Outline

• What is Linux?
• Why use Linux?
• What happens when you log in?
• Shells and environment
• Commands
• Filesystem basics
• Processes
• More about shells

Slides available at
What is Linux?

- Part of the Unix family of operating systems.
- Started in early ‘90s by Linus Torvalds.
- Technically refers only to the kernel; software from the GNU project and elsewhere is layered on top to form a complete OS. Most is open source.
- Several distributions are available – from commercial-grade, like RHEL or SUSE, to more consumer-focused, like Ubuntu.
- Runs on everything from embedded systems to supercomputers.
Why Use Linux?

- Linux command-line syntax may seem overwhelming to the new user, but:
- It’s the default operating system on virtually all HPC systems
- It’s extremely flexible
- It tries not to get in your way
- It’s fast and powerful
- It was designed by programmers and thus has many potent tools for software development

- You can get started with a few basic commands and build from there
History of Linux

Brian Kernighan
1970
“space travel” to Unix

Dennis Ritchie
1971
C

Richard Stallman
1983
Gnu Not Unix

Linus Torvalds
1991
Linux kernel for personal computers
users

shell: bash, csh

programs commands

Linux kernel

Computer hardware
How do you log in?

• To a remote system, use Secure Shell (SSH)

• From Windows – GUI app such as PuTTY

• From Linux – ssh on the command line
  
  `ssh -X username@login.rc.colorado.edu`

• From Mac OS X – ssh from the Terminal, or GUI such as Cyberduck or Fugu
What happens when you log in?

- Login is authenticated (password or key)
- Assigned to a tty
- Shell starts
- Environment is set up
- Prompt
What identifies a Linux user?

- Username / UUID
- Group / GID
- Password (or other authentication info)
- GECOS
- Default shell
- Home directory
Shells

The shell parses and interprets typed input; passes results to the rest of the OS; returns response as appropriate

- 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 C
- Korn (ksh) – early extension of Bourne; was heavily used for programming
- Z (zsh) – includes features of bash and tcsh
Shell features

- Tab completion
- History and command-line editing
- Scripting and programming
- Built-in utilities
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. Use `.my.bashrc` in RC!
• Initialization scripts should not produce output!
  
• `set var_name[=value]` (shell)
• `export VAR_NAME[=value]` (environment)
• `env` (shows current variables)
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
Anatomy of a Linux command

- Command [flags] [flag arguments] [target(s)]
- `tar –c –f archive.tar mydir`
- Flags do 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
Most important Linux command

man

man <command>
man –k <keyword>
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
**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)
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 2**nd** 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
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
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/admin/mary/payroll/June2012`
- Case-sensitive; spaces in names discouraged
  - `.`, `..`, and `~` are shorthand.

Much more on this in the next session!
Navigating the filesystem

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

- **nano** – simple and intuitive to get started with; not powerful; keyboard driven
- **vi/vim** – universal; keyboard driven; powerful but some learning curve required
- **emacs** – keyboard or GUI versions; helpful extensions for programmers; well-documented
- **OpenOffice / LibreOffice** – for WYSIWYG

http://xkcd.com/378/
Processes

• A process is a unique task; it may have threads
• Examples:
  • Foreground vs background ( & )
  • jobs command
  • Ctl-C vs Ctl-Z ; bg
  • kill
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 ruprech
  ```
- Quoting – save this for a future session!
Thank you!

Slides available at:

http://researchcomputing.github.io