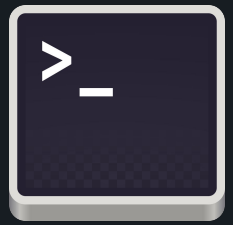


Command Line Fundamentals

Day 1

Follow Along At sipb.mit.edu/iap/2022/clf/

Motivation

Motivation (cont.)

Supercomputing:

- Typical laptop: 10^{11} floating point operations/sec
- Aurora supercomputer: 10^{18} operations/sec – 10 million times faster!
- High-performance computing needed in many applications (academia and industry)
- Need a way to connect to these servers

General:

- Future classes/careers
- Automate repetitive operations

SSH

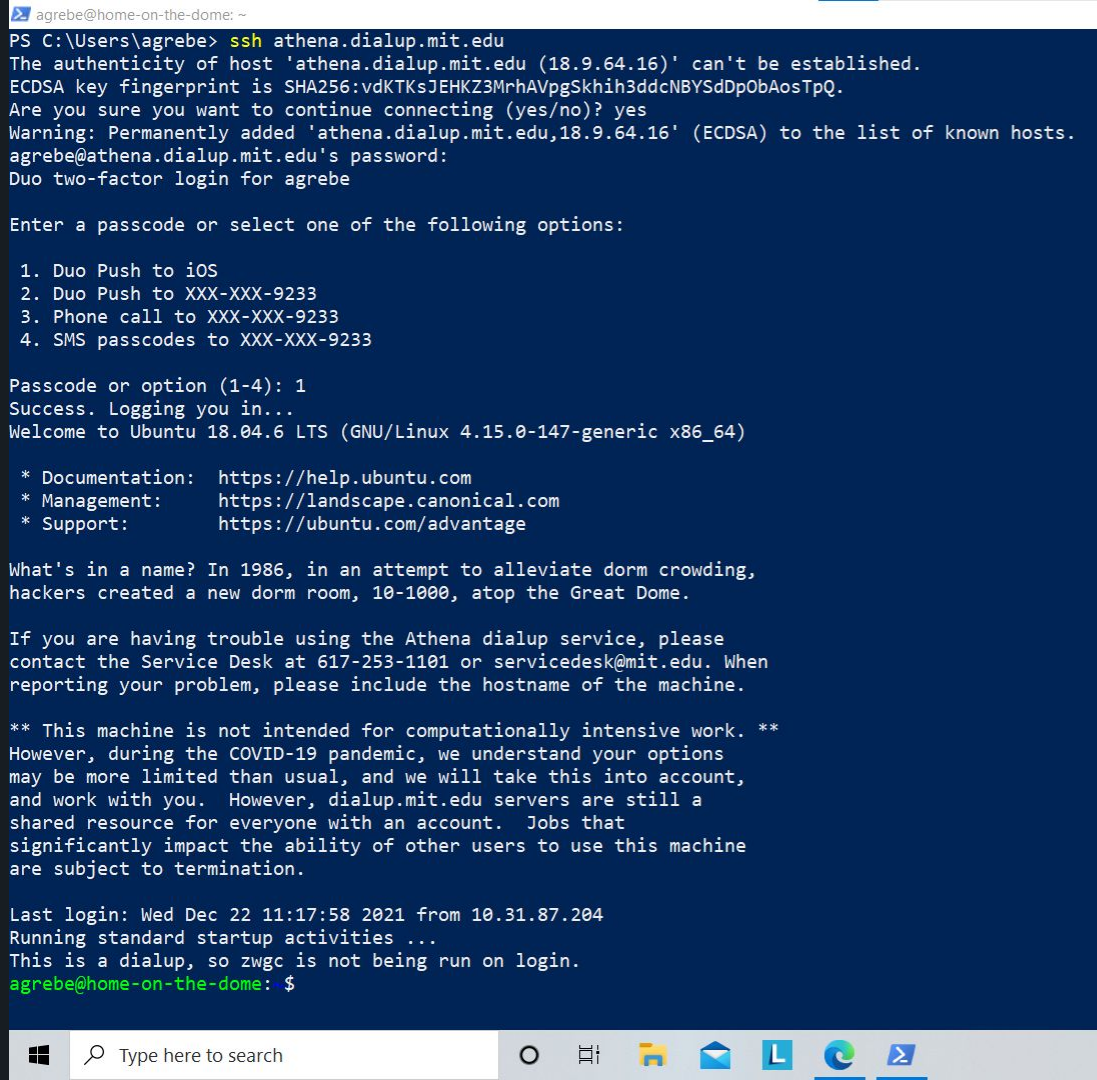
- Secure Shell – allows secure connection between local machine and remote server
- Standard method of connection for non-Windows servers (almost all servers)
  - All top 500 supercomputers are Linux based (including Microsoft Azure!)
- (Almost) entirely command-line based (unlike Windows Remote Desktop)
- Client support on most major operating systems (Windows 10, MacOS, Linux)
- MIT provides access to Athena dialup servers (athena.dialup.mit.edu)
  - Athena access also available online (https://athena.dialup.mit.edu)

SSH (cont.)

- Syntax: ssh *username@server.name*
  - Ex: ssh tim@athena.dialup.mit.edu
  - Note: If user account on local machine matches that of server, username can be omitted
    - Ie: ssh athena.dialup.mit.edu
- First connection: Will ask for confirmation to accept server's fingerprint
  - This is saved to make sure server does not change (due to spoofing attack)
  - Safe practice to double check
- Type password at prompt and complete Two Factor Authentication (2FA) (if required)
  - Some servers allow/require SSH keys instead of password (essentially very long password stored on local machine)

Windows PowerShell

- Available on Windows 10
- Search for “PowerShell”
- Supports SSH and some basic commands (but different syntax from MacOS/Linux)
- Note: Windows 7 requires third-party tools
  - PuTTY
  - Chrome browser plugins



Linux/MacOS Terminal

- Both of these OSs are based off Unix, hence similar syntax
- SSH process very similar to Windows
- You can run most if not all commands natively (on your machine) without needing to connect to Athena
  - A couple small things to note, e.g. sed (covered tomorrow) is different on MacOS than Linux
  - This course will focus on Linux

Terminal Navigation

- After logging in/opening the Terminal, you're placed in your home directory
  - "directory" = folder
  - Stores most fies of interest to user, as well as sub-folders (e.g. Desktop, Documents, etc.)

- Two most important commands: ls and cd
- ls: list contents (files + sub-directories) within current directory
- cd: change directory
  - Allows movement between folders

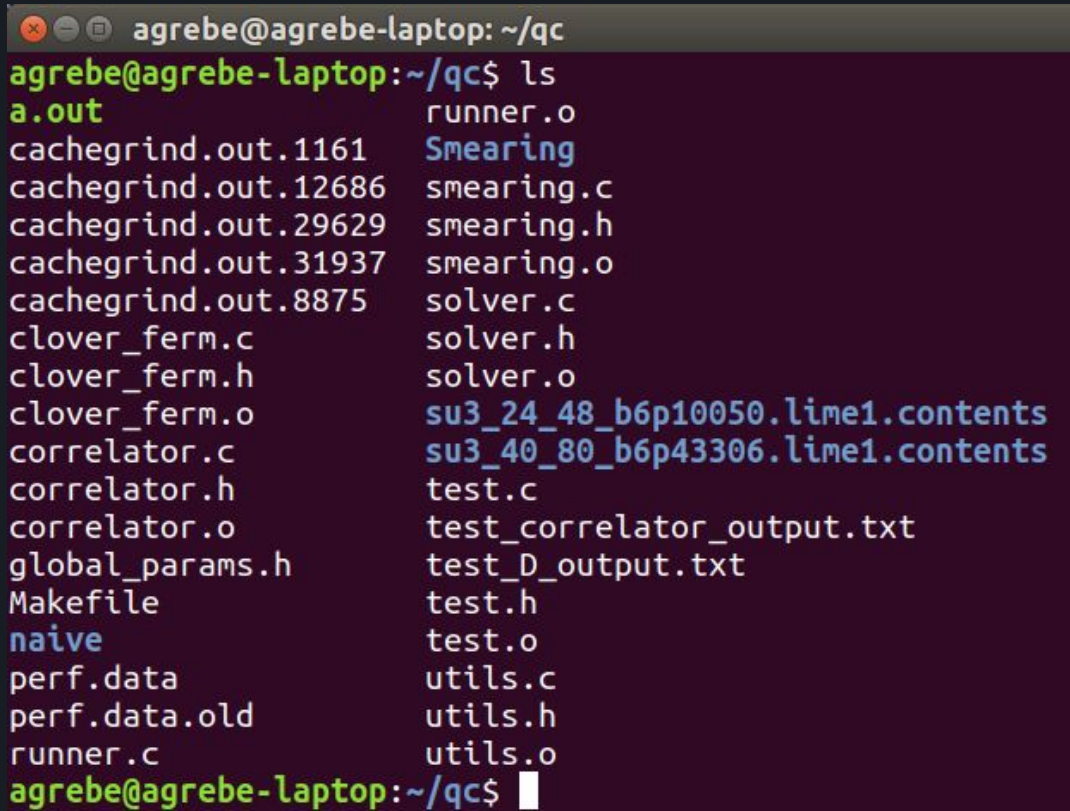
- Can be intimidating at first!
  - Takes some practice to become comfortable

Color Scheme for ls

- Ordinary file: white
- Directory: blue
- Executable file: green
- Other colors for other unique files

Note: Coloring may be different on different OSs or there may not be any coloring at all

*Default file coloring for Ubuntu 18



File Paths for Navigation

- cd and (optionally) ls take directory or file path as argument
- Special file paths:
  - . – current directory
  - .. – parent directory (one folder level up)
  - ~ – home directory (where we started)
  - / – top-level directory (contains everything on disk, including system files)
- Paths are relative to current directory, unless starting with / or ~
- Can have paths multiple directories deep
  - cd *dir/subdir/subsubdir/*
  - Absolute paths (starting with /) typically quite deep
    - Eg: /afs/athena.mit.edu/user/a/g/agrebe/Public/intro-terminal
- You can reveal the absolute path to your current folder location by using pwd

Flags

Most commands supports additional parameters in the form of *flags* that help extend the capability of the command

Ex:

- ls -l: List all files/directories with extra information such as who can edit it, how big it is, etc.
  - Useful on systems without color – directories identifiable by letter “d” in beginning of each line
- ls -a: List all files/directories, including hidden files
  - Hidden files start with a dot. Eg: .config

Copying Files

- cp used for copying
- Syntax: cp *source destination*
  - *source* is file to be copied
  - *destination* is either new file name or directory to put new file
  - Can copy multiple files to a new directory: cp *source1 source2 … sourceN destination*
- DANGER: cp overwrites existing files **without warning**
  - *destination* could be important file (with same name as source)
  - *destination* could be directory with important file (with same name as source)
- cp -r copies directory with all its contents
  - -r flag means recursive (since all sub-directories copied)

Moving Files

- mv used to move files (without keeping old version)
- Same syntax as cp: mv *source destination*
  - Also works with multiple sources
- DANGER: As with cp, overwrites files with the same name **without warning**
- Destination can be file or folder
  - *destination* can be used as new file name to rename current file

Creating Directories and Files

- mkdir creates new directory
  - Syntax: mkdir *directory*
  - Can create multiple directories at once (mkdir *dir1 dir2 … dirN*)
  - To make nested directories use -p flag as so: mkdir -p dir/subdir/…

- touch creates a new empty file
  - Syntax: touch *file*
  - Less common – usually just create files with content
  - If you touch an existing file nothing bad happens (its “modified” timestamp is updated)

Deleting Files

- rm permanently deletes files from system
  - Syntax: rm *file1 file2 … fileN*
- DANGER: Cannot be reversed (**rm is forever**; NOT sent to trash/recycling)
  - Could potentially be restored by professional services (and therefore not “secure erase”) but cannot be done within standard Linux OS
- rm -r removes entire directories

Interesting Find:

- sudo rm -rf / deletes entire system (including OS) and all data
  - sudo: run as root (equivalent of Windows administrator)
  - -f flag: force (will delete everything, including system files, without prompting)
  - /: top-level directory (contains everything in system as sub-directory)
- With great power comes great responsibility!

Viewing Files

- less is used to view the contents of a file on a scrollable text interface
  - Syntax: less *file*
  - Use q to exit the file, j/↓ and k/↑ to scroll down and up respectably
  - Best used for large files

- cat is used to print the contents of a file onto the terminal
  - Syntax: cat *file*
  - Best used for small files

Vi/Vim Basics

- A powerful text editor that can be found in virtually all Unix-based PCs/servers. Saves you time over other text editors/methods, once you get over its learning curve.

- File Creation: vim *file*
  - Ex: vim example.txt (Don't forget the file extension in the file name)

- Modes: INSERT, NORMAL, VISUAL, COMMAND

- Command Mode
  - Saving and closing
    - :w - save
    - :wq - save and exit file
    - :q! - exit without saving

- Escaped Mode
  - Cursor movement:

k/↑
h/← l/→
j/↓

More Vim

- Escaped Mode (cont.)
  - w - Jump to the start of the next word, where punctuation also separates words
    - Ex: Hello. World.
      W→ Hello. World.
  - W to move cursor to the start of the next word, where spaces separate words
    - Ex: Hello. World.
      w→ Hello. World.
  - b - Jump backwards to the start of next word (and punctuation)
  - B - Jump backwards to the start of next word

- Insert Mode
  - Add text to the file
  - Use i to insert into the character after your cursor
  - Use a to insert into the character before your cursor

- Note
  - All modes can be escaped with the ESC key (returns you to NORMAL mode)
  - Careful, there's a difference between lowercase and uppercase keystrokes!

Even More Vim

- **Visual Mode**
  - Press v and move cursor around to select a chunk of text. You can then apply the copy, paste, or delete keys to it:
    - y to copy (yank)
    - p to paste (put)
    - x to delete
- **Careful**
  - Sometimes a file might require root permission to edit it. When that happens, prepend vim with sudo:
    - sudo vim *file*

In your free time, consider going through Vim's own tutorial (30 min). You can do this by typing vimtutor in the terminal.

Running a program

To run certain programs on Linux, you may need to execute a Bash script file (.sh) or an ELF binary (Linux equivalent of .exe). But to do so, must mark a file as executable:

chmod +x *file*

If your terminal supports colors, you file will show up as colored (green on Ubuntu) on ls.

To actually run, type:

./file or sh *file* (*)

*There's a subtle difference with these, where ./ runs based off the shebang line and sh is for bash scripts only

What about Python files?

Nowadays, Python3 comes preinstalled on MacOS and most Linux distros. To start up the Python command line interface, simply type:

```
python3
```

And to exit:

```
exit()
```

To run a python script:

```
python3 script_name.py
```

Note:

- If you are doing any file / directory navigation, remember that the program will use the directory you *currently are in*
- Use the --version flag to verify which Python version you have installed

A Few Extras

- Use history to show all commands you have run on your current shell session.

- Use clear to declutter your screen. Note, this does not affect your history.

Getting help

There are many great online resources that you can use for help to get help with the terminal. Don’t be afraid to use them!

- If you are working with a special command or tool, most likely there are online documentation for it. Eg: Git documentation
  - For a more thorough explanation of a command, use: man *command*
- Forums like SuperUsers and StackOverflow are your friends! Eg: ls -l
- Cheat sheets like these can be useful to remind yourself of commonly used commands.

Practice Exercises and Feedback!

For the remaining time, we have left a few problems that you can complete to get some terminal practice. We recommend you work on them and ask questions as needed, we'll be around if you need help!

Link to slides: sipb.mit.edu/iap/2022/clf/

Practice Exercises

1. Copy the folder /afs/athena.mit.edu/user/a/g/agrebe/Public/intro-terminal/exercises to your home directory. Navigate into your copy.
2. Open the file “secret-code.txt” and discover the secret code
3. Remove the subfolder “trash” – it’s full of old, unneeded files
4. Fix the spelling of “missspeled-file.txt”
5. Create a new directory “folder” inside exercises
6. Create an empty file blank.txt inside this folder

Practice Exercises (cont.)

7. Using Vim, create a Python file that prints out “Hello World!”, then run it in the terminal.

8. Using Vim, change the file above to print "hello world" (undercase H and W, remove !), and print the *file contents* onto the terminal. Then, Delete this file.

9. Using Vim, create a Bash script (.sh) with the following content:

```
#!/bin/bash
time=$(date)
echo "Current timestamp: $time"
```

Mark it as executable and run it.

Answers to Exercises

1. cp -r /afs/athena.mit.edu/user/a/g/agrebe/Public/intro-terminal/exercises .
   cd exercises
2. vim secret-code.txt (secret code is 42)
3. rm -rf trash
4. mv missspeled-file.txt misspelled-file.txt
5. mkdir folder
6. cd folder
   touch blank.txt

Answers to Exercises (cont.)

7. vim my_script.py

Go into INSERT mode. Type ‘print("Hello world!")’.

Save and exit the file using :wq

python3 my_script.py

Answers to Exercises (cont.)

8. vim my_script.py

Move your cursor around while in ESCAPE mode, and enter INSERT mode as needed to apply the changes.

Save with :wq

cat my_script.py

rm my_script.py

Answers to Exercises (cont.)

9. vim my_script.sh

Go into INSERT mode. Type out the provided commands. Save and exit with :wq

chmod +x my_script.sh

./my_script.sh (or sh my_script.sh)