Center for Computation & Technology

LSU INFORMATION TECHNOLOGY SERVICES

Introduction to Linux for HPC

Kathy Traxler

ktraxler@cct.lsu.edu

Topics

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

Topics

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

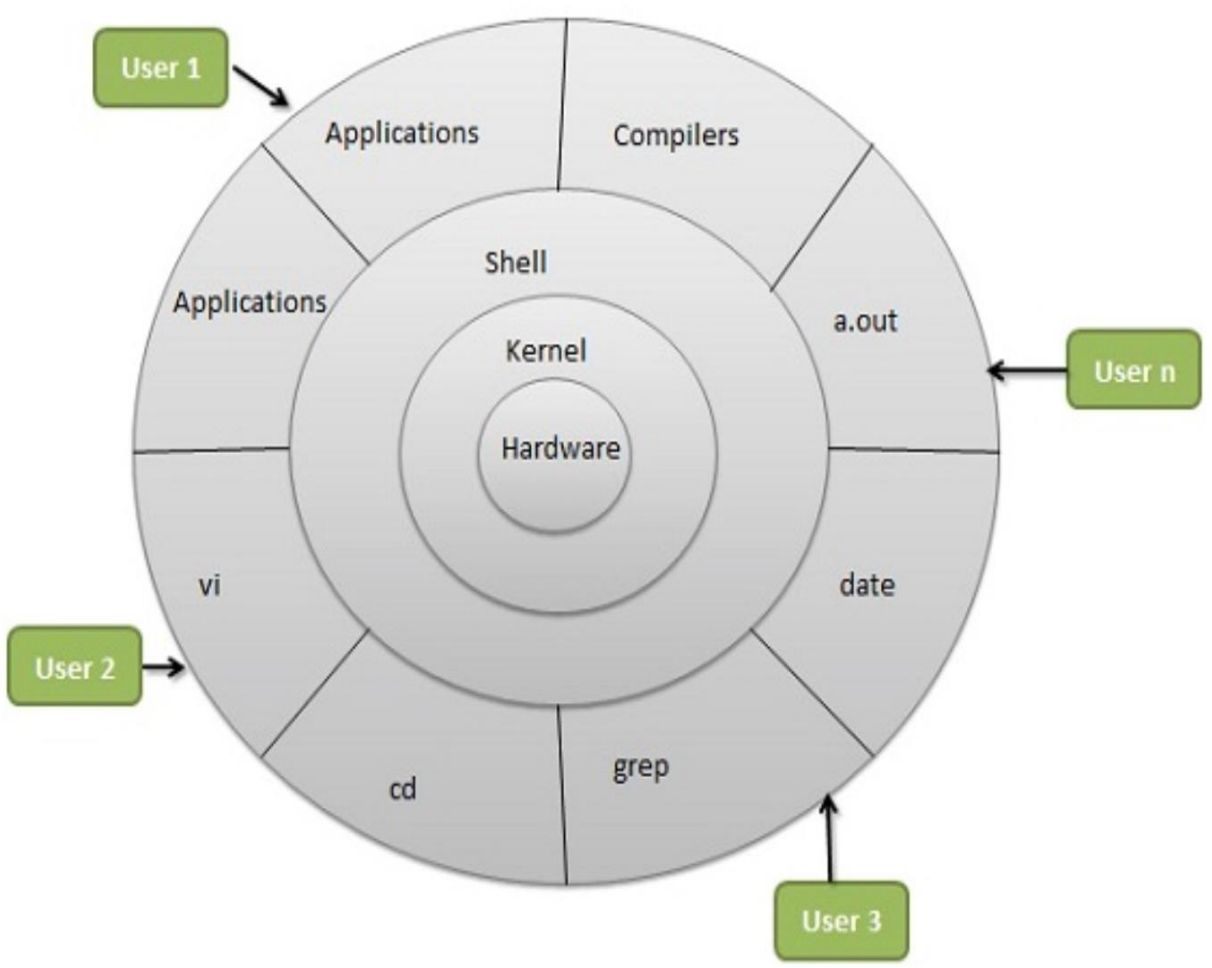
What is Linux

- Linux is an open source operating system based on the UNIX operating system created at ATT Bell Labs
- Today we refer to *nix systems as there are so many Unox like systems out there

What you need to know

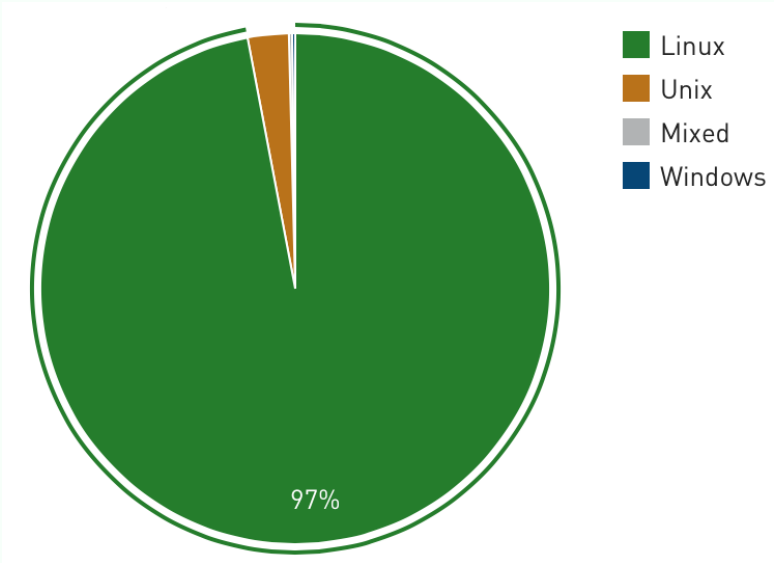
- Linux is a very powerful operating system
- You as a user have many options making Linux more difficult to learn
- To get started you need just a few basic commands that for me have been the same from old DEC Unix systems to AIX to any *nix I have used

Linux System Architecture

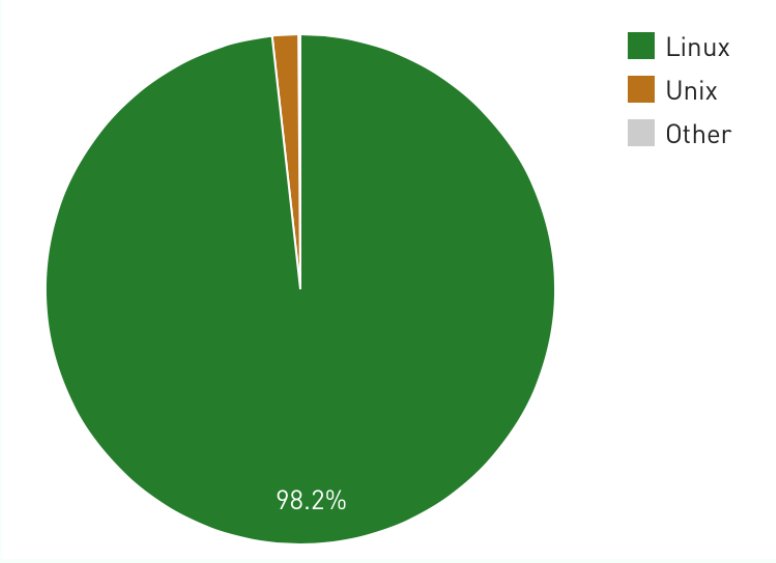


Why Linux for HPC

OS family system share



OS family performance share

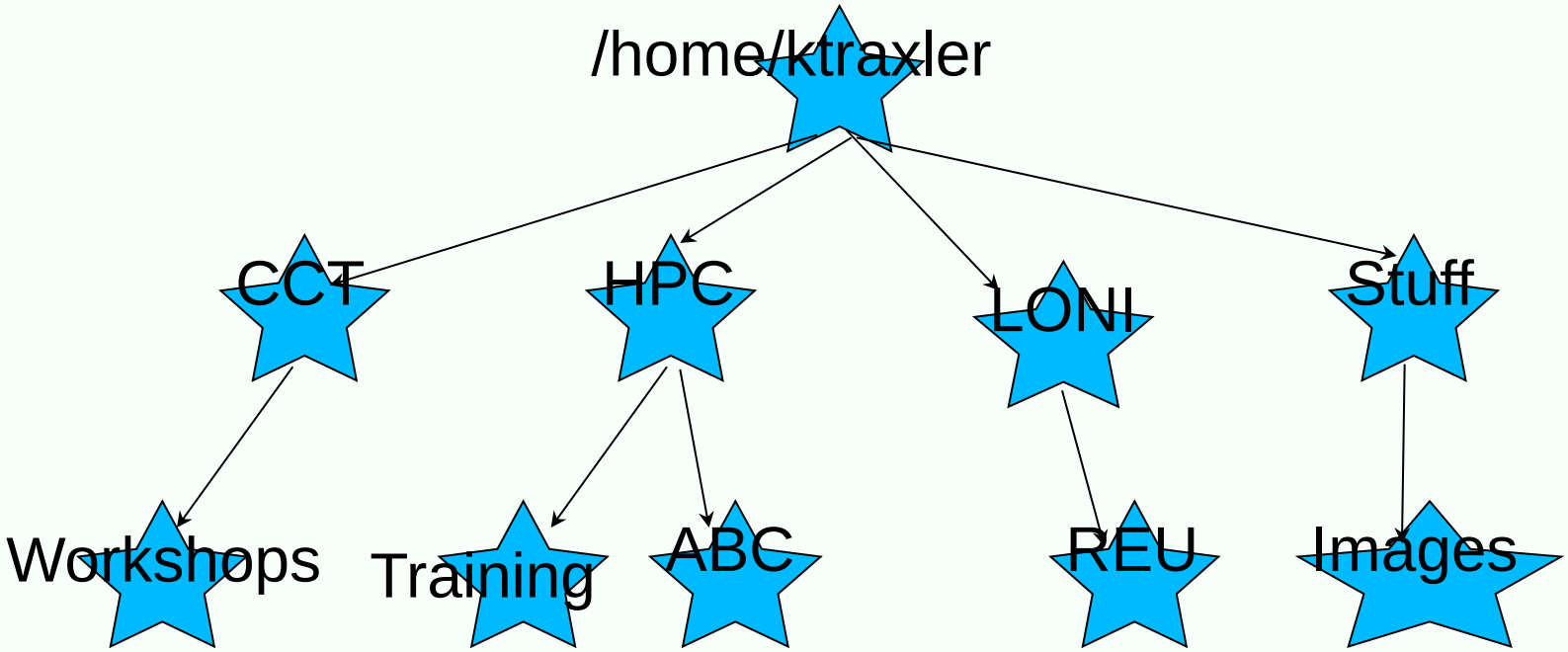


Linux is the most popular OS used in supercomputers

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

File System

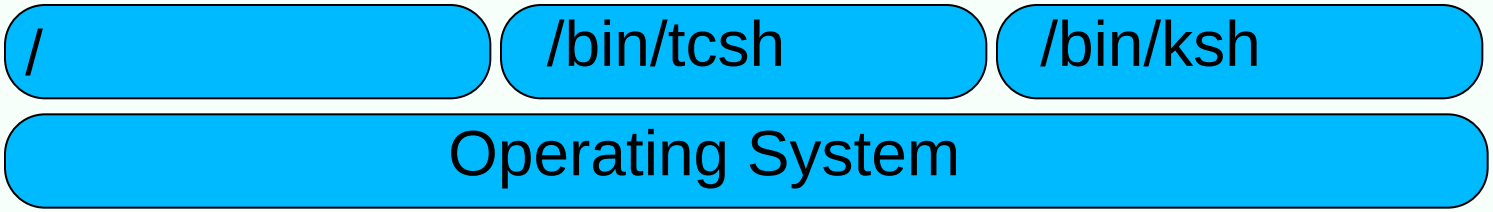
- The *nix file systems are a tree with the root as your home directory
- From your LONI home account you create directories and files
- You navigate down through those files


/home/ktraxler
CCT
HPC
LONI
Stuff
Workshops
Training
ABC
REU
Images

Everything is a FILE!

- Everything in *nix languages is a file
  - input and output streams
  - files
  - directories (you can edit them in vi)
  - volumes

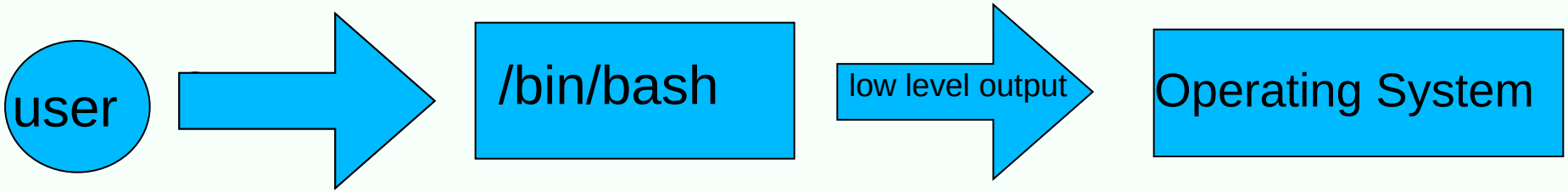
What is the shell?



- The shell is a high level interface to the operating system for users
- This is the “prompt” that you get when you login
- Different shells are preferred by different users, but thay all provide the same access to the underlying OS

How does it work?

- User issues shell command
- Shell sends a lower level command to the OS
- The OS executes the command and returns any results back to the user via the shell



Features of the shell

- All modern, common shells provide typical language constructs (comparison, flow control, etc).
- A set of shell commands and constructs can be saved into a text file and be run as a program; these are called shell scripts
- Shells can track global variables that are referred to as their environment

What is the environment?

- The shell's environment is used to store useful system information;
- This information is stored as "environmental variables"
- Some variables are set when you first login;
- Other variables can be customized by the user using a specfic set of files contained in your HOME directory

Viewing Your Environment

- View the entire environment
  - env | more #this command is common to all shells
- View a specific variable
  - printenv VARNAME #common to all shells
  - echo $VARNAME #common to all shells

Notable environmental variables

- HOME
  - Your home directory
- PATH
  - List of colon delimited paths that should be searched for executables
- EDITOR
  - sets up the path to your preferred editor

Not all shells are created equal

- Different shells behave differently and have different commands for similar functionality
- The 2 common families:
  - Bourne Shell: bash, ksh, zsh
  - C-Shell: csh, tcsh
- Bourne
  - Good as a login shell and as a basis for a program
- C-Shell
  - Good only as a login shell; avoid using it to program

Manipulating the environment

- Creating or modifying a global environmental variable:
  - export VARNAME='value' #bourne shell
  - setenv VARNAME 'value' #c-shell
  - example:
    - export PATH=PATH:/usr/local
    - extra care must be taken when modifying the global variables because this will determine the way your entire shell works!

Environment set up

- When you log in interactively, the system default environment is set using the following files
  - /etc/profile
  - /etc/csh.cshrc
- These files automatically set up the default path and vital user system variables

Customizing the default environment

- /bin/bash users may create the following files in their home directory
  - .bash_profile #runs first after /etc/profile
  - .bashrc #runs after .bash_profile
  - .profile #runs after .bashrc
- /bin/bash users may also create a file that is executed when one logs out
  - .bash_logout
  - .bash_profile must NOT produce any standard out since it will break commands such as rsync

Customizing the default environment

- /bin/tcsh users may create the following files in their home directory
  - .cshrc #runs first after /etc/csh.cshrc

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.

Basic Commands

- Command: a directive to a computer program (interpreter) to perform specific tasks

- 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 be changed via PS1

- Command format: command_name [options] arguments

  ls –l /home/user

Basic Commands

- ls - list all files directories and symbolic links in a given directory
  - at the prompt type: ls and hit return
  - ls -l
    - ktraxler@l1f1n01$ **ls -l**
    - total 24
    - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 prog1
    - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 prog2
    - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 public_html
    - ktraxler@l1f1n01$

ls (2)

- ktraxler@l1f1n01$ **ls -al**
- total 104
- drwxr-sr-x 5 ktraxler sys 512 Oct 10 22:02 .
- drwxr-sr-x 344 sys sys 6144 Oct 10 21:59 ..
- -rw------- 1 ktraxler sys 909 Apr 30 16:45 .bash_history
- -rw-r--r-- 1 ktraxler sys 684 Apr 30 14:38 .bashrc
- -rw-r--r-- 1 ktraxler sys 207 Apr 30 14:25 .soft
- -rw-r--r-- 1 ktraxler sys 4569 Oct 10 22:02 .soft.cache.csh
- -rw-r--r-- 1 ktraxler sys 4609 Oct 10 22:02 .soft.cache.sh
- drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 prog1
- drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 prog2
- drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 public_html
- ktraxler@l1f1n01$

Make a directory

- mkdir
  - mkdir path/newdir
  - creates a directory named “newdir” in the path “path”
    - ktraxler@l1f1n01$ **ls -l**
    - total 24
    - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 prog1
    - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 prog2
    - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 public_html
    - ktraxler@l1f1n01$ **mkdir prog3**
    - ktraxler@l1f1n01$ **ls -l**
    - total 32
    - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 prog1
    - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 prog2
    - drwxr-sr-x 2 ktraxler sys 512 Oct 10 22:16 prog3
    - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 public_html
    - ktraxler@l1f1n01$

Change Directory

- cd - change directory
  - used to move throughout the *nix file system
  - cd
    - changes from the directory you're in to your home directory. Good to know when you get lost.
    - ktraxler@l1f1n01$ **ls -l**
    - total 40
    - -rw-r--r-- 1 ktraxler sys 33 Oct 10 22:18 file1a
    - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 prog1
    - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 prog2
    - drwxr-sr-x 2 ktraxler sys 512 Oct 10 22:16 prog3
    - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 public_html
    - ktraxler@l1f1n01$ **cd prog2**
    - ktraxler@l1f1n01$ **pwd**
    - /home/ktraxler/prog2

Change Directory (2)

- cd path/dirname
  - changes from the directory you're in to the directory at the end of the path. IF that directory exists in the given path.
  - ktraxler@l1f1n01$ **mkdir prog3/prog4**
  - ktraxler@l1f1n01$ **mkdir prog3/prog4/prog5**
  - ktraxler@l1f1n01$ **ls -l**
  - total 40
  - -rw-r--r-- 1 ktraxler sys 33 Oct 10 22:18 file1a
  - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 prog1
  - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 prog2
  - drwxr-sr-x 3 ktraxler sys 512 Oct 10 22:36 prog3
  - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 public_html
  - ktraxler@l1f1n01$ **cd prog3/prog4/prog5**
  - ktraxler@l1f1n01$ **pwd**
  - /home/ktraxler/prog3/prog4/prog5
  - ktraxler@l1f1n01$

Copy Command

- cp file1 path/file2
  - creates a duplicate file1 named file2
  - leaves source file intact
  - ktraxler@l1f1n01$ **cp file1a prog3/help1**
  - ktraxler@l1f1n01$ **ls -l**
  - total 40
  - -rw-r--r-- 1 ktraxler sys 33 Oct 10 22:18 file1a
  - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 prog1
  - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 prog2
  - drwxr-sr-x 3 ktraxler sys 512 Oct 10 22:41 prog3
  - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 public_html
  - ktraxler@l1f1n01$ **ls -l prog3**
  - total 16
  - -rw-r--r-- 1 ktraxler sys 33 Oct 10 22:41 help1
  - drwxr-sr-x 3 ktraxler sys 512 Oct 10 22:36 prog4
  - ktraxler@l1f1n01$

Copy Command

- ktraxler@l1f1n01$ **ls -l**
- total 24
- drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 prog1
- drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 prog2
- drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 public_html
- ktraxler@l1f1n01$ **cp -r prog1 prog1.bak**
- ktraxler@l1f1n01$ **ls -l**
- total 32
- drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 prog1
- drwxr-xr-x 2 ktraxler sys 512 Oct 10 22:07 prog1.bak
- drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 prog2
- drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 public_html
- ktraxler@l1f1n01$

Remove files

- rm - used to delete files and directories
  - rm is not recoverable unless there are backups elsewhere
  - ktraxler@l1f1n01$ **ls -l**
  - total 40
  - -rw-r--r-- 1 ktraxler sys 33 Oct 10 22:18 file1a
  - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 prog1
  - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 prog2
  - drwxr-sr-x 3 ktraxler sys 512 Oct 10 22:41 prog3
  - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 public_html
  - ktraxler@l1f1n01$ **rm file1a**
  - ktraxler@l1f1n01$ **ls -l**
  - total 32
  - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 prog1
  - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 prog2
  - drwxr-sr-x 3 ktraxler sys 512 Oct 10 22:41 prog3
  - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 public_html

Remove Directories

- rm -rf path/dirname or rm path/filename
  - recursively deletes all directories and files
  - ktraxler@l1f1n01$ **ls -l**
  - total 32
  - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 prog1
  - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 prog2
  - drwxr-sr-x 3 ktraxler sys 512 Oct 10 22:41 prog3
  - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 public_html
  - ktraxler@l1f1n01$ **ls -l prog3**
  - total 16
  - -rw-r--r-- 1 ktraxler sys 33 Oct 10 22:41 help1
  - drwxr-sr-x 3 ktraxler sys 512 Oct 10 22:36 prog4
  - ktraxler@l1f1n01$ **rm -rf prog3**
  - ktraxler@l1f1n01$ **ls -l prog3**
  - ls: 0653-341 The file prog3 does not exist.
  - ktraxler@l1f1n01$

Move - rename a file

- mv
  - mv file1 file2
    - changes the name of the file
    - ktraxler@l1f1n01$ **ls -l**
    - total 40
    - -rw-r--r-- 1 ktraxler sys 33 Oct 10 22:18 file1
    - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 prog1
    - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 prog2
    - drwxr-sr-x 2 ktraxler sys 512 Oct 10 22:16 prog3
    - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 public_html
    - ktraxler@l1f1n01$ **mv file1 file1a**
    - ktraxler@l1f1n01$ **ls -l**
    - total 40
    - -rw-r--r-- 1 ktraxler sys 33 Oct 10 22:18 file1a
    - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 prog1
    - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 prog2
    - drwxr-sr-x 2 ktraxler sys 512 Oct 10 22:16 prog3
    - drwxr-sr-x 2 ktraxler sys 512 Mar 25 2007 public_html

Viewing Files

- the “cat” command will show the contents of an entire file
  - cat filename
  - if your file is extremely long you will get only the last lines that your terminal will hold
- the “more” command will show the contents of an entire file but one terminal screen at a time
  - more filename
- the “head” command will show 10 lines from the beginning of a file
    - head filename

Wild Cards

- Many linux commands will allow you to use wildcards. They are:
  - '?' substitute any one character here
  - '*' any string of characters
  - ls t[aeo]st.txt
    - lists every file taht starts with a 't', ends with a 'st.txt' and has either an 'a', 'e', or 'o' in the second position

Getting help from the system

- man COMMANDNAME
  - if manual pages are install the page for COMMANDNAME is displayed through more
- which COMMANDNAME
  - 10-4-1-195:~ kathy$ which java
  - /usr/bin/java
- whereis COMMANDNAME
  - 10-4-1-195:~ kathy$ whereis cp
  - /bin/cp
- whatis COMMANDNAME
  - 10-4-1-195:~ kathy$ whatis tar
  - tar(1) - tape archiver; manipulate "tar" archive files

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`

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
- The **`file`** command reveal what type of file the target is

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: your home directory contains directories `Desktop`, `Documents` and `Downloads`

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 of the file)
  - Group (g) (group owner of the file)
  - World (o) (everyone else who is 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

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

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`
  - You are only in one of the above groups due to software licensing
  - Groups in LSU HPC system
    `Users, Admins…`

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@smic.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:/work/user/somefile .`
- `scp -r code user@eric.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

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” 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

Editor cheatsheet (1)

Cursor Movement	vi	emacs
move left	h	C-b
move down	j	C-n
move up	k	C-p
move right	l	C-f
jump to beginning of line	0	C-a
jump to end of line	$	C-e
goto line n	nG	M-x goto-line ⟵ n
goto top of file	1G	M-<
goto end of file	G	M->
move one page up	C-u	M-v
move one page down	C-d	C-v

Editor cheatsheet (2)

File Manipulation	vi	emacs
save file	`:w`	`C-x C-s`
save file and exit	`:wq, ZZ`	
quit	`:q`	`C-x C-c`
quit without saving	`:q!`	
delete a line	`dd`	`C-a C-k`
delete *n* lines	*n*`dd`	`C-a M-`*n*` C-k`
paste deleted line after cursor	`p`	`C-y`
paste before cursor	`P`	
undo edit	`u`	`C-_`
delete from cursor to end of line	`D`	`C-k`
search forward for *patt*	`\`*patt*	`C-s` *patt*
search backward for *patt*	`?`*patt*	`C-r` *patt*
search again forward (backward)	`n`	`C-s(r)`

Editor cheatsheet (3)

File Manipulation (contd)	vi	emacs
replace a character	`r`	
join next line to current	`J`	
change a line	`cc`	
change a word	`cw`	
change to end of line	`c$`	
delete a character	`x`	`C-d`
delete a word	`dw`	`M-d`
edit/open file *file*	`:e file`	`C-x C-f file`
insert file *file*	`:r file`	`C-x i file`
split window horizontally	`:split or C-ws`	`C-x 2`
split window vertically	`:vsplit or C-wv`	`C-x 3`
switch windows	`C-ww`	`C-x o`

Exercise (1)

- Login to a Linux machine and open a terminal
- Enter the following commands or carry out operations asked for.
- Understand what you are doing and ask for help if unsure. Some commands are incorrect or will fail; if this is the case, enter the correct ones

Exercise (1)

- $ echo hello world
- $ pwd
- $ whoami
- $ cd /tmp
- $ cd –
- $ mkdir test/testagain
- $ cd test/testagain
- $ touch file

❖ Go back to your home directory

❖ Figure out which shell you are using

Exercise (4)

If you have never used vim or emacs, go through the vim tutorial: vimtutor

```
===============================================================================
=        W e l c o m e       t o       t h e     V I M     T u t o r -
         Version 1.7         =
===============================================================================

Vim is a very powerful editor that has many commands, too many to explain in a
tutor such as this. This tutor is designed to describe enough of the commands
that you will be able to easily use Vim as
an all-purpose editor.
```

More Hands on Exercises

- For more hands on exercises visit:

- http://krt3.lsu.edu/training/linux/linux-for-hpc.html
- http:/cli.learncodethehardway.orbash_cheat_sheet.pdf
- http://vim.rtorr.com/

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
- Online courses: http://moodle.hpc.lsu.edu
- Contact us
  - Email ticket system: sys-help@loni.org
  - Telephone Help Desk: 225-578-0900
  - Instant Messenger (AIM, Yahoo Messenger, Google Talk)
    - Add “lsuhpchelp”