

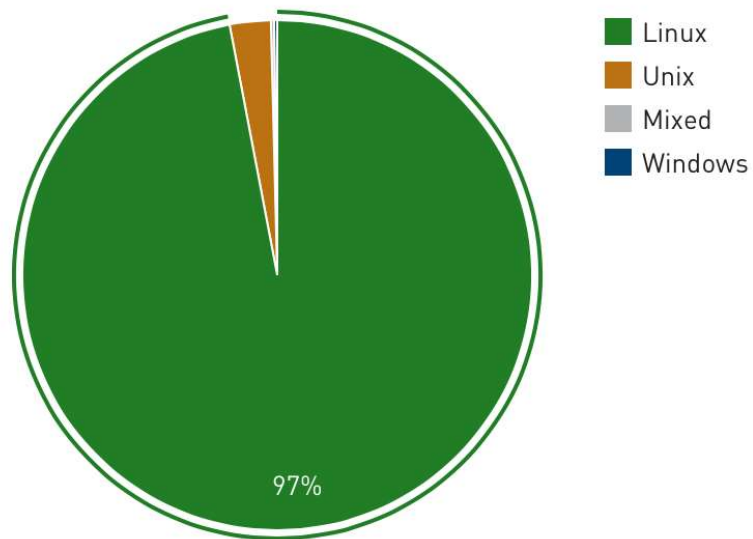
Introduction to Linux

Yuwu Chen
HPC User Services
LSU HPC LONI
sys-help@loni.org

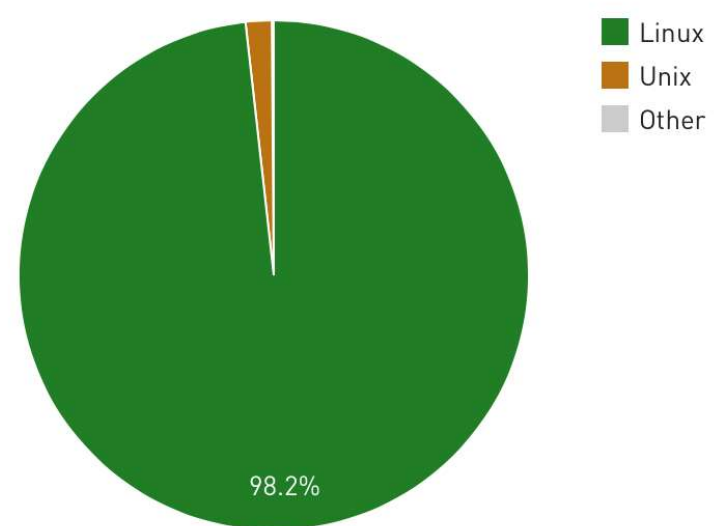
Louisiana State University
Baton Rouge
January 29, 2020

Why Linux for HPC - 2014

OS family system share



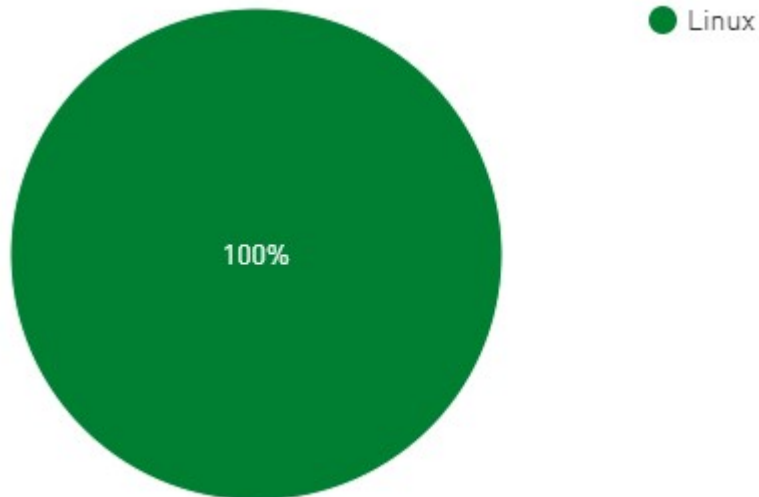
OS family performance share



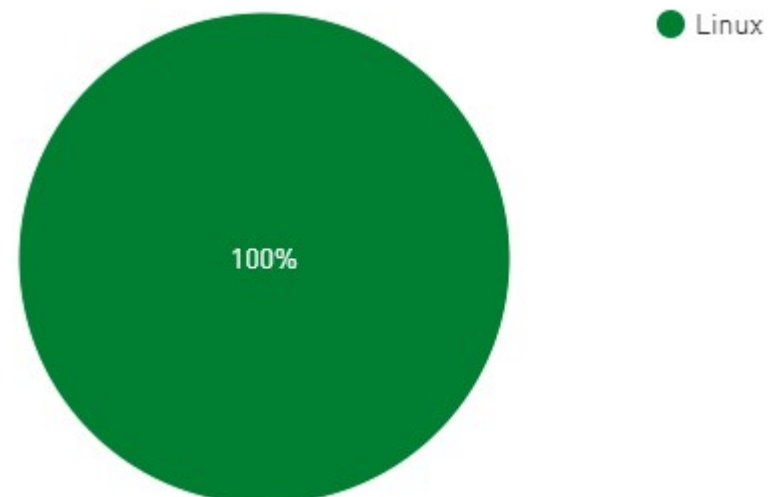
<http://www.top500.org/statistics/list/> November 2014

Why Linux for HPC - 2019

OS family system share



OS family performance share



Linux is the most popular OS used in supercomputers

<http://www.top500.org/statistics/list/> November 2019

Why you should learn Linux?

- Linux is everywhere, not just on HPC
 - Linux is used on most web servers. Google or Facebook
 - Android phone
- Linux is free and you can personally use it
- You will learn how hardware and operating systems work
- A better chance of getting a job

Roadmap

- What is Linux
- Linux file system
- Basic commands
- File permissions
- Variables
- Use HPC clusters
- Processes and jobs
- File editing

History of Linux (1)

- Unix was initially designed and implemented at AT&T Bell Labs 1969 by Ken Thompson, Dennis Ritchie, Douglas McIlroy, and Joe Ossanna
- First released in 1971 and was written in assembly language
- Re-written in C by Dennis Ritchie in 1973 for better portability (with exceptions to the kernel and I/O)

History of Linux (2)

- Linus Torvalds, a student at University of Helsinki began working on his own operating system, which became the "Linux Kernel", 1991
- Linus released his kernel for free download and helped further development

Hello everybody out there using minix-
I'm doing a (free) operation system (just a hobby,
won't be big and professional like gnu) for 386(486) AT clones.

I've currently ported bash (1.08) and gcc (1.40),
and things seem to work. This implies that i'll get
something practical within a few months, and I'd like to know
what features most people want. Any suggestions are welcome,
but I won't promise I'll implement them :-)



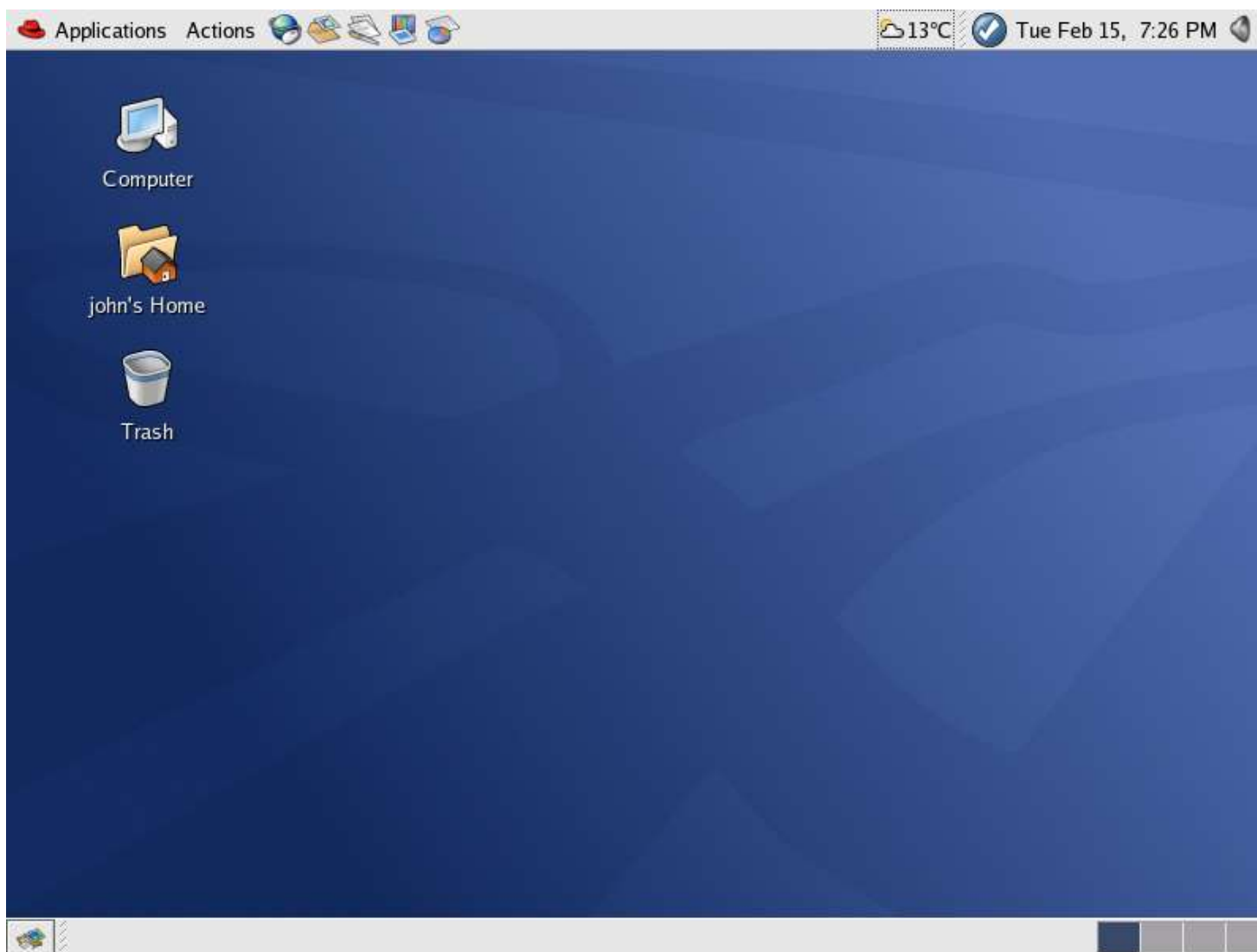
History of Linux (3)

- Linux was the kernel only, applications on top of the kernel were still missing
- The GNU Project by Richard Stallman started in 1983
-Creating a “complete Unix-compatible software system” with entirely free software
- "GNU/Linux"(Linux): Linux kernel + free software from the GNU project
- GNU/Linux (Linux) released under the GNU Public License (GPL): Free to use, modify and re-distribute **iff** later distributions are also under GPL

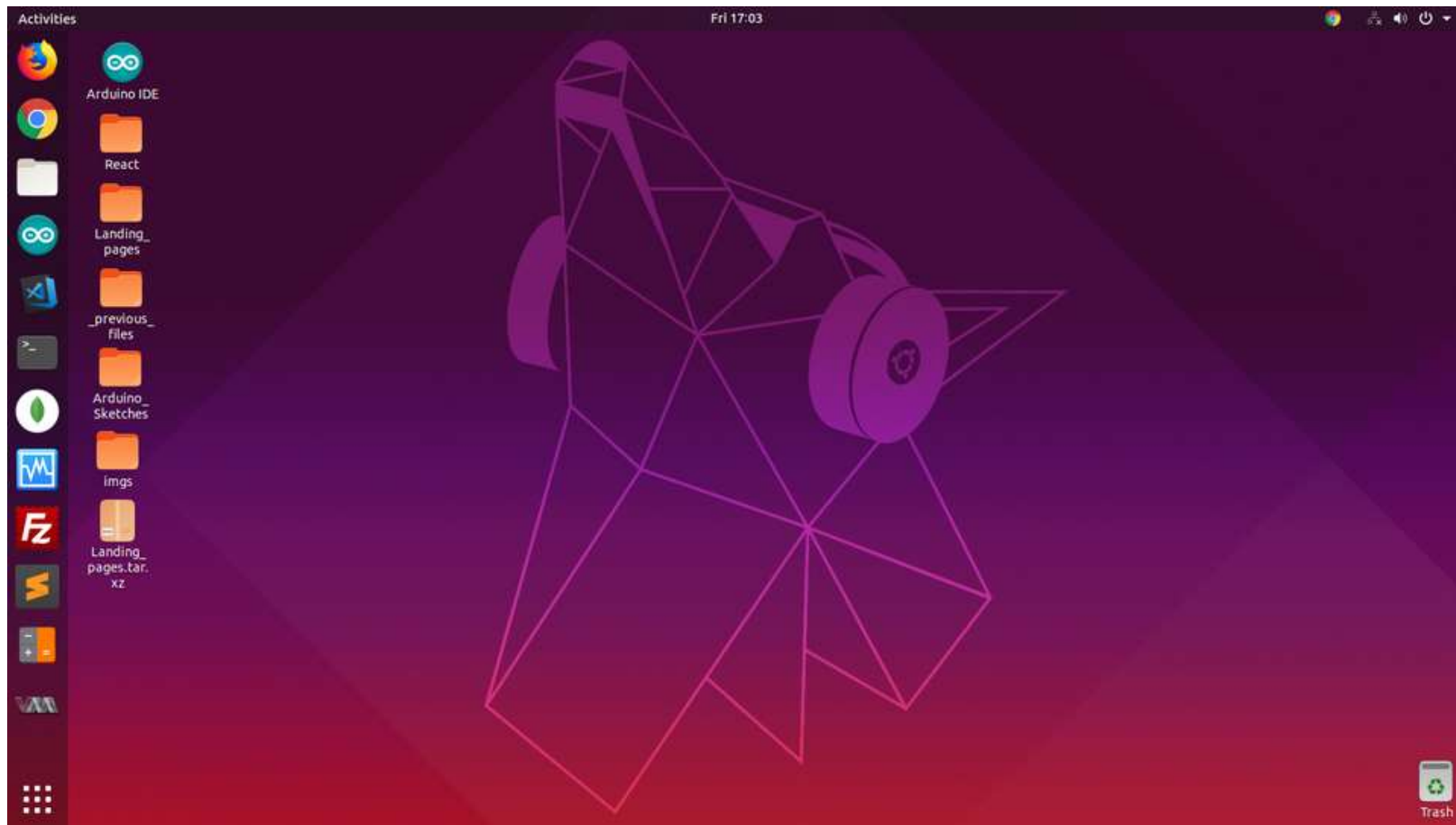
What is Linux

- Essential components: Linux kernel + GNU system utilities + installation scripts + management utilities etc.
- Many software vendors release their own installable packages, known as **distributions**
 - Debian, Ubuntu, Mint
 - Red Hat, Fedora, CentOS, Scientific Linux
 - Slackware, OpenSUSE, SLES, SLED
 - Gentoo
- Linux distributions offer a variety of desktop environment
Redhat, KDE, GNOME, XFCE, LXDE, Cinnamon, MATE

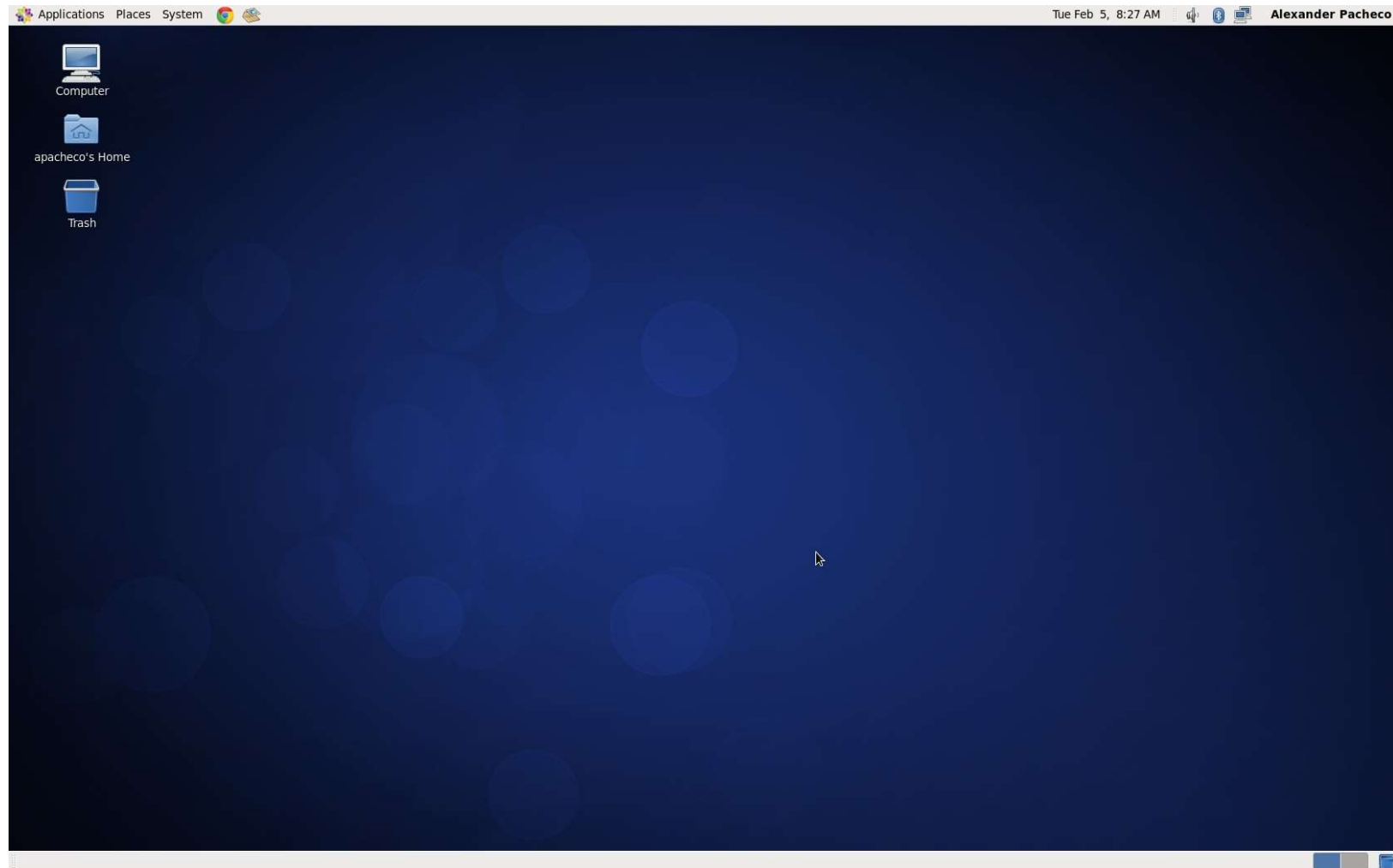
Redhat Desktop



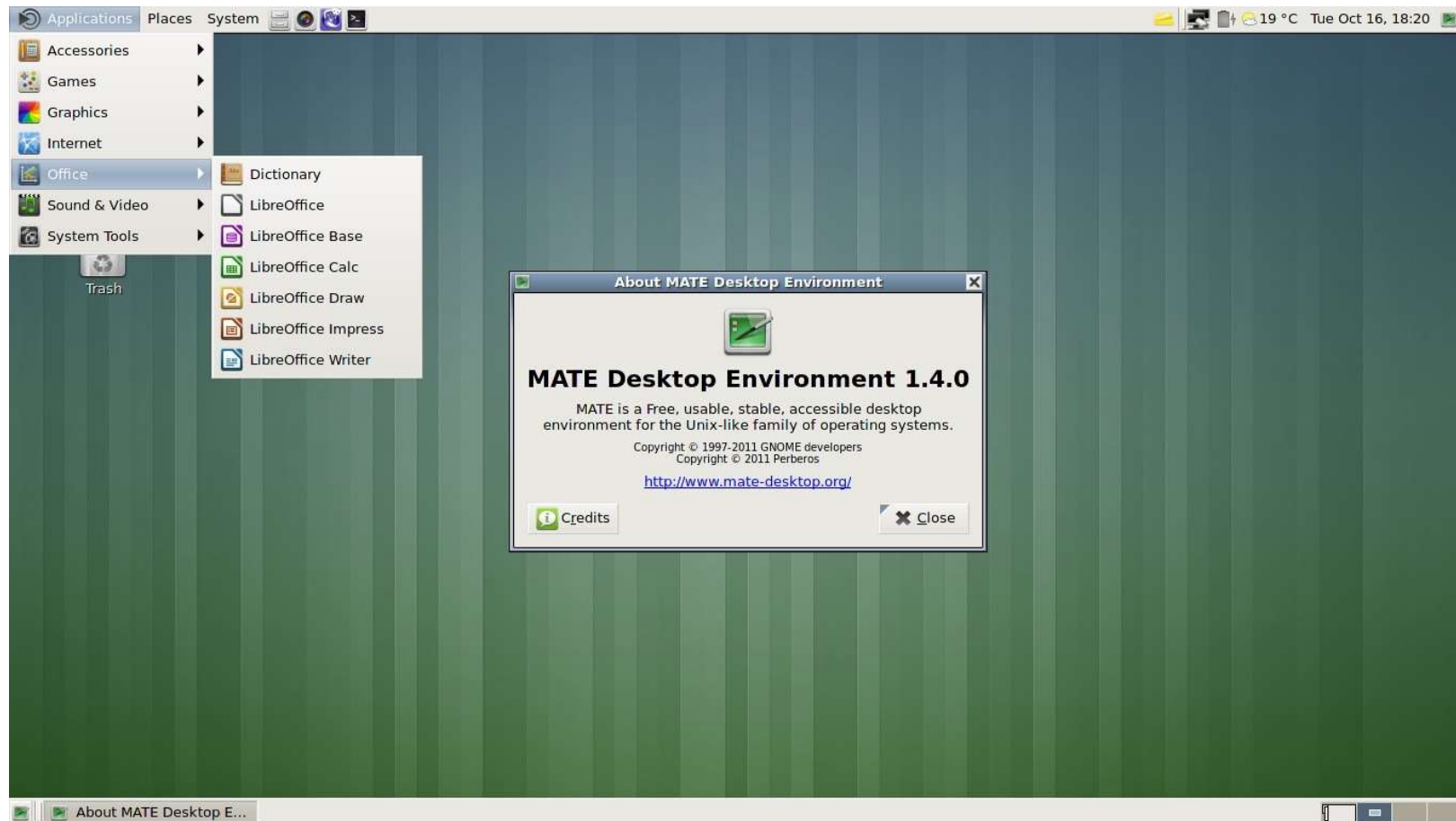
Ubuntu Desktop



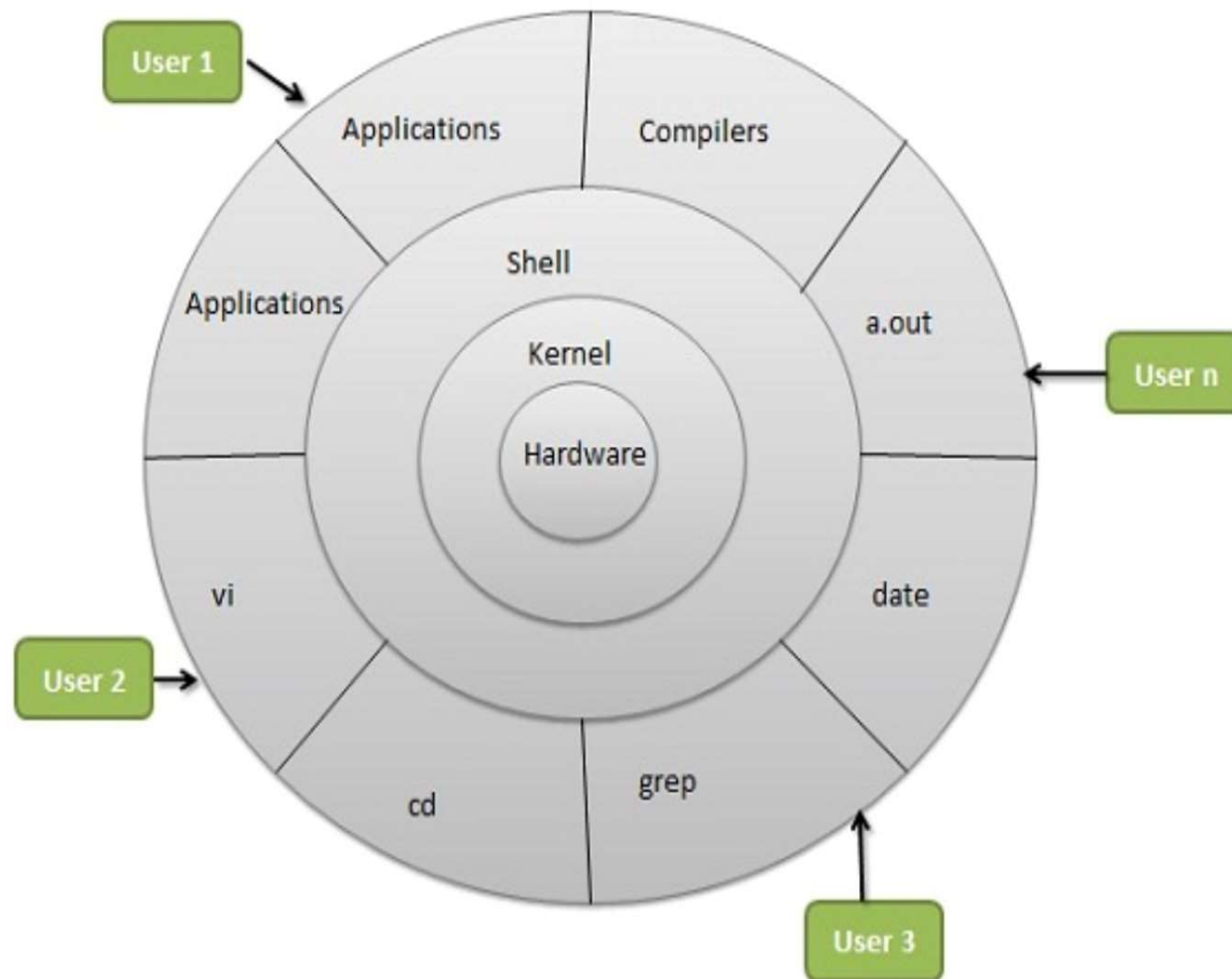
CentOS GNOME Desktop



Debian MATE Desktop



Linux System Architecture



Linux Kernel

What is a Kernel

- The core component of an OS
- Manage the system's resources, memory, file systems...
- Provide the lowest level abstraction layer to upper layer components
- Inter-process communications and system calls are used to make services available

Linux Shell

What is a Shell

- An application running on top of the kernel and provides a powerful interface to the system
- Process user's commands, gather input from user and execute programs
- Types of shell with varied features
 - sh
 - csh
 - ksh
 - bash
 - tcsh

Shell Comparison

| Software | sh | csch | ksh | bash | tcsh |
|-------------------------|----|------|-----|------|------|
| Programming language | y | y | y | y | y |
| Shell variables | y | y | y | y | y |
| Command alias | n | y | y | y | y |
| Command history | n | y | y | y | y |
| Filename autocompletion | n | y* | y* | y | y |
| Command line editing | n | n | y* | y | y |
| Job control | n | y | y | y | y |

*: not by default

<http://www.cis.rit.edu/class/simg211/unixintro/Shell.html>

Shell Comparison

| Software | sh | csH | ksh | bash | tcsh |
|-------------------------|----|-----|-----|-------------|------|
| Programming language | y | y | y | y | y |
| Shell variables | y | y | y | y | y |
| Command alias | n | y | y | y | y |
| Command history | n | y | y | y | y |
| Filename autocompletion | n | y* | y* | y | y |
| Command line editing | n | n | y* | y | y |
| Job control | n | y | y | y | y |

*: not by default

<http://www.cis.rit.edu/class/simg211/unixintro/Shell.html>

Bash Session

```

mars@marsmain ~ $ pwd
/home/mars
mars@marsmain ~ $ cd /usr/portage/app-shells/bash
mars@marsmain /usr/portage/app-shells/bash $ ls -al
total 130
drwxr-xr-x  3 portage portage 1024 Jul 25 10:06 .
drwxr-xr-x 33 portage portage 1024 Aug  7 22:39 ..
-rw-r--r--  1 root  root    35808 Jul 25 10:06 ChangeLog
-rw-r--r--  1 root  root    27002 Jul 25 10:06 Manifest
-rw-r--r--  1 portage portage 4645 Mar 23 21:37 bash-3.1_p17.ebuild
-rw-r--r--  1 portage portage 5977 Mar 23 21:37 bash-3.2_p39.ebuild
-rw-r--r--  1 portage portage 6151 Apr  5 14:37 bash-3.2_p48-r1.ebuild
-rw-r--r--  1 portage portage 5988 Mar 23 21:37 bash-3.2_p48.ebuild
-rw-r--r--  1 portage portage 5643 Apr  5 14:37 bash-4.0_p10-r1.ebuild
-rw-r--r--  1 portage portage 6230 Apr  5 14:37 bash-4.0_p10.ebuild
-rw-r--r--  1 portage portage 5648 Apr 14 05:52 bash-4.0_p17-r1.ebuild
-rw-r--r--  1 portage portage 5532 Apr  8 10:21 bash-4.0_p17.ebuild
-rw-r--r--  1 portage portage 5660 May 30 03:35 bash-4.0_p24.ebuild
-rw-r--r--  1 root  root    5660 Jul 25 09:43 bash-4.0_p28.ebuild
drwxr-xr-x  2 portage portage 2048 May 30 03:35 files
-rw-r--r--  1 portage portage 468 Feb  9 04:35 metadata.xml
mars@marsmain /usr/portage/app-shells/bash $ cat metadata.xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE pkgmetadata SYSTEM "http://www.gentoo.org/dtd/metadata.dtd">
<pkgmetadata>
  <herd>base-system</herd>
  <use>
    <flag name='bashlogger'>Log ALL commands typed into bash; should ONLY be
      used in restricted environments such as honeypots</flag>
    <flag name='net'>Enable /dev/tcp/host/port redirection</flag>
    <flag name='plugins'>Add support for loading builtins at runtime via
      'enable'</flag>
  </use>
</pkgmetadata>
mars@marsmain /usr/portage/app-shells/bash $ sudo /etc/init.d/bluetooth status
Password:
* status: started
mars@marsmain /usr/portage/app-shells/bash $ ping -q -c1 en.wikipedia.org
PING rr.esams.wikimedia.org (91.198.174.2) 56(84) bytes of data:

--- rr.esams.wikimedia.org ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 2ms
rtt min/avg/max/mdev = 49.820/49.820/49.820/0.000 ms
mars@marsmain /usr/portage/app-shells/bash $ grep -i /dev/sda /etc/fstab | cut --fields=--3
/dev/sda1      /boot
/dev/sda2      none
/dev/sda3      /
mars@marsmain /usr/portage/app-shells/bash $ date
Sat Aug  8 02:42:24 MSD 2009
mars@marsmain /usr/portage/app-shells/bash $ lsmod
Module              Size  Used by
rndis_wlan          23424   0
rndis_host           8696   1 rndis_wlan
cdc_ether            5672   1 rndis_host
usbnet              18688   3 rndis_wlan,rndis_host,cdc_ether
parport_pc          38424   0
fglrx               2388128 20
parport             39648   1 parport_pc
iTCO_wdt            12272   0
i2c_i801             9380   0
mars@marsmain /usr/portage/app-shells/bash $

```

What can you do with a shell?

- Check the current shell
 - `echo $SHELL`
- List available shells on the system
 - `cat /etc/shells`
- Change to another shell
 - `exec sh`
- Date and time
 - `date`
- `wget`: get online files
 - `wget https://ftp.gnu.org/gnu/gcc/gcc-7.1.0/gcc-7.1.0.tar.gz`
- Compile and run applications
 - `gcc hello.c -o hello`
 - `./hello`

Think about this:

- Why not just use graphic user interface (GUI) instead of the command line interface in shell?
- A certain type of shell is the same across all Linux distributions
- Command line interface is fast and stable

Linux Applications

- GNU compilers, e.g., gcc, gfortran
- OpenOffice
- Editors, e.g., vim, emacs
- parallel
- wget
- cat, ls, cp
-

<https://directory.fsf.org/wiki/GNU>

Roadmap

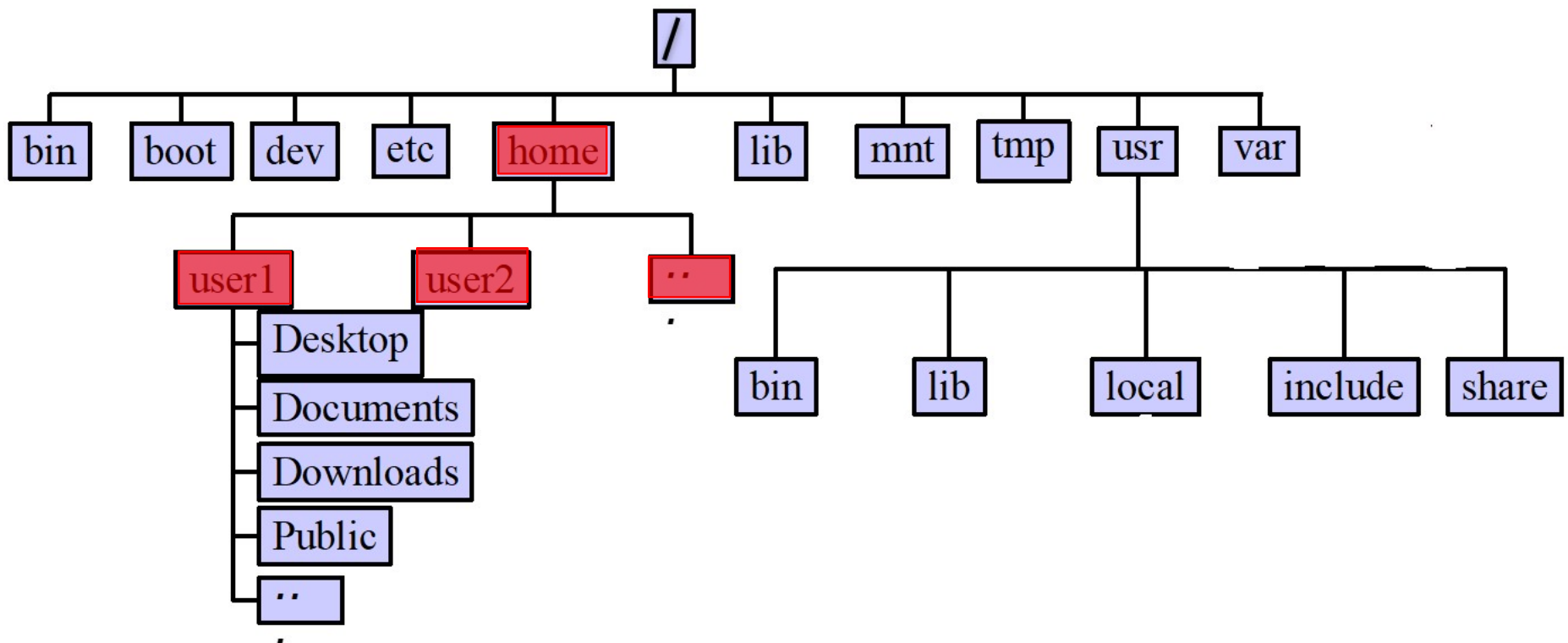
- What is Linux
- Linux file system
- Basic commands
- File permissions
- Variables
- Use HPC clusters
- Processes and jobs
- File editing

Files

- All data in Linux/UNIX are organized in a file format
- A file is a collection of data created by users, system admins...
 - Documents composed of ascii text
 - Program written in high level programming languages
 - Executables, such as programs
 - Directory containing information about its content

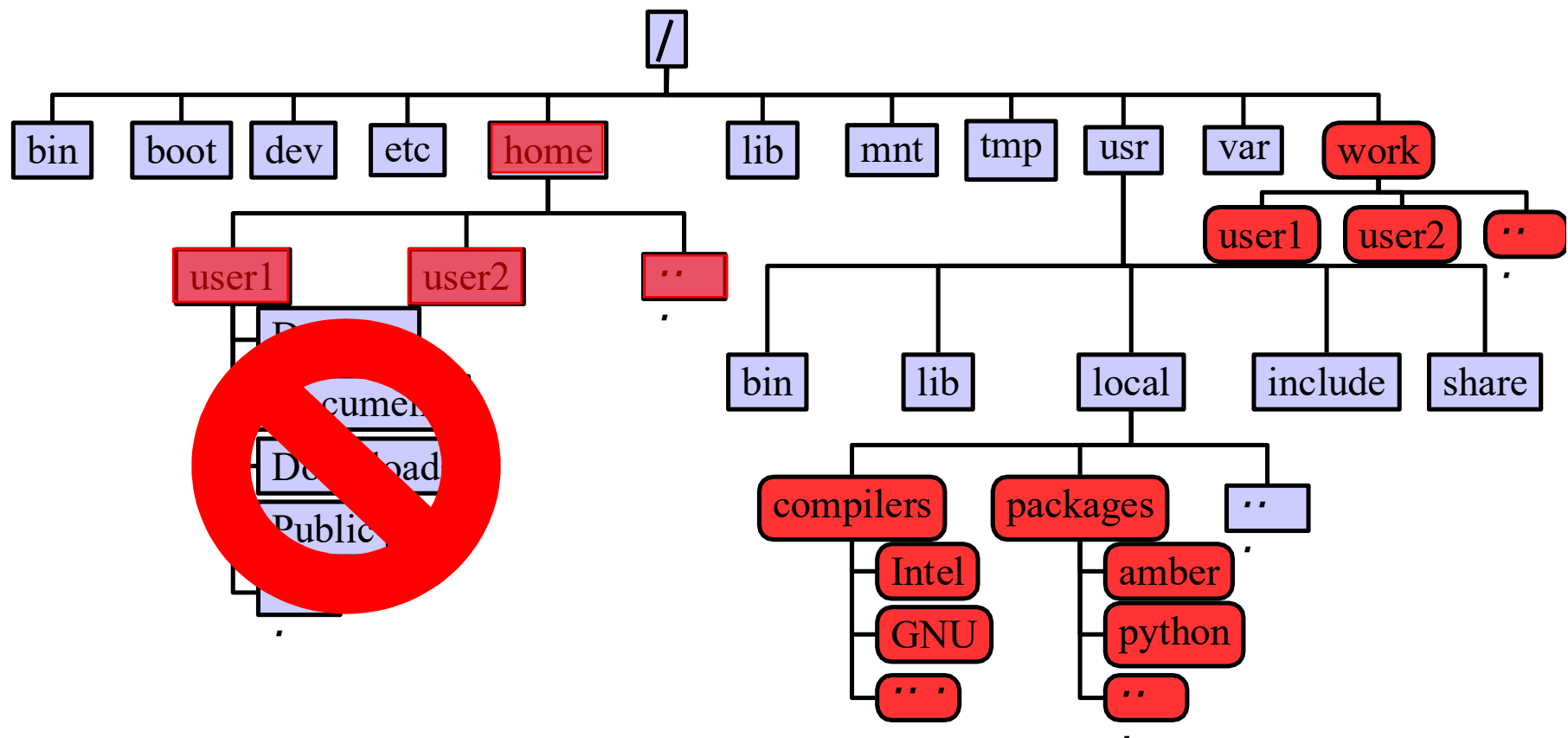
File Directory Structure - PC

- ❖ All files are arranged in directories.
- ❖ These directories are organized into the file system



File Directory Structure - HPC

- ❖ All files are arranged in directories.
- ❖ These directories are organized into the file system



Important Directories

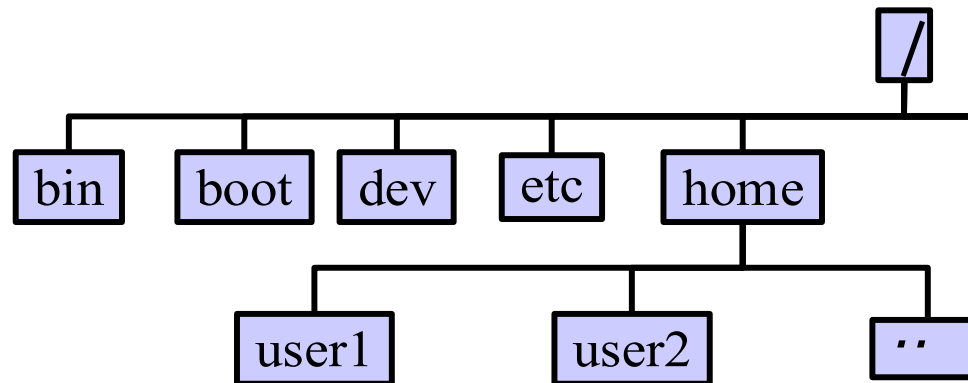
| | |
|-------------|---|
| /bin | contains files that are essential for system operation, available for use by all users. |
| /lib,/lib64 | contains libraries that are essential for system operation, available for use by all users. |
| /var | used to store files which change frequently (system level not user level) |
| /etc | contains various system configurations |
| /dev | contains various devices such as hard disk, CD-ROM drive etc |
| /sbin | same as bin but only accessible by root |
| /tmp | temporary file storage |
| /boot | contains bootable kernel and bootloader |
| /usr | contains user documentations, binaries, libraries etc |
| /home | contains home directories of all users. This is the directory where you are at when you login to a Linux/UNIX system. |

File Path

Definition: position/address in the directory tree

- Absolute path
 - Uniquely defined and does NOT depend on the current path
 - `/tmp` is unique
- Relative path
 - Depend on the current location in the directory tree
 - `.` is the current working directory
 - `..` is one directory up
 - `../tmp` is not unique

File Path Examples



Linux is case sensitive

- All names are case sensitive
 - Commands, variables, files etc.
- Example: MyFile.txt, myfile.txt, MYFILE.TXT are three different files in Linux

Roadmap

- What is Linux
- Linux file system
- **Basic commands**
- File permissions
- Variables
- Use HPC clusters
- Processes and jobs
- File editing

Basic Commands

- **Command format:** `command_name [options] arguments`
`ls -l /home/$USER`
- **Command prompt:** a sequence of characters used in a command line interface to indicate the readiness to accept commands
 - Prompt user to take action
 - A prompt usually ends with one of the characters `$,%#,:,>` and often includes information such as user name and the current working directory
 - The format can be changed via `$PS1`

```
[wfeinste@mike5 ~]$ echo $PS1
[\u@\h \W]\$
[wfeinste@mike5 ~]$
```


Get More Information

- **man**: show the manual for a command or program
 - The manual shows how to use the command and list the different options and arguments
 - **Usage:** `man <command name>`
 - **Example:** `man ls`
- **apropos**: show all of the man pages that may be relevant to a certain command or topic
 - **Usage:** `apropos <string>`
 - **Example:** `apropos editor`
- **info**: More information of a command
 - `info ls`

Commands: pwd and cd

- **pwd**: print the current working directory
 - Usage: `pwd`
 - Example: `pwd`
- **cd**: allow one to change the current working directory
 - Usage: `cd [destination]`
 - Example: `cd /tmp`
 - The default destination is the home directory if `[destination]` is omitted
 - `~` stands for home directory (bash)

Commands: ls

- **ls** command list the contents of a directory
 - Usage: **ls** <options> <path>
 - Example: **ls**
 - The default path will be current working directory if `path` is omitted.
- Options
 - **-l**: show long listing format
 - **-a**: (**--all**) show hidden files(name starts with an "." is hidden)
 - **-r**: reverse order when sorting
 - **-t**: show modification times
 - **-h**: (**--human-readable**) use file sized in SI units (bytes, kbytes, megabytes etc.)
 - **-d**: (**--directory**) list directory entries instead of contents, and do not dereference symbolic links. (e.g. **ls -d */**)

Commands: cat, more/less, head/tail

- Display the content of a file to screen
 - **cat**: show content of a file
 - **more**: display contents one page at a time
 - **less**: display contents one page at a time, and allow forward/backward scrolling
- Usage: `cat/more/less <options> <filename>`
- **head**: output the first part of files
- **tail**: output the last part of files
- Usage: `head/tail <options> <filename>`
- Be careful when using those commands on binary files
 - **file**: reveal what type of file the target is

Short Summary

- Get the current working directory and change directory
 - `pwd`
 - `cd`
- **ls** List the contents of a directory
 - `-l`: show long listing format
 - `-r`: reverse order when sorting
 - `-t`: show modification times
 - `-h`: (`--human-readable`) use file sized in SI units (bytes, kbytes, megabytes etc.)
- Display the content of a file to screen
 - `cat`
 - `less/more`

Auto-completion

- Allows automatic completion of typing file, directory or command name via the TAB key
 - Convenient, also error-proof
 - If there is no unique name, all matching names will show
- The default feature in `bash` and `tcsh`
- Example: if your home directory contains directories: `Desktop`, `Documents` and `Downloads`
 - Enter command `ls D`, then press tab
 - Enter command `ls Do`, then press tab
 - Enter command `ls Dow`, then press tab

Wildcards

Linux allows the use of wildcards for strings

- *: any number of characters
 - Example: `ls *.gz` will list all the file ending with .gz
- ?: any *single* character
- [: specify a range
e.g.: `ls test[1-9]` list the file test1,test2 ...

Hidden file

In Linux, a hidden file is any file that begins with a "." (dot). When a file is hidden it can not be seen with the bare `ls` command.

- `ls -a`: (--all) show hidden files
 - Example: `ls -a` will list all the file in the directory

Commands: mkdir

- **mkdir**: create a directory
- Usage: **mkdir** <options> <path>
- Example: `mkdir ~/testdir`
- By default, the directory is created in the current directory
- Options
 - p**: create the target directory as well as any directories that appear in the path but does not exist

Commands: cp

- **cp**: copy a file or directory
- Usage: **cp** <options> <sources> <destination>
- Example: `cp $HOME/.bashrc ~/testdir`
- Options
 - **-r**: copy recursively, required when copying directories.
 - **-i**: prompt if file exists on destination and can be copied over.
 - **-p**: preserve file access times, ownership etc.
- By default **cp** will overwrite files with identical names (!!)
- If there are more than one source files, then the destination must be a directory

Commands: rm

- **rm**: removes files and directories
- Usage: **rm** <options> <list of files/dirs>
- Examples: `rm testdir/.bashrc ~/testfile`
- Options
 - `-r`: remove recursively, required when deleting directories
 - `-i`: prompt if the file really needs to be deleted
 - `-f`: force remove (override the `-i` option)
- BE CAREFUL: DELETED FILES ***CANNOT*** BE RECOVERED!!!

Commands: mv

- **mv** : moves or renames a file or directory
- **Usage:** `mv <options> <sources> <dest>`
- **Example:** `mv test test1`
- Use the `-i` option to prompt if a file or directory will be overwritten.
- If there are more than one source files, the destination must be a directory

Commands: alias

- **alias**: create a shortcut to another command or name to execute a long string
- Usage
 - bash/sh/ksh: `alias <name>="<actual command>"`
 - csh/tcsh: `alias <name> "<actual command>"`
- Example
 - bash/sh/ksh: `alias lla="ls -altr"`
 - csh/tcsh: `alias lls "ls -altr"`
- **alias** can be used to prevent files from being deleted accidentally
 - Example: `alias rm "rm -i"`
- **alias**: list all aliases currently defined (without arguments)
- **unalias**: remove an alias

Commands: history

- **history**: print out the bash history of the current user to the screen
- Usage
 - `history`
- Example
 - `history 30`
 - `history | grep qsub`

Roadmap

- What is Linux
- Linux file system
- Basic commands
- **File permissions**
- Variables
- Use HPC clusters
- Processes and jobs
- File editing

File Permission

```
weis-MacBook-Pro:Laplace2D wei$ ll -al
```

```
total 56
```

| | | | | | | | | |
|------------|----|-----|-------|------|-----|----|-------|----------------------|
| drwxr--r-- | 9 | wei | staff | 306 | Jun | 13 | 16:13 | . |
| drwxr-xr-x | 12 | wei | staff | 408 | May | 4 | 17:09 | .. |
| -rwxr--r-- | 1 | wei | staff | 1614 | May | 4 | 17:09 | README |
| -rwxr--r-- | 1 | wei | staff | 2083 | May | 4 | 17:09 | answer.c |
| -rwxr--r-- | 1 | wei | staff | 1945 | May | 4 | 17:09 | answer.f90 |
| -rwxr--r-- | 1 | wei | staff | 2004 | May | 4 | 17:09 | exercise.c |
| -rwxr--r-- | 1 | wei | staff | 1833 | May | 4 | 17:09 | exercise.f90 |
| lrwxr-xr-x | 1 | wei | staff | 10 | Jun | 13 | 16:13 | linkMe -> exercise.c |
| -rwxr--r-- | 1 | wei | staff | 1572 | May | 4 | 17:09 | timer.h |

Linux File Permission

- Designed as the multi user environment, the access restriction of files to other users on the system is embedded.
- Three types of file permission
 - Read (r)
 - Write (w)
 - Execute (x)
- Three types of user
 - User (u) (owner)
 - Group (g) (group members)
 - World (o) (everyone else on the system)

Linux File Permission

Each file in Linux has the following attributes:

- Owner permissions: determine what actions the owner of the file can perform on a file
- Group permissions: determine what actions a user, who is a member of the group that a file belongs to, can perform on a file
- Other (world) permissions: indicate what action all other users can perform on a file

File Permission

```
weis-MacBook-Pro:Laplace2D wei$ ll -al
total 56
drwxr--r--  9 wei  staff   306 Jun 13 16:13 .
drwxr-xr-x 12 wei  staff   408 May  4 17:09 ..
-rwxr--r--  1 wei  staff  1614 May  4 17:09 README
-rwxr--r--  1 wei  staff  2083 May  4 17:09 answer.c
-rwxr--r--  1 wei  staff  1945 May  4 17:09 answer.f90
-rwxr--r--  1 wei  staff  2004 May  4 17:09 exercise.c
-rwxr--r--  1 wei  staff  1833 May  4 17:09 exercise.f90
lrwxr-xr-x  1 wei  staff    10 Jun 13 16:13 linkMe -> exercise.c
-rwxr--r--  1 wei  staff  1572 May  4 17:09 timer.h
```

The first column indicates the type of a file/dir/link

- d: for directory
- l: for symbolic link
- - for normal file

File Permission

```
weis-MacBook-Pro:Laplace2D wei$ ll -a
total 56
drwxr--r--  9 wei  staff   306 Jun  1
drwxr-xr-x 12 wei  staff   408 May
-rwxr--r--  1 wei  staff  1614 May
-rwxr--r--  1 wei  staff  2083 May
-rwxr--r--  1 wei  staff  1945 May
-rwxr--r--  1 wei  staff  2004 May
-rwxr--r--  1 wei  staff  1833 May
lrwxr-xr-x  1 wei  staff    10 Jun  1
-rwxr--r--  1 wei  staff  1572 May
```

Owner
(u)

Group
(g)

Others
(o)

Changing File Permission

- **chmod** is a *NIX command to change permissions on a file
- Usage: `chmod <option> <permissions> <file or directory name>`
- `-R`: change permission recursively in a directory
- **chmod in Symbolic Mode:**

| Chmod operator | Description |
|----------------|--|
| + | Adds the designated permission(s) to a file or directory. |
| - | Removes the designated permission(s) from a file or directory. |

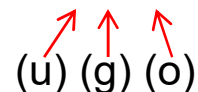
e.g. `chmod u+rwx filename`
`chmod o-w filename`

Changing File Permission

- **chmod in Absolute Mode:**

| Number | Octal Permission Representation | Ref |
|--------|---|-----|
| 0 | No permission | --- |
| 1 | Execute permission | --X |
| 2 | Write permission | -W- |
| 3 | Execute and write permission: 1 (execute) + 2 (write) = 3 | -WX |
| 4 | Read permission | r-- |
| 5 | Read and execute permission: 4 (read) + 1 (execute) = 5 | r-X |
| 6 | Read and write permission: 4 (read) + 2 (write) = 6 | rw- |
| 7 | All permissions: 4 (read) + 2 (write) + 1 (execute) = 7 | rwX |

e.g. `chmod 755 test.txt`



 (u) (g) (o)

Permission Effect on File vs Directory

| Permission | File | Directory |
|------------|-------------------------------------|---|
| r | read the file content | Ls files under the directory |
| w | write to the file | create new files and directories, delete existing files and directories, rename and move the existing files and directories |
| x | execute the file (if executable) | cd into the directory |

User Groups at HPC/LONI

- Users are organized into groups
 - **groups** command to find your group membership
- Group membership makes sharing files with members of a group easy
- Each user is in at least one group and can be in multiple groups
 - Groups in LONI systems:
lsuusers, latechusers, unousers,
ullusers, sususers, tulaneusers,
loniusers, xavierusers
 - Groups in LSU HPC system
Users, Admins, xsede...
 - You are only in one of the above groups due to software licensing

Roadmap

- What is Linux
- Linux file system
- Basic commands
- File permissions
- **Variables**
- Use HPC clusters
- Processes and jobs
- File editing

Variables

- Linux allows the use of variables
 - Similar to programming languages
 - Number, character or string
- Rules for variable names
 - Must start with a letter or underscore
 - Number can be used anywhere else
 - Do not use special characters such as @, #, %, \$
 - Case sensitive
 - Allowed: VARIABLE, VAR1234able, var_name, _VAR
 - Not allowed: 1var, %name, \$myvar, var@NAME
- Two types of variables:
 - Global variables (ENVIRONMENT variables)
 - Local variables (user defined variables)

Global Variables

- Environment variables provide a simple way to share configuration settings between multiple applications and processes in Linux
 - Using all uppercase letters
 - Example: `PATH`, `LD_LIBRARY_PATH`, `DISPLAY` etc.
- To reference a variable, prepend `$` to the name of the variable
- Example: `$PATH`, `$LD_LIBRARY_PATH`, `$DISPLAY` etc.
- `printenv`/`env` list the current environmental variables in your system.

List of Some Environment Variables

| | |
|-----------------|---|
| PATH | A list of directory paths which will be searched when a command is issued |
| LD_LIBRARY_PATH | colon-separated set of directories where libraries should be searched for first |
| HOME | indicate where a user's home directory is located in the file system. |
| PWD | contains path to current working directory. |
| OLDPWD | contains path to previous working directory. |
| TERM | specifies the type of computer terminal or terminal emulator being used |
| SHELL | contains name of the running, interactive shell. |
| PS1 | default command prompt |
| PS2 | Secondary command prompt |
| HOSTNAME | The systems host name |
| USER | Current logged in user's name |
| DISPLAY | Network name of the X11 display to connect to, if available. |

Editing Variables

- Assign values to variables

| Type | sh/ksh/bash | csch/tcsch |
|----------------------|-------------------|-------------------|
| Shell (local) | name=value | set name=value |
| Environment (global) | export name=value | setenv name value |

- Shell variables is only valid within the current shell, while environment variables are valid for all subsequently opened shells.
- Example: useful when running a script, where exported variables (global) at the terminal can be inherited within the script.
 - \$ export v1=one
 - \$ bash
 - \$ echo \$v1 → one

Editing Variables at the Curent Login

Example: to add a directory to the PATH variable

```
sh/ksh/bash: export PATH=/path/to/executable:${PATH}
```

```
csh/tcsh: setenv PATH /path/to/executable:${PATH}
```

- Path order matters, first in line takes a higher priority

Persistent variables for each login

- No change at each login
- Make setting changes available in both login and non-login shells
- Define these variables in the ~/.bashrc file.
 - vim ~/.bashrc
 - source ~/.bashrc

Input & Output Commands

The basis I/O statements:

- **echo**: display info to screen
 - The `echo arguments` command will print arguments to screen or standard output, where `arguments` can be a single or multiple variables, string or numbers
- **read**: reading input from screen/keyboard/prompt
 - The `read` statement takes all characters typed until the Enter key is pressed
 - Usage: `read <variable name>`
 - Example: `read name`

Input & Output Commands

- Examples
 - `echo "hello !"`
 - `hello`
- By default, `echo` eliminates redundant whitespaces (multiple spaces and tabs) and replaces it with a single whitespace between arguments.
- To include redundant whitespace, enclose the arguments within double quotes

Other Useful Commands

| | |
|-----------------|--|
| passwd | Change password (does not work on LSU HPC and LONI systems) |
| chsh | Change default shell (does not work on LSU HPC and LONI systems) |
| df | Report disk space usage by filesystem |
| du | Estimate file space usage - space used under a particular directory or files on a file system. |
| sudo | Run command as root (only if you have access) |
| mount | Mount file system (root only) |
| umount | Unmount file system (root only) |
| shutdown | Reboot or turn off machine (root only) |
| top | Produces an ordered list of running processes |
| free | Display amount of free and used memory in the system |
| find | Find a file |
| alias | enables replacement of a word by another string |

Other Useful Commands

| | |
|--------------|---|
| vi | Edit a file using VI/VIM |
| emacs | Edit a file using Emacs |
| file | Determine file type |
| wc | Count words, lines and characters in a file <code>wc -l file</code> |
| grep | Find patterns in a file <code>grep alias file</code> |
| awk | File processing and report generating <code>awk '{print \$1}' file</code> |
| sed | Stream Editor <code>sed 's/home/HOME/g' file</code> |
| set | manipulate environment variables <code>set -o emacs</code> |
| touch | change file timestamps or create file if not present |
| date | display or set date and time |
| which | Location of a command |

Other Useful Commands

| | |
|----------------|---|
| ln | Link a file to another file <code>ln -s file1 file2</code> |
| wait | Wait for each specified process and return its termination status. |
| which | Shows the full path of (shell) commands |
| who | Show who is logged on |
| w | Another command to show who is logged in |
| whoami | Print effective userid |
| finger | User information lookup program |
| whatis | Display manual page descriptions |
| history | Display the command history list with line numbers. An argument of <code>n</code> lists only the last <code>n</code> lines. |

❑ `man command`: learn more about these commands

Roadmap

- What is Linux
- Linux file system
- Basic commands
- File permissions
- Variables
- **Use HPC clusters**
- Processes and jobs
- File editing

Login Remote Systems

- Most Linux systems allocate secure shell connections from other systems
- Log in using the **ssh** command to the LSU HPC and LONI clusters
- Usage: `ssh <username>@<remote host name>`
 - Example: `ssh user@mike.hpc.lsu.edu`
- **-X** option: forward the display of an application
- The default port is 22 for `ssh`
 - `ssh -p <port number> <username>@<hostname>`

File Transfer between Two Systems

- **scp** : copy files between two hosts over the ssh protocol
- Usage:
 - `scp <options> <user>@<host>:/path/to/source`
`<user>@<host>:/path/to/destination`
- If the user name is the same on both systems, omit `<user@>`
- If transferring files from or to localhost, `<user>@<host>:` option can be omitted
- Options are `-r` and `-p`, same meaning with `cp`
- Examples
 - `scp user@mike.hpc.lsu.edu:/home/user/file1 .`
 - `scp -r file user@mike.loni.org:/home/user`

File Transfer between Two Systems

- **rsync** is another utility for file transferring
- Usage: `rsync <options> <source> <destination>`
- Delta-transfer algorithm
 - Only transfer the bits that are different between source and destination
- Widely used for backups and mirroring as an improved copy command for everyday use
- Command options
 - -a: archive mode
 - -r: recursive mode
 - -v: increase verbosity
 - -z: compress files during transfer
 - -u: skip files that are newer on the receiver
 - -t: preserve modification times

From a download link on a website (usually opened with a web browser)

- From a download link on a website (usually opened with a web browser)
 - Right click on the link and then copy the link location
 - `wget <paste_your_copied_link_here>`

Compressing and Archiving a single file

Reduce storage usage or bandwidth while transferring files.

- **Compress:** `gzip`, `zip`, `bzip2`
- **Decompress:** `gunzip`, `unzip`, `bunzip2`
- Options
 - Recursively, use the `-r` option
 - Overwrite files while compressing/uncompressing, use the `-f` option
- By convention
 - gzipped files: `.gz`, `.z` or `.Z`
 - zipped files: `.Zip` or `.zip`
 - bzipped files: `.bz2` or `.bz`

Compressing and Archiving Files

- **tar**: create and manipulate streaming archived files.
- Usage: **tar** **<options>** **<file>** **<path>**
 - **<file>** tar archived file, usually with extension .tar
 - **<path>** files/directories being archived
- Common options
 - -c: create/compress an archive file
 - -x: extract/decompress an archive file
 - -t: list contents of archive (for testing)
 - -z: filter the archive through gzip
 - -j: filter the archive through bzip2
 - -f: archive
 - -v: verbosely list files processed

Compressing and Archiving Files

tar: create and manipulate streaming archived files.

Examples:

- File compressing
 - `tar czvf file.tgz ${HOME}/*`
 - `tar cjvf file.tbz2 ${HOME}/*`
- File decompressing
 - `tar xzvf file.tgz -C [dest]`
 - `tar xjvf file.tbz2 -C [dest]`
- File listing for testing
 - `tar tvf file.tgz`

Pipes

- Pipe commands: connect two or more commands together using “|”
- **grep**: searches certain patterns from a file(s)


```
cat file | grep [option] pattern
```

```
history | grep qsub
```

| Option | Description |
|--------|---|
| -v | Print all lines not match pattern |
| -n | Print the matched line and line number |
| -l | Print only the names of files with matching pattern |
| -i | Match either upper- or lowercase. |
| -c | Print the count of matching lines |

Pipes: sort, wc, more, less

- `sort`: arranges lines of text alphabetically or numerically
- `ls | sort -k2`

| Option | Description |
|--------|---------------------------|
| -n | Sort numerically |
| -r | Reverse the order of sort |
| -k | Sort by a certain column |
| -t | Field separator |

- `ls | wc`
- `cat file | more`
- `Cat file | less`

I/O Redirection

- Three file descriptors for I/O streams (everything is a file in Linux)



- I/O redirection allows users to connect applications
 - <: a file to STDIN as input
 - >: save STDOUT to a file
 - >>: append STDOUT to a file

I/O Redirection Examples

- Write STDOUT to file:
`ls -l > ls.out`
- Write STDERR to file:
`ls -l 2 > ls.err`
- Write STDERR and STDOUT to one file:
`ls -l > output 2>&1`
- Discard STDOUT and STDERR:
`command > /dev/null 2>&1`
- Discard STDOUT but write STDERR to file
`command > /dev/null 2 > command.err`

Roadmap

- What is Linux
- Linux file system
- Basic commands
- File permissions
- Variables
- Use HPC clusters
- **Processes and jobs**
- File editing

Processes

- Process: an instance of a running program
- Linux create and start a new process (PID) for each command
 - **ps** or **top**
- A process can be run in :
 - **Foreground**: the command prompt is not returned until the current process has finished executing.
 - **Background**: the command prompt back to do some other useful work e.g. `ls -l &`

Processes and Jobs

- Two ways to send a job into the background:
 1. **command &**
 2. suspend a running job using **Ctrl-z** and **bg**.
- When a process is running in background or suspended, it will be entered on to a list along with a job number (not PID) **jobs -l**
- Restart a suspended job to foreground
fg %<job number>

Managing Processes and Jobs

- To kill or terminate a process:
 - Job running in foreground: `Ctrl-c`
 - Job whose job ID you know: `kill %<job number>`
 - Job whose PID you know: `kill <PID>`

Roadmap

- What is Linux
- Linux file system
- Basic commands
- File permissions
- Variables
- Use HPC clusters
- Processes and jobs
- File editing

File Editing

- Most commonly used editors on Linux/Unix systems
 - vi or vim (vi improved)
 - Emacs
- vi/vim is installed by default on Linux/Unix systems and has only a command line interface (CLI).
- Emacs has both a command line interface (CLI) and a graphical user interface (GUI).
 - use `emacs -nw` to open file in console
- Other editors: nano, pico, kate, gedit, gvim, kwrite, nedit

File Editing (vi)

vi works in two modes:

- Command mode
 - This is the mode when entering vi
 - Commands can be issued at the bottom of the screen, e.g. copy, paste, search, replace etc.
 - Press “i” to enter editing mode
- Editing mode
 - Text can be entered in this mode
 - Press “esc” key to go back to the command mode

Most used commands (vi)

| Description | Command |
|---------------------------------|--------------|
| Insert at cursor | i |
| Insert at the beginning of line | I |
| Delete a line | dd |
| Copy a line | yy |
| Paste | p |
| Search forward | /pattern |
| Search backward | ?pattern |
| Search again | n |
| Go to line #n | :n |
| Replace text | %s/new/old/g |
| Save and exit | :wq |
| Exit without save | :q |

Editor cheatsheet (1)

0 Cursor Movement

- move left
- move down
- move up
- move right
- jump to beginning of line
- jump to end of line
- goto line *n*
- goto top of file
- goto end of file
- move one page up
- move one page down

vi

- h
- j
- k
- l
- 0
- \$
- nG
- 1G
- G
- C-u
- C-d

emacs

- C-b
- C-n
- C-p
- C-f
- C-a
- C-e
- M-x goto-line ← *n*
- M-<
- M->
- M-v
- C-v

Editor cheatsheet (2)

File Manipulation

- save file
- save file and exit
- quit
- quit without saving
- delete a line
- delete *n* lines
- paste deleted line after cursor
- paste before cursor
- undo edit
- delete from cursor to end of line
- search forward for *patt*
- search backward for *patt*
- search again forward (backward)

vi

- :w
- :wq, ZZ
- :q
- :q!
- dd
- *n*dd
- p
- P
- u
- D
- \patt
- ?patt
- n

emacs

- C-x C-s
-
- C-x C-c
-
- C-a C-k
- C-a M-n C-k
- C-y
-
- C-_
- C-k
- C-s *patt*
- C-r *patt*
- C-s (r)

Editor cheatsheet (3)

File Manipulation (contd)

- replace a character
- join next line to current
- change a line
- change a word
- change to end of line
- delete a character
- delete a word
- edit/open file *file*
- insert file *file*
- split window horizontally
- split window vertically
- switch windows

vi

- r
- J
- cc
- cw
- c\$
- x
- dw
- :e *file*
- :r *file*
- :split or C-ws
- :vsplit or C-wv
- C-ww

emacs

-
-
-
-
-
- C-d
- M-d
- C-x C-f *file*
- C-x i *file*
- C-x 2
- C-x 3
- C-x o

Shell Scripts

- Script: a program written for a software environment to automate execution of tasks
 - A series of shell commands put together in a file
 - When the script is executed, those commands will be executed one line at a time automatically
- The majority of script programs are “quick and dirty”, where the main goal is to get the program written quickly
 - May not be as efficient as programs written in C and Fortran

Script Example (~/.bashrc)

```
# .bashrc
```

```
# Source global definitions
```

```
if [ -f /etc/bashrc ]; then
```

```
    . /etc/bashrc
```

```
fi
```

```
# User specific aliases and functions
```

```
export PATH=$HOME/packages/eFindsite/bin:$PATH
```

```
export LD_LIBRARY_PATH=$HOME/packages/eFindsite/lib:$LD_LIBRARY_PATH
```

```
alias qsubI="qsub -I -X -l nodes=1:ppn=20 -l walltime=01:00:00 -A  
my_allocation"
```

```
alias lh="ls -altrh"
```

Take-home message

- Why we HPC is offering Linux training. Why you should learn Linux.
- Linux System Architecture
 - Kernel, shell, application
 - Multi-user environment
- Linux file system
 - /home directory
 - /work directory on HPC
 - File installation path
- Basic commands (pwd, ls, cat...), auto-completion and wild cards
- File permissions: read, write and exe for owner, group and others
- Variables
 - ENVIRONMENT VARIABLES
- Vim editor

Next Week Training

- **HPC User Environment 1, February 5**
 - Overview of the HPC hardware and software environment
- **Weekly trainings during regular semester**
 - Wednesdays “9:00am-11:00am” session, Frey 307 CSC
- **Programming/Parallel Programming workshops**
 - Usually in summer
- **Keep an eye on our webpage:**
<http://www.hpc.lsu.edu/training/tutorials.php#upcoming>

Getting Help

- User Guides
 - LSU HPC: <http://www.hpc.lsu.edu/docs/guides.php#hpc>
 - LONI: <http://www.hpc.lsu.edu/docs/guides.php#loni>
- Documentation: <http://www.hpc.lsu.edu/docs>
- Archived HPC training:
<http://www.hpc.lsu.edu/training/archive/tutorials.php>
- Contact us
 - Email ticket system: sys-help@loni.org
 - Telephone Help Desk: 225-578-0900