Logging into CURC and working with Linux

August 18, 2025

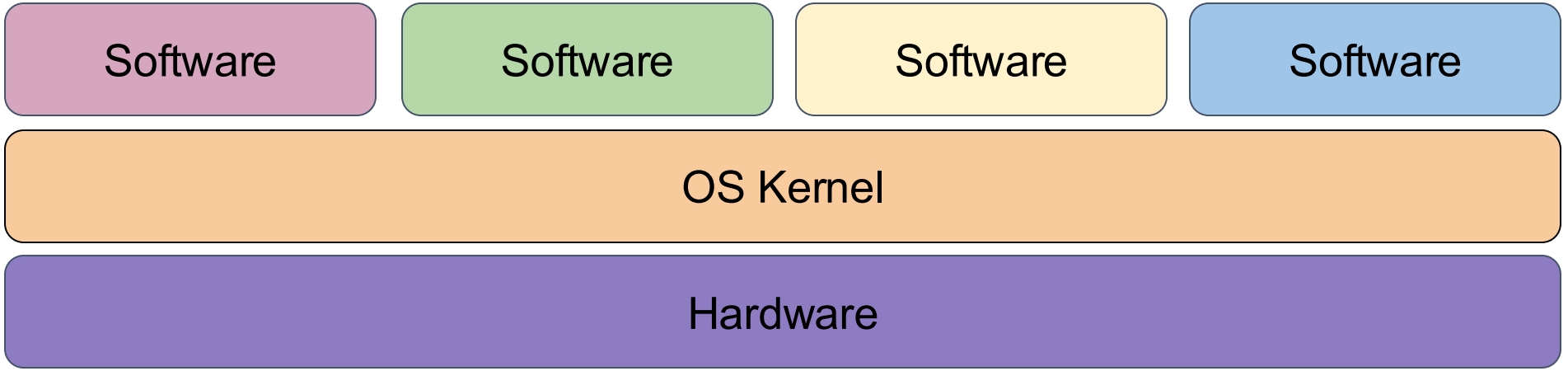
Andy Monaghan

Learning Goals

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

What is Linux?

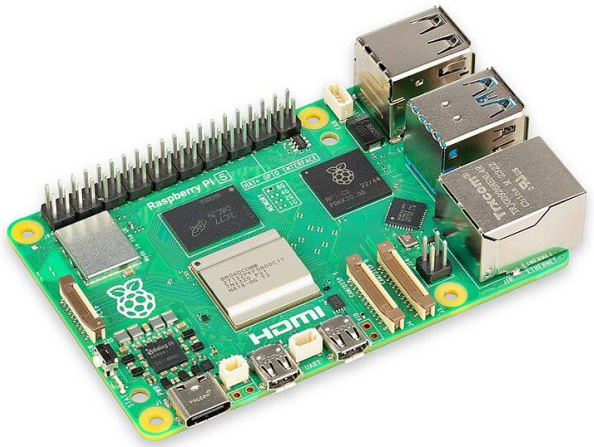
- Created by Linus Torvalds (1991)
- “Unix”-based operating system (like Mac OS)
- Supports a variety of hardware and software systems



images courtesy of wikicommons

Linux Distros

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

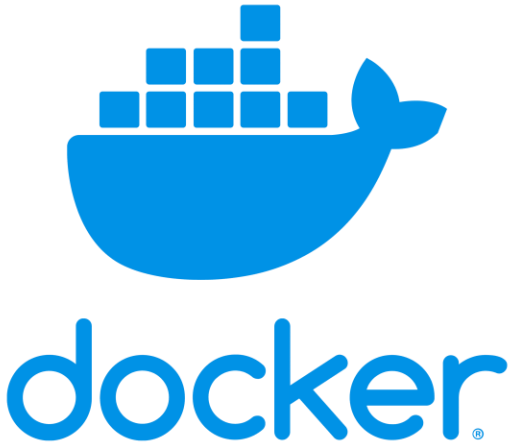


Ubuntu

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 (or use Powershell)
  - PuTTY: https://www.putty.org
  - Git BASH: https:gitforwindows.org

- Any platform: https://ondemand-rmacc.rc.colorado.edu/
  - This is currently the only method for RMACC users

Logging into CURC

- 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:

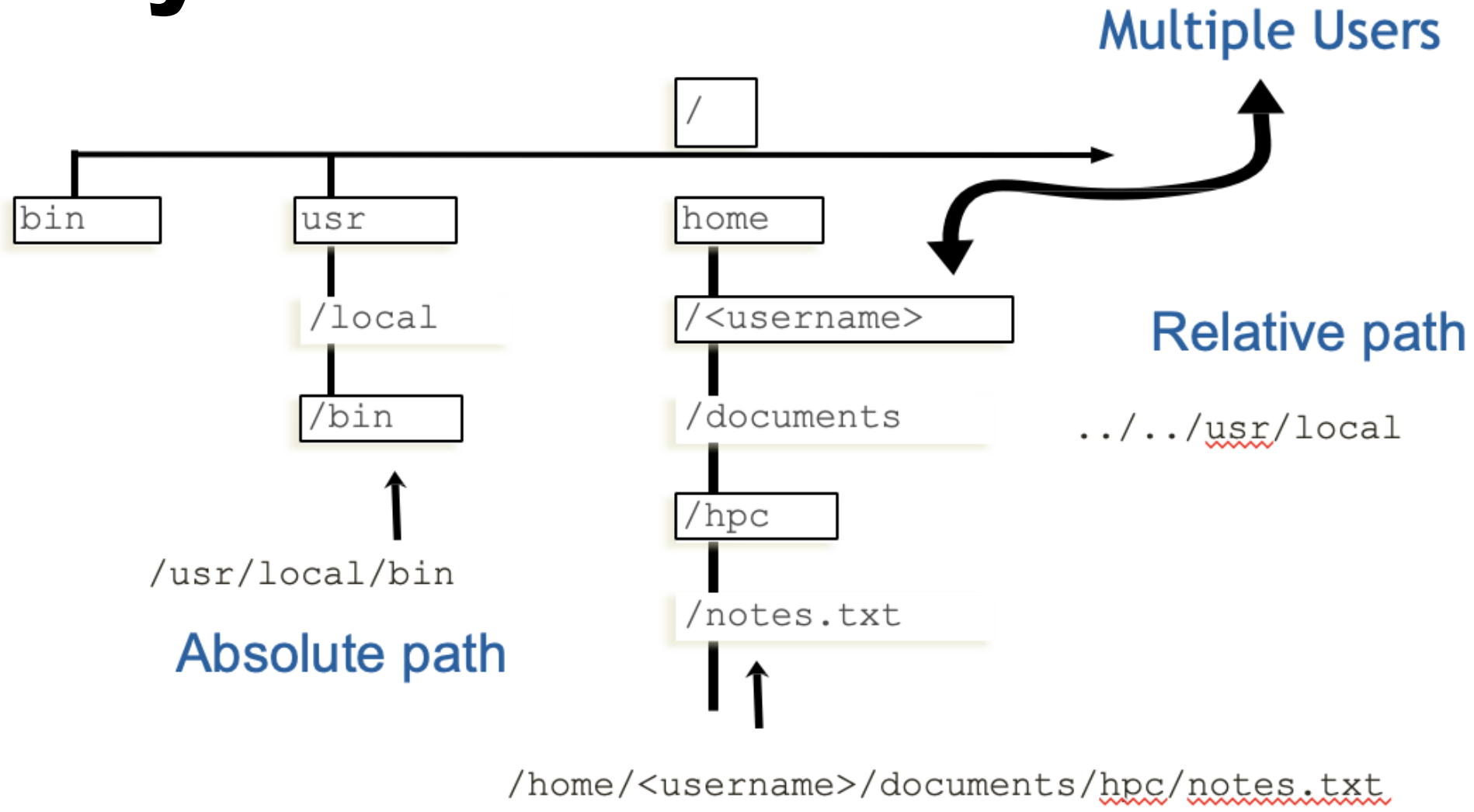
  `.` (the current directory)

  `..` (the directory one level above)

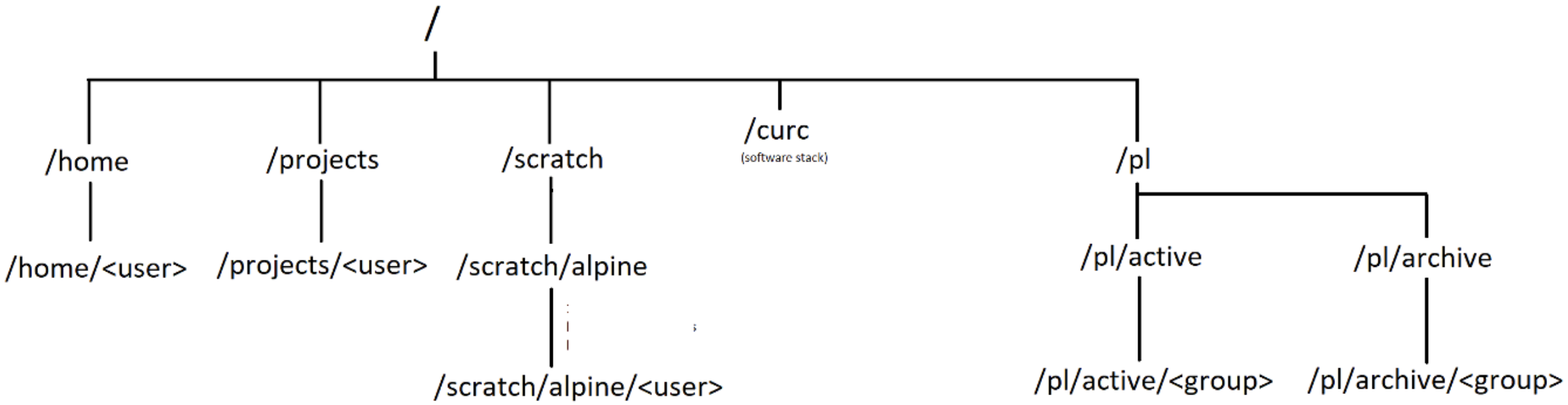
  `~` (home directory)

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

Filesystem


Multiple Users
/
bin
usr
home
/local
/<username>
/bin
/documents
/hpc
/notes.txt
Relative path
../../usr/local
/usr/local/bin
Absolute path
/home/<username>/documents/hpc/notes.txt

Your personal directories on CURC



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
```

Change Directories

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

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

Make Directories

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

```
mkdir rc_temp
```

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)
```

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>
```

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

```
vim notes.txt
```

```
nano 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
```

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.

Anatomy of a shell script

- Executed interactively (terminal) or programmatically (scripts)
- In shell scripts, the first line must contain #!/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.

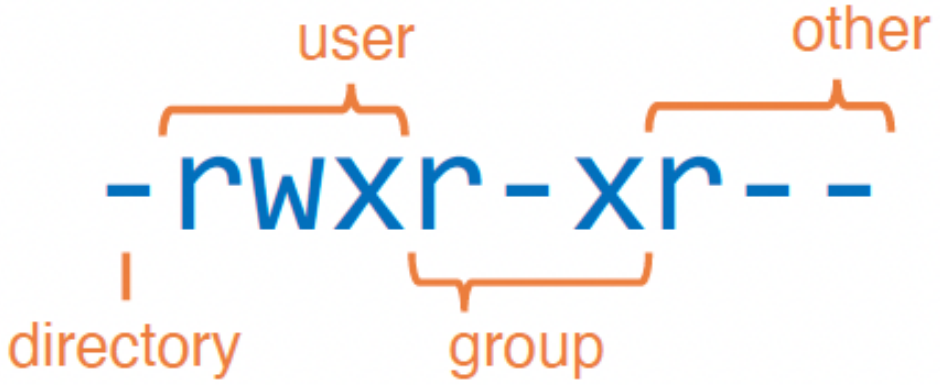
```
#!/bin/bash                 Shell to run
# list files in /tmp.       Comments
cd /tmp                     Change directories
ls                          List everything in /tmp
```

Alternatives for Scripting

- `csh/tcsh` C-shell (tcsh: updated version of csh)
- `ksh` Korn shell; related to sh/bash
- `perl` exceptional text manipulation and parsing
- `python` excellent for scientific and numerical work
- `ruby` general scripting
- `make` building executables from source code

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
```

Let's create a script, *"test.sh"*

Step 1: open new file

```
nano test.sh
```

Step 2: add the following text:

```
#!/bin/bash
# list files in /tmp
cd /tmp
ls
```

Step 3: save the file and exit

Now run test.sh

Step 4: make the script executable

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

Step 5: run the script:

```
./test.sh
```

Local vs Global Variables

- A variable can contain a number, a character, a string of characters.
- Environment variables are global – effective in subsequent shells
- Shell variables are local- only effective in the current shell itself

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

Thank you! Questions?