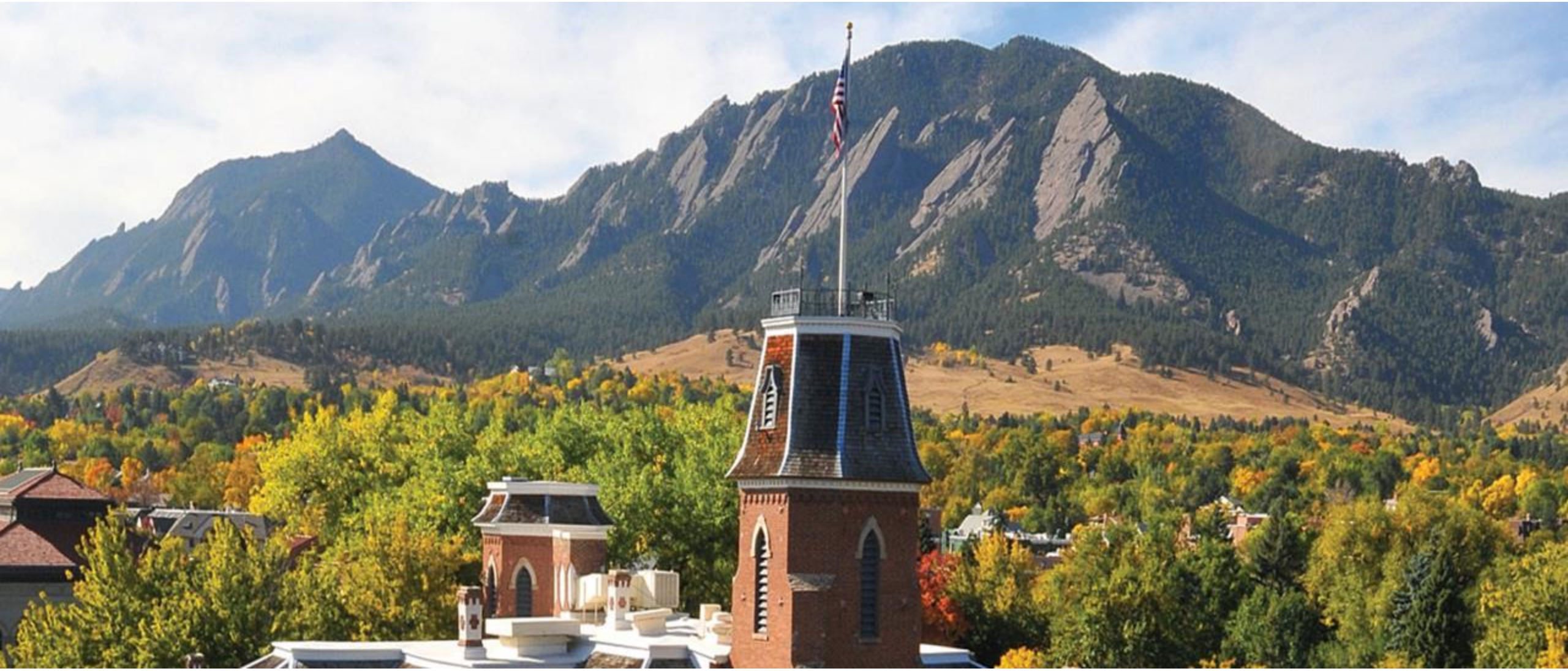
# Working with Linux



# Working with Linux

Workshop Type: Short Course
Instructor: Brandon Reyes
Contact: rc-help@colorado.edu
Website:

- https://www.colorado.edu/rc
- https://curc.readthedocs.io/en/latest/

Slides and other files available for download and viewing:
https://github.com/ResearchComputing/working_with_linux_on_hpc_shortcourse

Contributors: Michael Schneider, Layla Freeborn, Andrew Monaghan, Brandon Reyes, John Reiland, Mohal Khandelwal

# Meet the User Support Team

Layla Freeborn

Brandon Reyes

Andy Monaghan

Michael Schneider

John Reiland

Dylan Gottlieb

Mohal Khandelwal

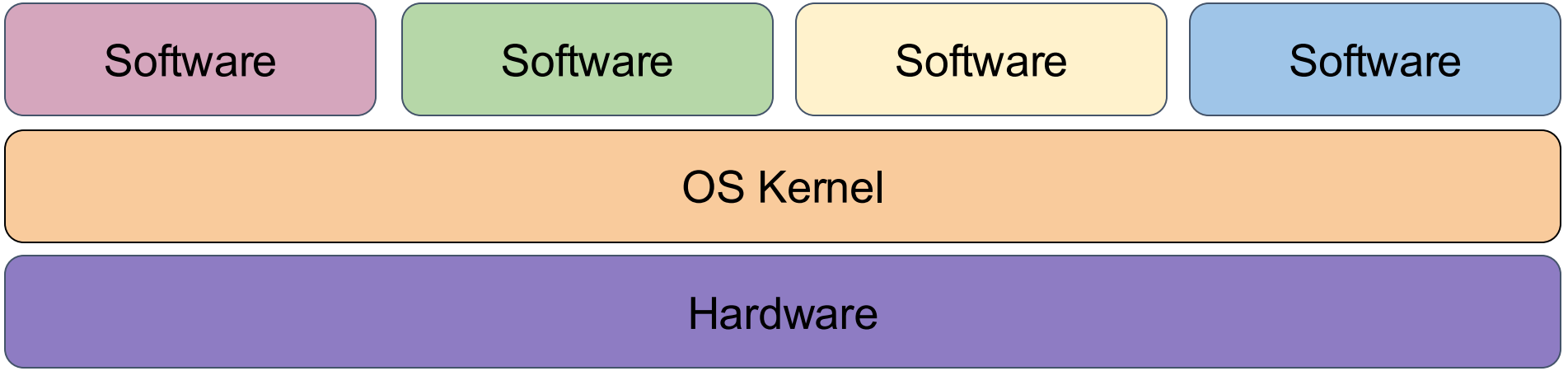
Ragan Lee

# Learning Goals

- Why Linux?
- Working with Files
- Working with Scripts

# What is Linux?

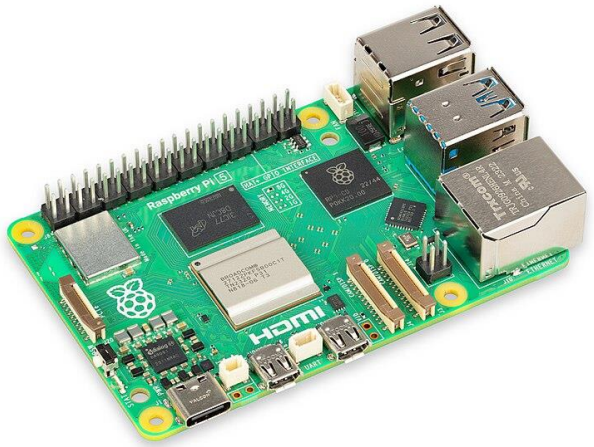
- Created by Linus Torvalds (1991)
- “Unix”-based operating system (like Mac OS)
- Supports a variety of hardware and software systems




images courtesy of wikicommons

# Linux Distros

- Variety of distributions, or distros, available
- Embedded systems (Raspberry PI)
- “Windows” replacement (Ubuntu)
- Commercial/Industry Supported (RedHat)

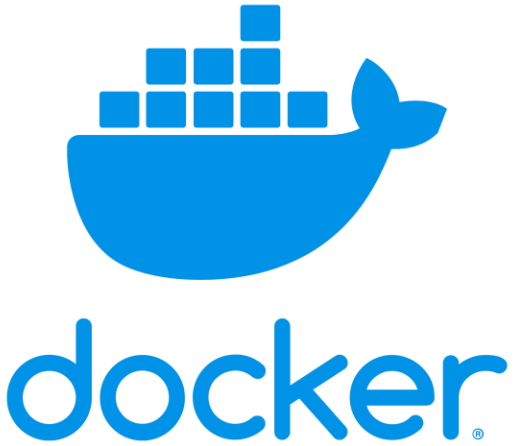


Ubuntu

images courtesy of wikicommons

# Why Use Linux?

- Most common Operating System for HPC systems
- Extremely flexible, fast, and powerful
- Built-in support for many software development workflows





images courtesy of wikicommons

# Opening a Terminal

- Mac: Go to Applications → Utilities → Terminal
- Windows:
  - Download a terminal emulator e.g.
    - Git BASH: https:gitforwindows.org
  - Use Windows Subsystem for Linux (WSL)
- Use our service called Open OnDemand (more on this later):
  - CU Boulder: https://ondemand.rc.colorado.edu/
  - Everyone else: https://ondemand-rmacc.rc.colorado.edu/

# Logging into CURC via terminal

- ssh <rc_username>@login.rc.colorado.edu
- Enter your password
- Authenticate by accepting the Duo push to your smartphone

https://curc.readthedocs.io/en/latest/access/logging-in.html

# Alt: Logging into CURC via browser

- Navigate to https://ondemand-rmacc.rc.colorado.edu
- Choose your organization
- Enter your password
- Authenticate by accepting the Duo push to your smartphone
- Select the “Clusters” app to bring up an Alpine terminal

https://curc.readthedocs.io/en/latest/open_ondemand/index.html

# The Linux Filesystem

- System of arranging files and directories (folders)
- Levels in full paths separated by forward slashes:

  e.g. /home/user/scripts/analyze_data.sh
- Case-sensitive; spaces in names discouraged
- Some shorthand:

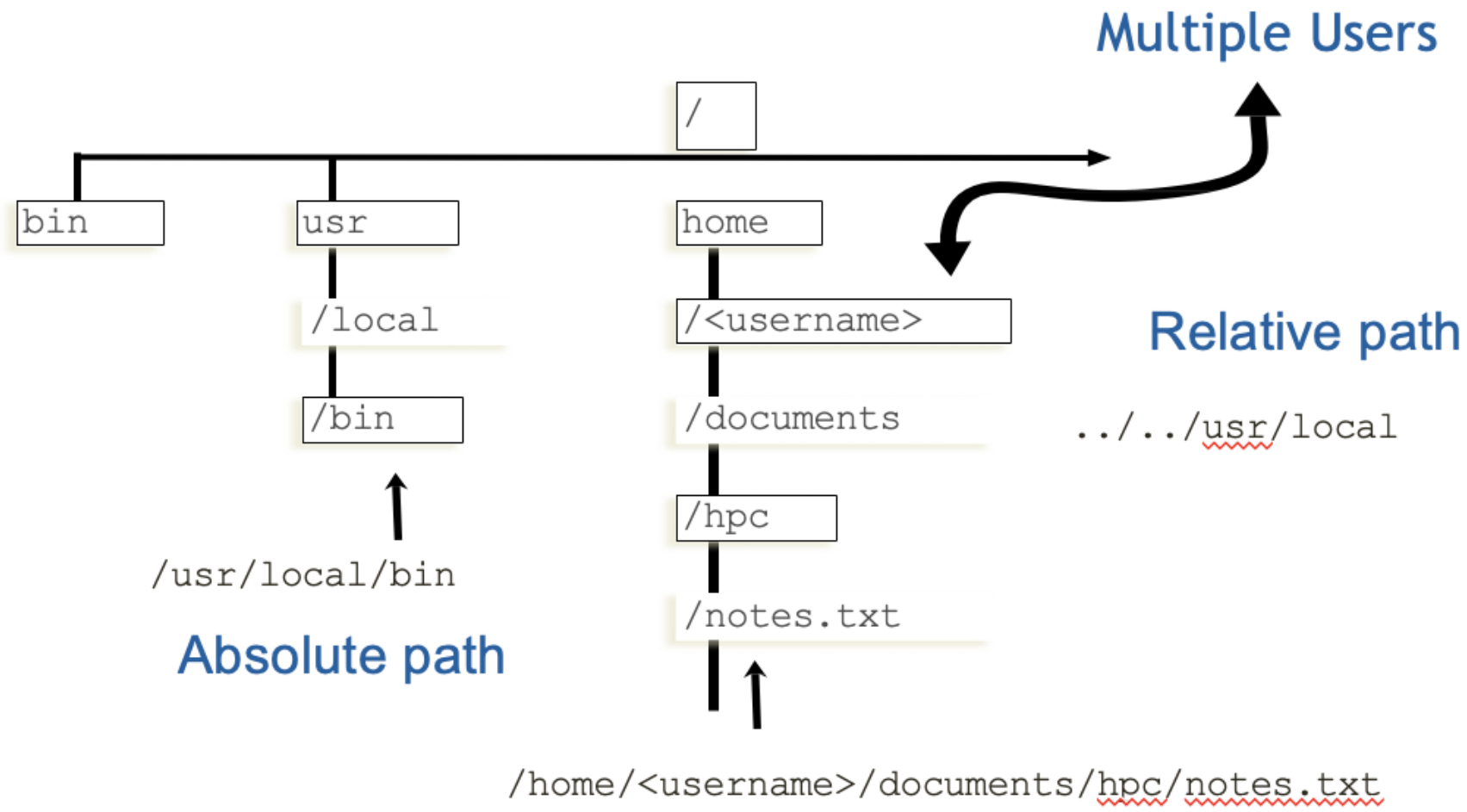
  `.` (the current directory)

  `..` (the directory one level above)

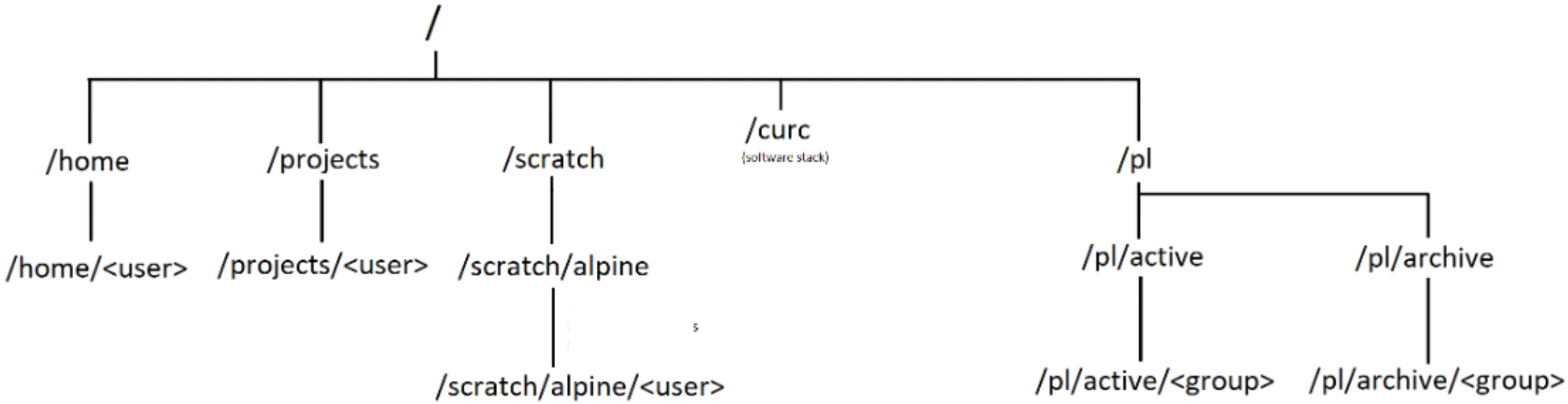
  `~` (home directory)

  `-` (previous directory, when used with `cd`)

# Filesystem Layout



# Your personal directories on CURC



# Environment variables

- Environment variables store important information needed by Linux users and programs
- Type env to see your currently set up environment variables
- Useful environment variables:

| | |
|---|---|
| PATH | directories to search for commands |
| HOME | home directory |
| PWD | current working directory |
| USER | username |
| LD_LIBRARY_PATH | directories to search for dynamically-loaded libraries |

# Display a line of text

```
echo "Hello!"

echo $USER
```

# Anatomy of a Linux command

- command [flags] [target(s)]

```
ls -l myworkdir
```

- Case is important!
- Use “man” command to view a command’s **man**ual

```
man ls
```

# List Files and Directories

```
ls [option] [file/directory]
```

```
  -l (long format)              -a (all files)
  -h (readable file size)       -r (reverse sort)
  -S (sort by file size)
  -t (sort by modification time)
```

# Change Directories

```
cd <path/to/take>
```

```
cd /projects/$USER
```

# Make Directories

```
mkdir <path/directory_name>
```

```
mkdir rc_temp
```

# Text Editors

- **nano** – Beginner friendly
- **vi/vim** – Powerful, but steep learning curve
- **emacs** – Extendable, tons of additional features
- **VS Code** – via OnDemand
- Use local text editor and copy files manually to Alpine

# Create a Text File

```
nano notes.txt
```

# See all content in a file

```
cat notes.txt
```

# Head Command – First X Lines of File

```
head <path/to/file>
```

```
head notes.txt
```

```
head -n 3 notes.txt
```

# Tail Command – Last X Lines of File

```
tail <path/to/file>

tail notes.txt

tail -n 3 notes.txt
```

# Copy Files

```
cp <source> <destination>

cp README.md ../

cp README.md ../Docs.md
```

# Remove Files

```
rm <path/to/file>

rm Docs.md

rm -r <directory>
```

**Be careful when using rm. By default, it does not ask you to confirm the deletion!**

# Intro to Shells and Shell Scripts

A **shell** is the environment in which commands are interpreted in Linux.

GNU/Linux provides numerous shells; the most common is the Bourne Again shell (bash).

Other common shells available on Linux systems include:

- sh, csh, tcsh, ksh, zsh

**Shell scripts** are files containing collections of commands for Linux systems that can be executed as programs. They are powerful tools!

# Shell script basics

- In shell scripts, the first line must specify the shell e.g. #!/bin/bash
- The program loader recognizes the #! and will interpret the rest of the line (/bin/bash) as the interpreter program.
- If a line starts with #, it is a comment and is not run.

```
#!/bin/bash
# the files in /tmp.
cd /tmp
ls
```

Shell to run

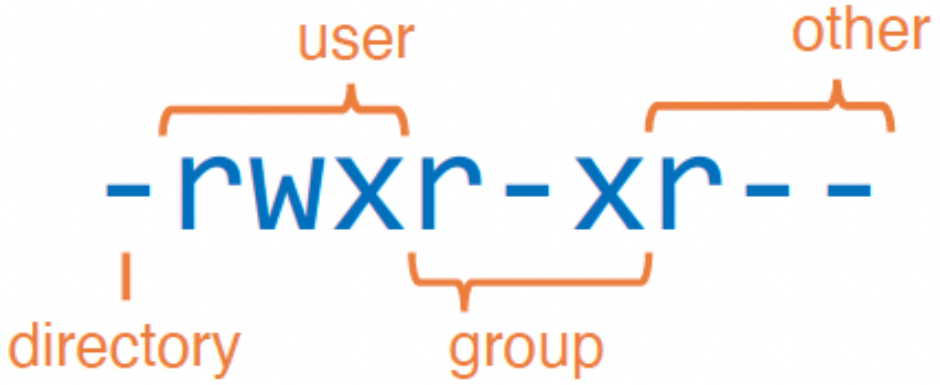
Comment

Change directories

List everything in /tmp

# Modes (aka permissions)

- View file/directory permissions `ls -l`
- 3 classes of users:
  - User (u) aka “owner”
  - Group (g)
  - Other (o)
- 3 types of permissions:
  - Read (r)
  - Write (w)
  - Execute (x)



# Modes (continued)

`chmod` changes modes:

To add write and execute permission for your group:

```
chmod g+wx filename
```

To remove execute permission for others:

```
chmod o-x filename
```

# Run test.sh

```
#!/bin/bash
# print ”Hello” to terminal
echo "Hello"
```

```
chmod u+x test.sh
./test.sh
```

# Local vs inheritable variables

- A variable can contain a number, a character, a string of characters.
- Environment variables are inheritable – can be used in subsequent shells
- Local variables - only effective in the current shell itself

# Local vs inheritable variables

```
[username@login-ci3 rc_temp] export g="Hi!"
```

```
#!/bin/bash
# set local variable
a="bb"
# print out the inherited variable
echo $g
```

# Local vs inheritable variables

```
[username@login-ci3 rc_temp] ./test.sh

[username@login-ci3 rc_temp] echo $a
```

# Thank you!

- **Documentation**: curc.readthedocs.io/
- **Trainings with Center for Research Data and Digital Scholarship (CRDDS)**: https://www.colorado.edu/crdds/
- **Helpdesk**: rc-help@colorado.edu
- **Consult Hours (Tuesday 12:00-1:00 in-person, Thursday 1:00-2:00 virtually)**

# Survey and feedback

https://tinyurl.com/curc-survey18