Intro to Linux on RMACC Summit

Andy Monaghan & Dan Trahan

rc-help@colorado.edu

www.colorado.edu/rc

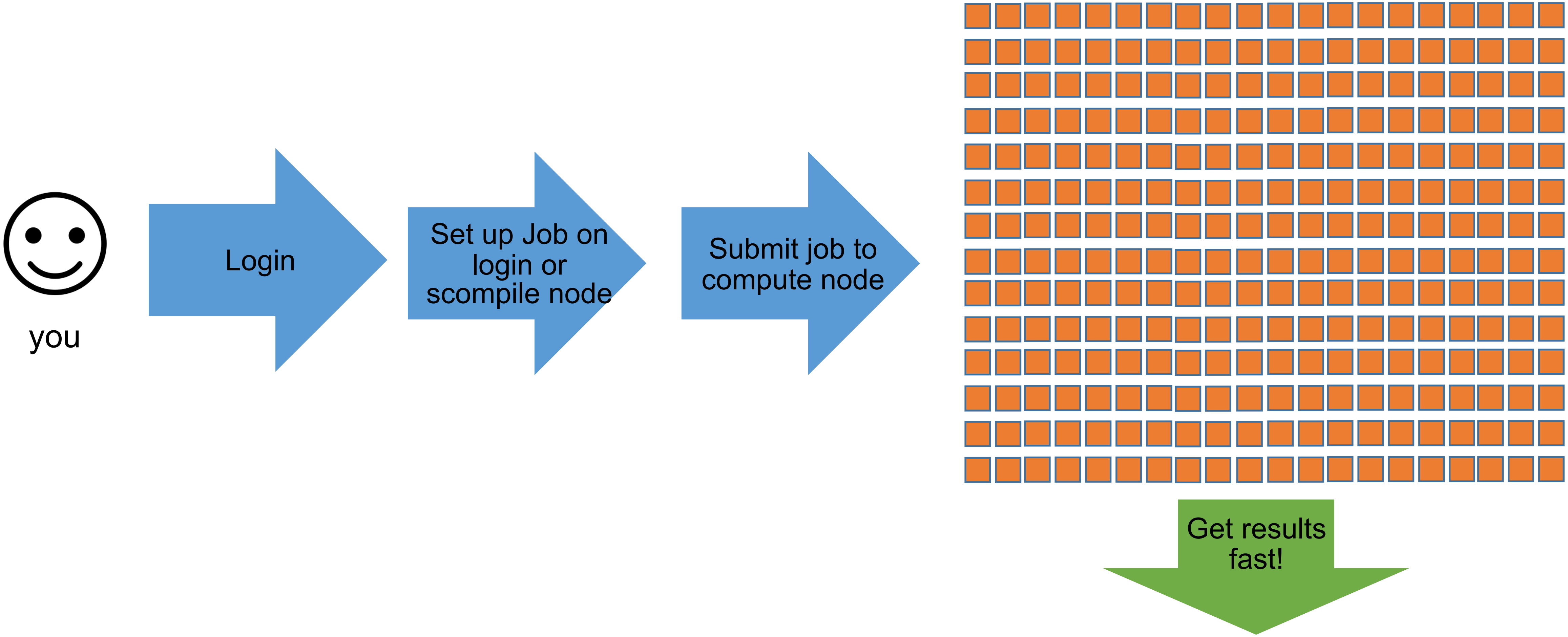
Slides available for download from:

https://github.com/ResearchComputing/CMU_HPC_2019

Outline for this presentation

- Typical workflow on RMACC Summit
- Logging in
- Basic Linux commands
- File editing
- Linux filesystem
- Environment variables
- Software modules on RMACC Summit
- Other Linux topics (environment variables, modes, wildcards)

Typical RMACC Summit Workflow



you

Login

Set up Job on login or scompile node

Submit job to compute node

Get results fast!

RC Access: Logging in

- If you have an RMACC account already, login as follows from a terminal:
  - `$ ssh janedoe@login.xsede.org` *[”janedoe”= your xsede username]*
    - *[enter xsede passcode and accept duo push or call to phone]*
    - `[janedoe@ssohub ~]$ gsissh rmacc-summit`

- If you do not have an RMACC account use one of our temporary accounts:
  - `$ ssh userNNNN@tlogin1.rc.colorado.edu`
    - *[enter temporary password provided by me]*

Working on RC Resources

- When you first log in, you will be on a login node. Your prompt will look like this (e.g.):

```
[user0049@tlogin1 ~]$
```

- The login nodes are lightweight virtual machines primarily intended to serve as ‘gateways’ to RC resources. If you plan to work on Summit (most will), your first step should always be to move to a Summit ‘scompile node’:

```
[user0049@tlogin1 ~]$ ssh scompile
```

- Now go to your working directory and download the material for this workshop:

```
[user0049@shas0137 ~]$ cd /scratch/summit/$USER
[user0049@shas0137 ~]$ git clone https://github.com/ResearchComputing/CMU_HPC_2019
```

Linux

- Part of the Unix-like family of operating systems.
- Started in early ‘90s
- Several distributions are available – from enterprise-grade, like RedHat Linux (RHEL), to more consumer-focused, like Ubuntu.
    - Blanca nodes presently run RHEL7
- Runs on everything from embedded systems to supercomputers.
- Linux is simple, flexible, fast, many potent tools

Anatomy of a Linux command

- command [flags] [flag arguments] [target(s)]
  - ls -l myworkdir/
- Case is important!
- Help on commands is available through the ”man” command (short for manual). E.g.,
  - man ls

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 -skh` shows size of a directory and all of its contents in KB and human readable)

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.
- **time** – shows how much wall time and CPU time a process has used
- **free** – memory usage

File-viewing commands

- **more** – 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 real time 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 (ps –ef | grep XX)
- **sort** – sorts lines in a file
- **find** – searches for files that meet specified criteria
- **wc** – count words, lines, or characters in a file

File editing with nano

- To edit a file:
    - nano myfile.txt

- From within Nano:
    - Ctrl+o save (need to confirm filename)
    - Ctrl+x exit
    - Ctrl+k cut
    - Ctrl+u paste

- Other population Linux editors: vi, emacs

The Linux Filesystem

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

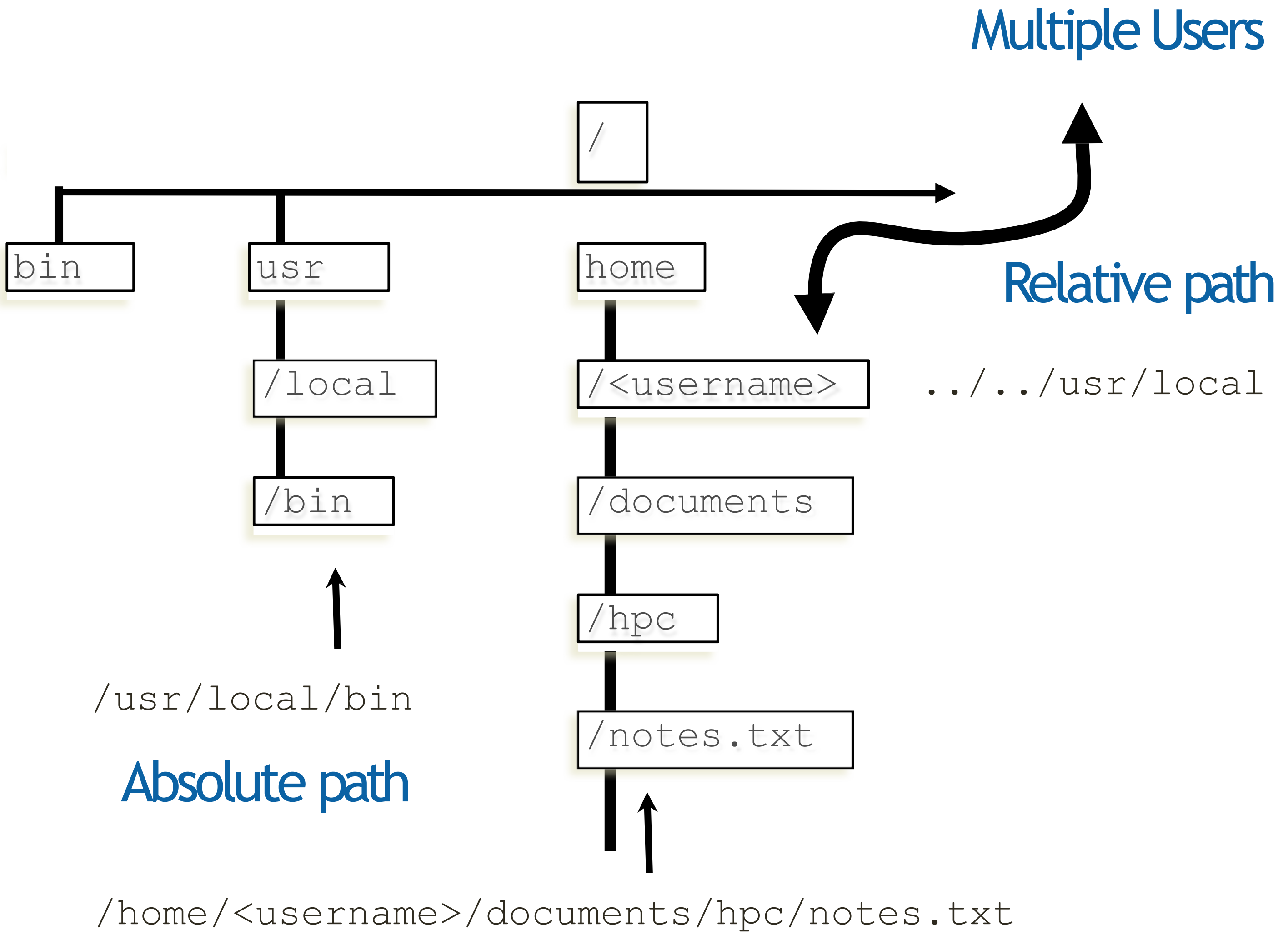
  . (the current directory)

  .. (the directory one level above)

  ~ (home directory)

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

Filesystem



Multiple Users

/

bin

usr

/local

/bin

home

/<username>

/documents

/hpc

/notes.txt

Relative path

../../usr/local

/usr/local/bin

Absolute path

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

Your personal directories on Summit

- /home/<username>
  - Very small: 2GB.
  - Backed up daily.
  - Good for ‘can’t lose’ files
- /projects/<username>
  - 250 GB
  - Backed up regularly
  - Good for storing scripts, self-installed software, some data
- /scratch/summit/<username>
  - Large: 10 TB partition
  - Fast filesystem -- Good for jobs with lots of I/O – run your jobs on here!
  - Not backed up
  - Temporary: data deleted 90 days from creation.

Environment variables

- Environment variables store important information needed by Linux users, programs, etc.

- Type ‘env’ to see your currently set 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 shared objects (dynamically-loaded libs)

Software

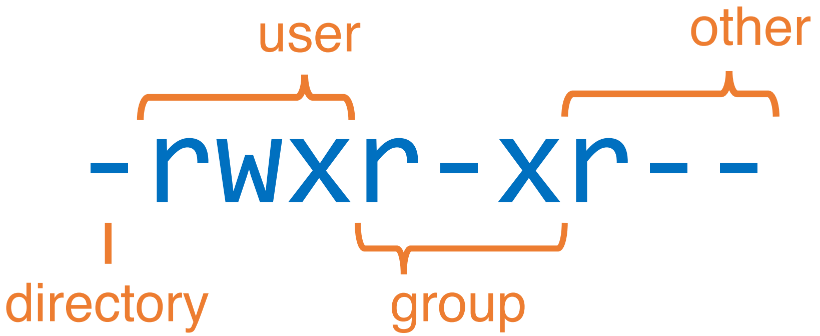
- Common software is available to everyone on the systems
- Can install your own software
  - But you are responsible for support
  - We are happy to assist
- RC uses modules to manage software
  - You can load modules to prepare your environment for using software
    - Loading sets any environment variables
    - Enables application to find appropriate libraries, etc.

Using Modules

- Some modules might require a specific hierarchy to load
  - For some modules, you may need to specify a specific version
    - For example, `module load R/3.3.0`
  - For other modules, you may be able to be more generic
    - For example, `module load matlab`
- Some modules may require you to first load other modules that they depend on
- To find dependencies for a module, type `module spider <package>`
- To find out what software is available, you can type `module avail`
- To set up your environment to use a software package, type `module load <package>/<version>`

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)



user

other

-rwxr-xr--

directory

group

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
```

Shell Wildcards and Special Characters

- `*` - matches zero or more characters
- `?` - matches a single character
- `#` - comment; rest of the line is ignored
- `\` - escape; don’t interpret the next character

Questions?

Presenter: Andrew Monaghan
Email rc-help@colorado.edu

Link to course evaluation:
http://tinyurl.com/curc-survey18

Documentation:
https://curc.readthedocs.io

Slides:
https://github.com/ResearchComputing/**CMU_HPC_2019**

- More detailed tutorials from our “HPC Fundamentals” course:

https://github.com/ResearchComputing/Fundamentals_HPC_Spring_2019