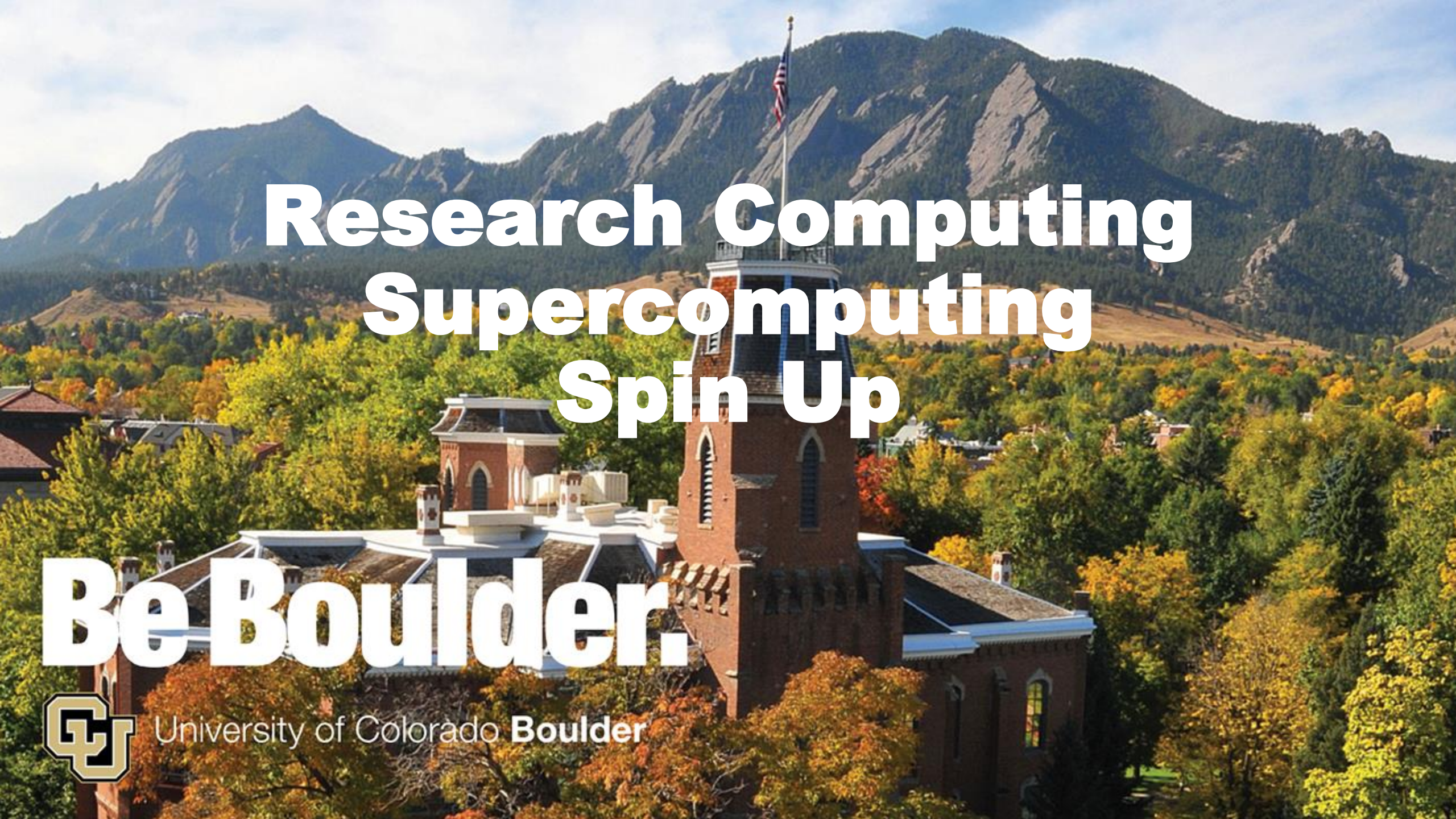

Research Computing
Supercomputing
Spin Up
Be Boulder.
University of Colorado Boulder

Introduction to Linux

Mea Trahan
Contact: datr2651@colorado.edu
Website:
https://www.colorado.edu/rc

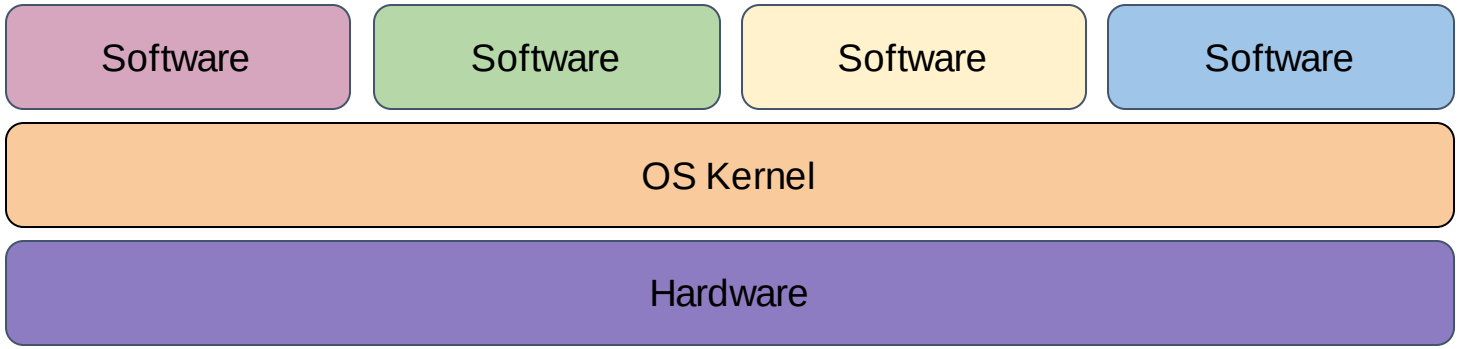
Slides and other files available for download and viewing:
https://github.com/ResearchComputing/Supercomputing_Spinup_Spring_2022

Outline

- What is Linux?
- Why use Linux?
- What happens when you log in?
- Shells and environment
- Files / Directories / Filesystems
- Commands
- Processes
- More about shells

What is Linux?

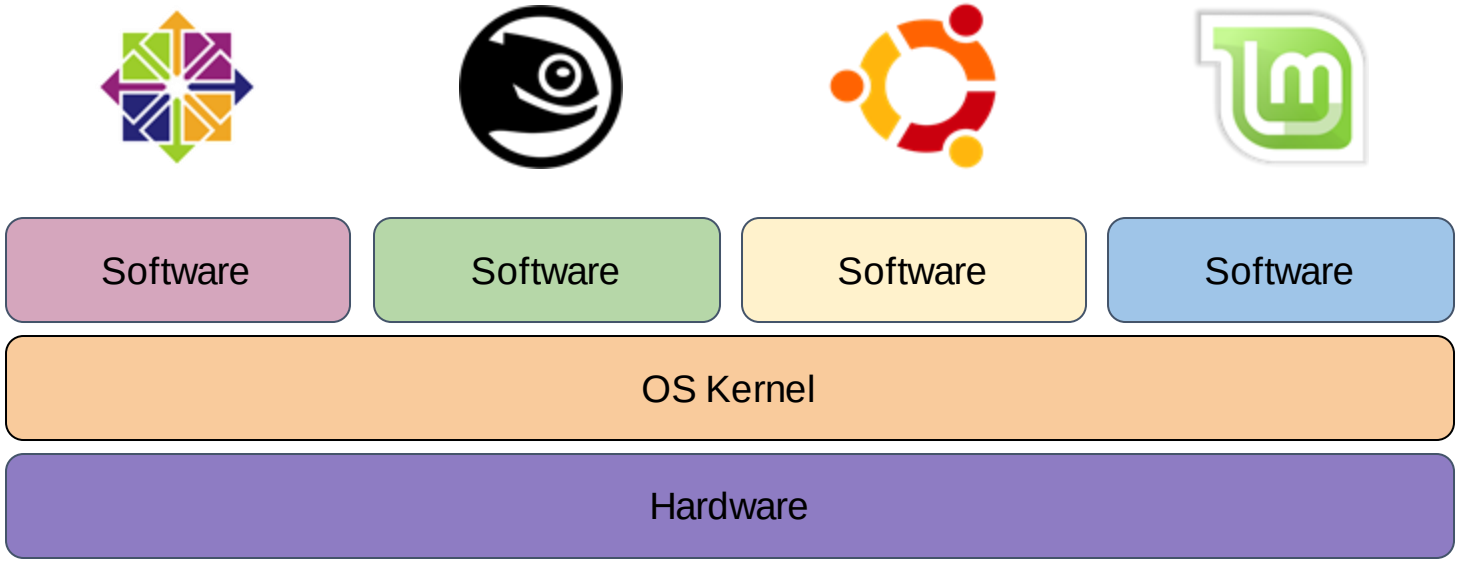
- Part of the Unix-like family of operating systems.
- Started in early ‘90s by Linus Torvalds.
- Typically refers only to the kernel with software from the GNU project and elsewhere layered on top to form a complete OS. Most is open source.



images courtesy of wikicommons

What is Linux?

- Several distributions are available; from enterprise-grade, like RHEL or SUSE, to more consumer-focused, like Ubuntu.
- Runs on everything from embedded systems to supercomputers.



images courtesy of wikicommons

Why Use Linux?

- Default operating system on virtually all HPC systems
- Extremely flexible
- Not overbearing
- Fast and powerful
- Many potent tools for software development
- You can get started with a few basic commands and build from there

How do you log in?

- To a remote system, use Secure Shell (SSH)
- From Windows
  - Non-GUI SSH application: Windows PowerShell
  - GUI SSH application: PuTTY
  - Putty is preferred method.
- From Linux, Mac OS X terminal, ssh on the command line:

```
ssh <username>@login.rc.colorado.edu
```

- Once you are logged on, type the following:

```
git clone https://github.com/ResearchComputing/Supercomputing_Spin_Up_Spring_2020
```

Useful SSH options

- `-X or -Y`
  - Allows X-windows to be forwarded back to your local display
- `-o TCPKeepAlive=yes`
  - Sends occasional communication to the SSH server even when you're not typing, so firewalls along the network path won't drop your "idle" connection

What happens when you log in?

- Login is authenticated (password or key)
- Assigned to a tty
- Shell starts
- Environment is set up
- “Message of the Day” prints
- Prompt

What identifies a Linux user?

- Username / UUID
- Group / GID
- Password (or other authentication info)
- GECOS
- Default shell
- Home directory (ie, home "folder" on disk)

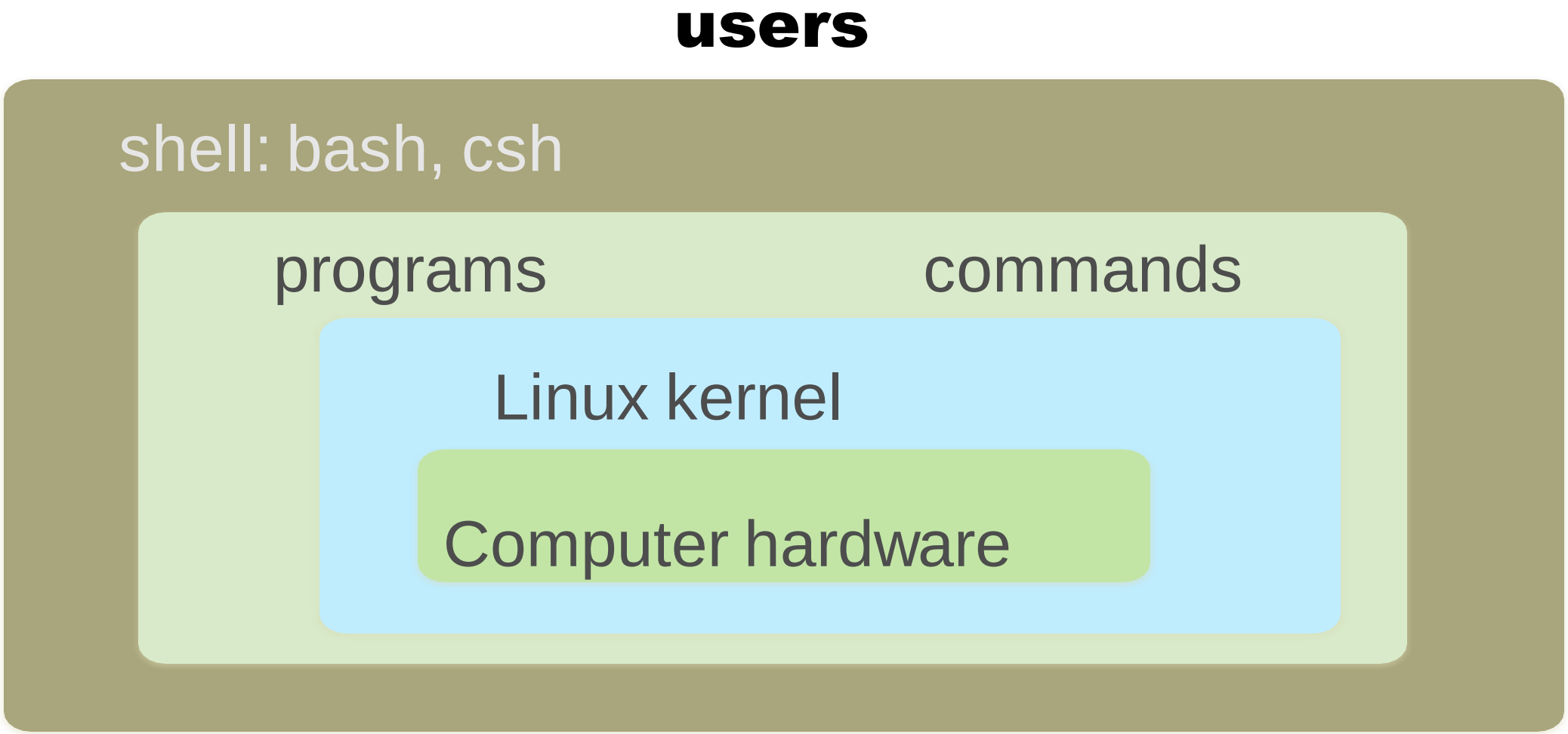
images courtesy of wikicommons

Shells

A command line interpreter: The shell interprets typed input, passes results to the rest of the OS, returns response

- Bourne (sh) – early and rudimentary
- **Bourne-again (bash) – has many user-friendly extensions; default in Linux**
- C (csh) – has C-like syntax
- T (tcsh) – extended version of csh
- Korn (ksh) – early extension of Bourne; was heavily used for programming
- Z (zsh) – includes features of bash and tcsh

Shells


users
shell: bash, csh
programs
commands
Linux kernel
Computer hardware

Shell features

- Tab completion
- History and command-line editing
- Scripting and programming
- Built-in utilities

Anatomy of a Linux command

- command [flags] [flag arguments] [target(s)]
  - `tar -c -f archive.tar mydir`

- Flags may not mean the same thing for different commands
- The same command may have different flags in different kinds of Unix (esp. Linux vs BSD)
- Case is important!
- Order of flags may be important

The most important Linux command man(ual):

man

man <command> (search with “/”)

man -k <keyword>

<command> --help

(or you can google the command on a site you like.)

The Linux Filesystem

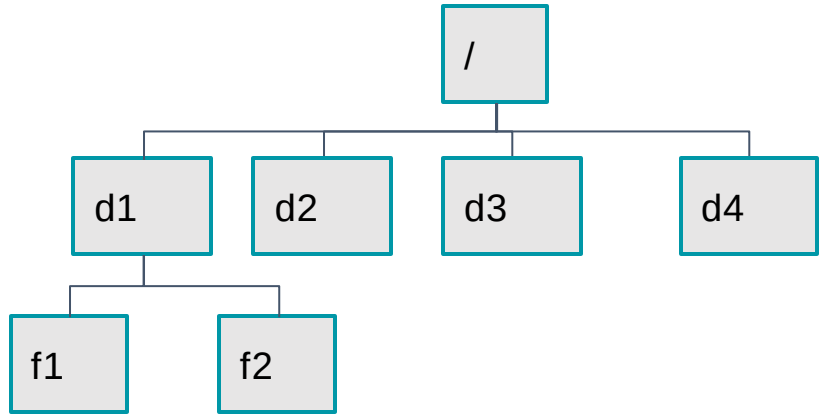
- System of arranging files on disk
- Consists of directories (folders) that can contain files or directories
- Levels in full paths separated by *forward* slashes:
  - e.g. /home/nunez/scripts/analyze_data.sh
- Case-sensitive; spaces in names discouraged
- Some shorthand:

. (the current directory)

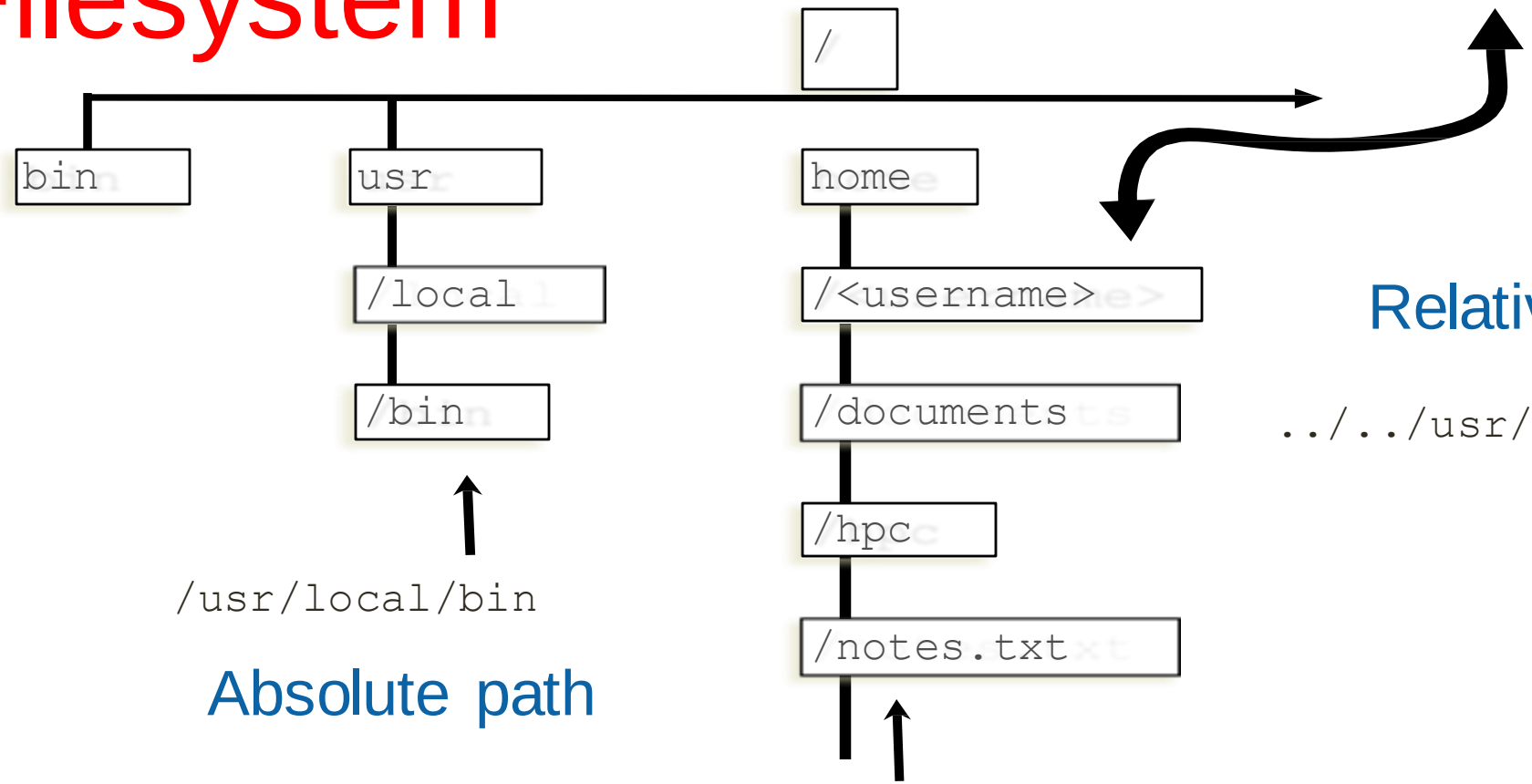
.. (the directory one level above)

~ (home directory)

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



Filesystem



Multiple Users

Relative path

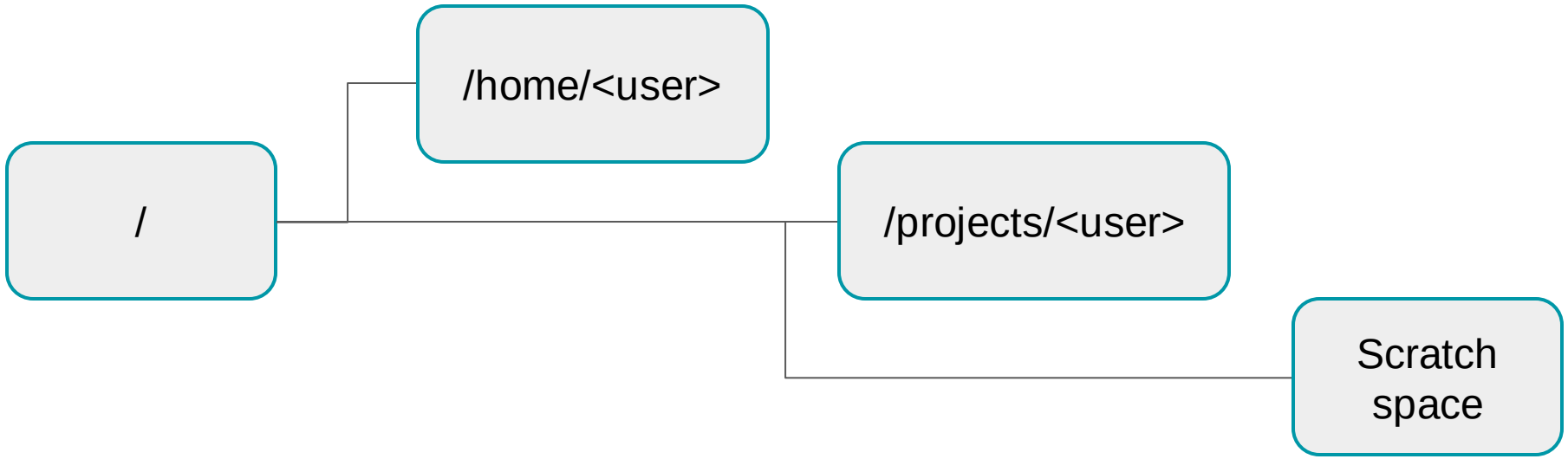
../../usr/local

/usr/local/bin

Absolute path

/home/<username>/documents/hpc/notes.txt

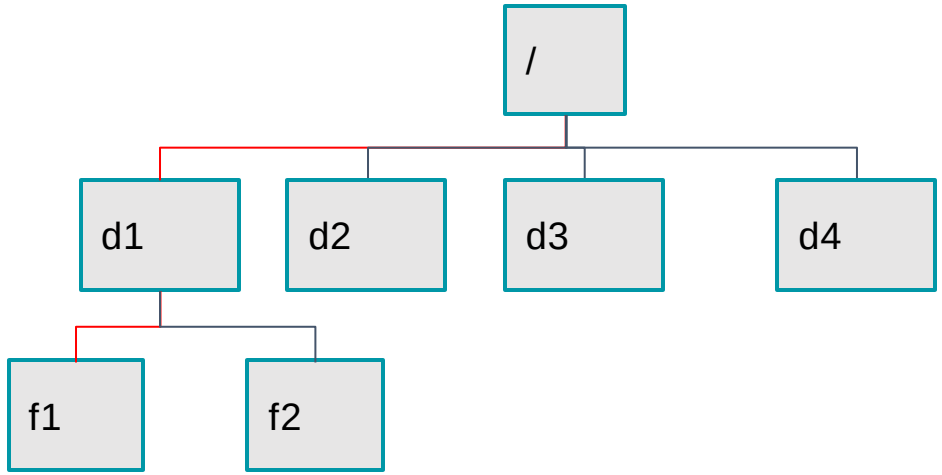
Your Filesystem on RC



Commands

Navigating the filesystem

- Examples:
  - ls
  - mkdir
  - cd
  - rm
- Permissions (modes)



File and Directory related commands

pwd – prints full path to current directory
cd – changes directory; can use full or relative path as target
mkdir – creates a subdirectory in the current directory
rmdir – removes an empty directory
rm – removes a file (`rm -r` removes a directory and all of its contents)
cp – copies a file
mv – moves (or renames) a file or directory

ls – lists the contents of a directory (`ls -l` gives detailed listing)
chmod/chown – change permissions or ownership
df – displays filesystems and their sizes
du – shows disk usage (`du -sk` shows size of a directory and all of its contents in KB)

Exercise 1: Navigation

1. Change to your home directory
2. Change to the repo for this class.
3. Print the path to your current directory
4. List the contents of this directory (extra print “long” list)
5. List the contents of the "bash_tutorial" directory without changing into it
6. Change into the "bash_tutorial" directory
7. Change into the "linux_startup" directory using a single command
8. Change to your home directory and create a new directory (you can pick the name). How can you be sure the new directory is there?

Bonus: Determine how many KB are in "bash_tutorial"

Viewing Files

images courtesy of wikicommons

File-viewing commands

less – displays a file one screen at a time
cat – prints entire file to the screen
head – prints the first few lines of a file
tail – prints the last few lines of a file (with -f shows in realtime the end of a file that may be changing)
diff – shows differences between two files
grep – prints lines containing a string or other regular expression
tee – prints the output of a command and also copies the output to a file
sort – sorts lines in a file
find – searches for files that meet specified criteria
wc – count words, lines, or characters in a file

Exercise 2: File Viewing

1. Change to the repo for this class.
2. Change to the “bash_tutorial” directory
3. Print out the entire “test.sh” file
4. Print out the last 10 lines of “local_vs_global.sh” file
5. Find how many words “case_example.sh” has

Bonus: Find how to print *only* the last 4 lines of “local_vs_global.sh”

Processes

- A process is a unique task; it may have threads
- Examples:
  - Foreground vs background (&)
  - jobs command
  - Ctrl-C vs Ctrl-Z ; bg
  - kill

Process and Program related commands

ps – lists processes (`ps -ef` lists all running processes)

top – shows processes currently using the CPU

kill – sends a signal to a process (kills process by default). Target is Process-ID; found in 2nd column of `ps -ef` output.

jobs – shows jobs currently in background

time – shows how much wall time and CPU time a process has used

nice – changes the priority of a process to get CPU time

Exercise 3: Processes

1. List all of *your* current processes
2. Monitor all current running processes in real time
3. Find the process id (pid) of *your* top process

Environment

- Set up using shell and environment variables
  - shell: only effective in the current shell itself
  - environment: carry forward to subsequent commands or shells
- Set default values at login time using `.bash_profile` (or `.profile`). Non-login interactive shells will read `.bashrc` instead.
- `var_name[=value]` (shell)
- `export VAR_NAME[=value]` (environment)
- `env` (shows current variables)
- `$VAR_NAME` (refers to value of variable)

Useful variables

- `PATH`: directories to search for commands
- `HOME`: home directory
- `DISPLAY`: screen where graphical output will appear
- `MANPATH`: directories to search for manual pages
- `LANG`: current language encoding
- `PWD`: current working directory
- `USER`: username
- `LD_LIBRARY_PATH`: directories to search for shared objects (dynamically-loaded libs)
- `LM_LICENSE_FILE`: files to search for FlexLM software licenses

Exercise 4

Screen dump your whole environment. Type:

```
env
```

File editing

- **nano** – simple and intuitive to get started with; not very feature-ful; keyboard driven
- **vi/vim** – universal; keyboard driven; powerful but some learning curve required
- **emacs** – keyboard or GUI versions; helpful extensions for programmers; well-documented
- **LibreOffice** – for WYSIWYG
- Use a local editor via an SFTP program to remotely edit files.

Modes (aka permissions)

- 3 classes of users:
  - User (u) *aka "owner"*
  - Group (g)
  - Other (o)
- 3 types of permissions:
  - Read (r)
  - Write (w)
  - Execute (x)

```
rwxr-xr--
```

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
```

To set only read and execute for your group and others:

```
chmod go=rx filename
```

Exercise 5:

1. Change directory to `~/<repo for this class>`
2. Use `cat` to show the contents of `hello.sh` in “linux_startup” directory
3. Try to run the program `”hello.sh”` by typing **./hello.sh** at the command line
4. Add execute permission to **`hello.sh`** using `chmod`
5. Can you run it now?
6. Is there a path issue? What are two ways you could get the script to run?

More about shells

- Input and output redirection
  - Send output from a command to a new file with >
  - Append output to an existing file with >>
  - Use a file as input to a command with <
- Pipes: | sends output of one command to another command

```
ps -ef | grep $USER
```

Thank you!

Please fill out the survey!!!
http://tinyurl.com/curc-survey18

Slides and files:
https://github.com/ResearchComputing/Supercomputing_Spinup_Spring_2022

A good introductory online tutorial:
http://www.ee.surrey.ac.uk/Teaching/Unix/index.html