can dig as deep as you'd like.
  - We don't need to over-complicate this

CLI

- The CLI that you use provides you a shell.
  - There are many kinds of shells that people use.
  - Most common: BASH, sh, zsh
    - /bin/bash, /bin/sh, /bin/zsh
- A shell is just an interface to a *NIX system.
    - It takes input, executes programs, provides output.
    - Even Windows them, but I won't bore you with it! (PowerShell, DOS, CMD)
- Your shell maintains what's available to you.
- It keeps track of where you are in the File System.
  - It does lots of other things, too.

Let's take a look

- env
  - Displays all of your environment variables
  - Observe the SHELL
  - Observe the working directory (PWD)
  - Observe the HOME directory
  - Observe the USER
- Your shell must keep track of all kinds of things.



Permissions

- Let's keep it simple.
  - Just like Windows:
    - There are things your user account can do
    - There are things your user account cannot do
      - In order to do those things, you need to have more privileges
- The highest level of privileges in Linux is the super user, "root", user.
- If your user has permission to do administrative things, you must elevate your privileges
  - This is done via sudo (super user do)
    - sudo <command>
    - Literally, super user do <command>

An Illustration

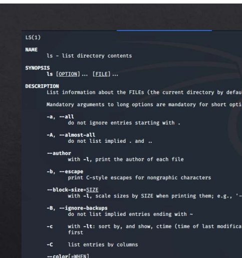
- Lots of new commands here:
  - Whoami
  - Id
  - Groups
  - Sudo
  - Exit
- Notice the changes in information between users

Commands

- Hey, those were a lot of commands!
  - Yep! We used them last time, too.
  - A lot of the time, their name implies what they do. Other times it isn't as clear right away.
- Scenario: You need to find out all of the files in your working folder (directory)
  - Remember, folders and directories are the same thing. Don't let the words confuse you.
- The great Google has informed you that the 'ls' command is the way to go.
  - How do we find out what it does?

Manual Pages (Man Pages)

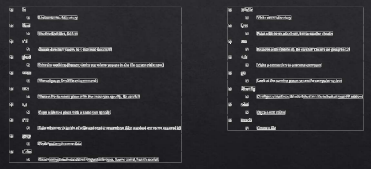
- On our system, if we type 'man ls' without quotes
- we are greeted with a manual page entry for the ls command.
- It provides an overview for the command, and other options you may provide to do other things with it.
- You can follow this process for nearly any command you come across in Linux
- An alternative is –h or –help e.g., ls –help



Scenario Continued

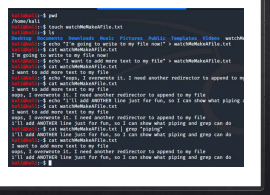
- What if that command didn't exist on your system?
- All Linux distributions keep track of all the packages installed on the system.
- You can install the package.
- The process to do this depends on your Package Manager
  - It also depends on your Operating System
  - It also usually requires elevated privileges
    - Therefore, no package installation in our Jupyter lab
      - Everything you need should be there

What are some common commands?



Input and Output

- Frequently, you'll need to use a command and keep the output somewhere.
  - Some commands can write directly to a file
  - Most of the time, you need to utilize 'redirection'
- Redirection operators: >,<, >>, <<, and the Pipe: |
- Let's walk through it



Introduction to Git

- What's Git?
  - Git is a way to track changes on a file.
  - Frequently used for "Source Control"
    - Software Development term
- What if you have 10, 100, 1000 programmers working on one project at once?
  - Need to be able to track who has done what
  - Need a way to deal with conflicts
  - Need a way to be able to collaborate separately, but maintain software in one place.

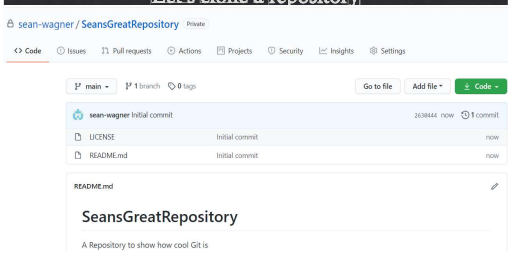
You may have heard of it

- It's used in a variety of popular products. The most common are Gitlab and Github.
  - Gitlab – Owned by Atlassian
  - Github – Now owned by Microsoft
- Git is primarily a command line utility
  - So, that's how we are going to practice it.

Common git commands

- git pull
  - Pulls all of the latest changes to the repository you are working in.
- git commit
  - Captures a snapshot of the current state of the project and the staged changes.
- git push
  - Send the files that you've added and changed and push them up to the branch that you're working on. Others can pull the changes down.
- git rm
  - Remove files from the repository (no longer tracked)
- git status
  - Shows the status of the repository.
- git clone
  - Grab a repository online and clone it locally to your system so you can build it or work on it.
- git add
  - Add your local files to the branch that you are working on
- git stash
  - Make a snapshot of your repository and store it away so you can go back to a clean repository.

Let's clone a repository

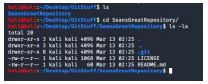


git clone

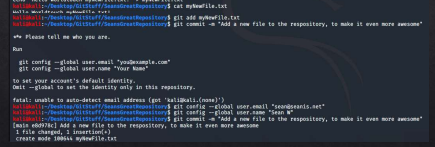
- Lets talk through it
  - We need to git clone <link to repo>
  - Asks us for credentials
  - You won't see the password as you type it

Looking around

- An ls command shows a new directory where we are, the repo!
  - Let's cd into it and see what's in there!
  - ls command with the -la options shows all files with more information
- There's my README file and my license that I specified for my repo!

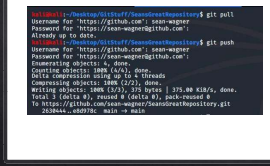


Let's add a file

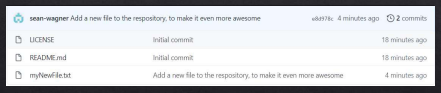


Let's push our changes up!

- Always do a pull before you push!
  - Someone else may be working, too.
  - It avoids massive headaches in the future, to make the good habit.
- Now, let's git push and send our new file up with what we put in it



Let's double check it worked.



In Class Exercise

- Use your github account and do the GitHub Hello World Exercise
  - https://guides.github.com/activities/hello-world/
- This will:
  - Allow you to create and use a repository
  - Start and manage a new branch
  - Make changes to a file and push them to GitHub as commits
  - Open and merge a pull request
- Bonus resource:
  - GitHub primer from Dr. Matt Hale from University of Nebraska Omaha