Linux Workshop

Laboratory for Molecular Simulation

& High Performance Research Computing

Lisa M. Pérez, Manager of the LMS and HPRC Enablement Staff

Michael B. Hall, Director of the LMS

http://lms.chem.tamu.edu/

mouse@tamu.edu

Phone: 979.845.9384

Office: Room 2109 Chemistry (CHAN)

What is Linux?

- The 1st Unix OS was Developed at Bell laboratories in Murray Hill, New Jersey, in 1969.
  - Macintosh OS (1979)
  - DOS - Disk Operating System (1980, Tim Paterson)
  - Linux (1991, Linus Torvalds)
- Linux is a popular operating system
  - Stable, Fast, Secure and Powerful
  - Designed for multi-user and multi-tasking
  - Easy to share data and programs securely
- Command line is not user friendly
  - "Unix is user friendly, it is just particular about who its friends are."
- Available for almost all hardware.
- Common Linux Operating Systems
  - Ubuntu, Fedora Core, Centos, Red Hat, SUSE, etc

Shared Resources

- CPU (Central Processing Unit) - Allocation to a process based on a priority scheme
- Memory

  - RAM (Random Access Memory): Used for fast access to data of a program
  - SWAP: Slower because the program needs to read/write the data needed from the hard drive. Swapping refers to moving entire processes in and out of main memory to disk.

  - **free** is a linux command to show memory availability
- Disk (Hard Drive(s))

  - On small systems, the user normally has access to the entire disk space available in the home and scratch partitions.
  - On larger systems, the user is limited to the disk space allocated to users via a quota system.

Setting up an account

- Username/User ID - unique name on a machine - often your TAMU netid
- Password – 8 or more characters that must contain a number or special character or both
- Shell - a program that lets the user communicate with the Linux kernel.
  - Great information about shells: www.linfo.org/shell.html
  - **Bash shell (bash)** - most commonly used shell on Linux systems
  - Bourne shell (sh) – often used for system administration.
  - C shell (csh)
    - T-shell (tcsh) - historicaly, most commonly used shell on UNIX systems
  - Kourne shell (ksh) – most commonly used on IBM/AIX systems
  - See http://www.freebsd.org/ports/shells.html for a long list of shells (zsh, ash, dash, fish, mudsh, etc)

Directives used in this Lecture

- **Bold** words should be entered explicitly
- *Italicized* words are variable depending on the information that the utility needs
- ␣ symbol is used to represent a space
- ↵ symbol is used to represent the enter/return key

Bash Shell Control

- Prompting
  - Bash prompt can be defined by the PS1 variable:
  - **PS1="[\u@\h \W] : "**
  - [*username@hostname folder*] :
  - [mouse@terra Linux] :
  - An active prompt means that the shell is ready for you to type a command.
- Command Interpretation and Execution
  - When a command is typed at the prompt, the Shell processes the command and sends it to the Linux kernel.
    - example: [mouse@terra ~] : **ls** ↵
    - [mouse@terra ~] : is the prompt and **ls** is the command
    - **ls** is a command to list all the files in the current directory
    - more about commands later.…
- Each shell has its own scripting language

A scripting language is a programming language that supports scripts: programs that automate the execution of tasks that could alternatively be executed command line. Scripting languages are often interpreted (rather than compiled).

Customizing the Environment

- Two important files for customizing your Bash Shell environment
  - **.bashrc** (prounced dot bashrc)
    - contains aliases, shell variables, paths, etc.
    - executed (sourced) upon starting a non-login shell.
  - **.bash_profile** (dot bash_profile)
    - also can contain aliases and shell variables
    - normally used for terminal settings
    - executed (sourced) upon login
    - if **.bash_profile** doesn't exist, the system looks for **.profile** (dot profile)
- **.␣.bashrc** (or **source␣.bashrc**)
  - Executes the commands in the .bashrc file
  - The ␣ character will be used to represent a space

.bashrc file contents

```
# Settings for an interactive shell
    if [ ! -z "$PS1" ]; then   #test to see if the variable PS1 is set
      PATH="$PATH:$HOME/bin:."
      # change what the prompt looks like
      PS1="\u@\h [\!]: "
      # mouse@terra [123]:
      fi

# personal aliases
    alias h= "history | more"
    alias m="more"
    alias ll= "ls  -la"
    alias ls= "ls -CF"
    alias rm= "rm  -i"
    alias cp= "cp  -i"
    alias mv= "mv  -i"
    alias x="chmod u+x"
    alias vmd= "/apps/vmd/vmd-1.9.3/bin/vmd"

# A line that begins with a # is a comment
```

The **PATH** variable is a list of folders. When you type in a command, the operating system has to find that command. The OS searches for the command in each folder that you list in your PATH variable in the order that they appear. This is true on Windows and Mac. To see your PATH, type: **echo␣$PATH**

.bashrc file contents (variables and functions)

```
# Syntax to set a local variable
#    varname=value
# Syntax to set a global variable
#    export␣varname=value
# Syntax to set an alias
#    alias␣name="value"
# Syntax to create a function
#    function␣name()␣{␣command␣;␣}

# Settings for the variables for the quantum code Gaussian 16
export  g16root=/apps/g16_B01
.  $g16root/g16/bsd/g16.profile
# the previous line executes
# .  /usr/local/g16_B01/g16/bsd/g16.profile
# because g16root is set to /apps/g16_B01

function  cc() { awk  -f  cc.awk  "$@".log>"$@".cc  ; }
# If you type cc test at the prompt, the following command will be executed:
# awk  -f  cc.awk  test.log > test.cc
```

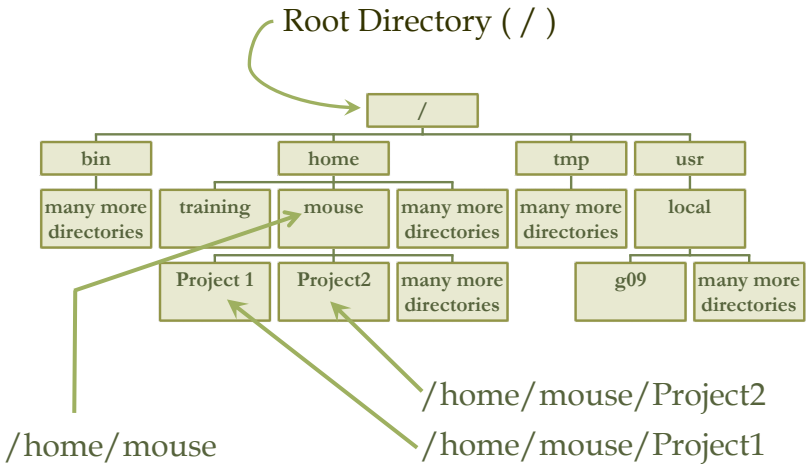

Simple Utilities

- **Bold** words should be entered explicitly
- *Italicized* words are variable depending on the information that the utility needs
- ␣ symbol is used to represent a space
- ↵ symbol is used to represent the enter/return key
- **man**␣*command*↵ or **info**␣*command*↵
  - displays manual entry for command
  - **man**␣**-k**␣*keyword*↵ or **apropos**␣*keyword*↵
    - lists all manual entries that contain your keyword
- **passwd**↵ - sets or changes your password
  - if passwd↵ doesn't work, check the documentation for the machine and then ask the administrator for assistance.

Simple Utilities

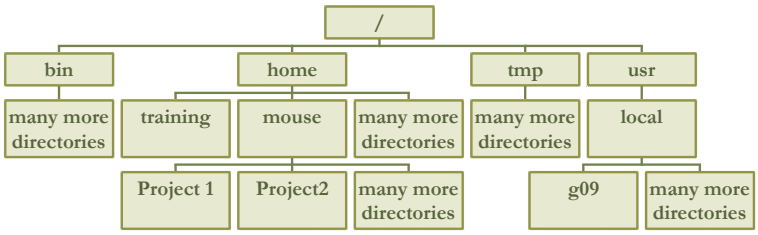
- **logout**↵ or **exit**↵ - closes a terminal or ssh session
- **date**↵ - displays the current date and time (not necessarily the correct date or time)
- **clear**↵ - clears your screen
- **hostname**↵ - prints the hostname to the screen
- **whereis** ␣*command*↵ - find a program
- **locate** ␣*command*↵ - find a file (program, dir, file, etc)
- **ctrl-c** (^c) - interrupts a process (avoid this as it can leave garbage/temporary files on the system)
- **^z** (ctrl-z) - Stops a process, but does NOT terminate it
  - **bg**↵ - puts the suspended process into the background
  - **fg**↵ - puts the suspended process into the foreground

File System Hierarchy



File System Hierarchy

- **pwd**↵ - **p**rints your current **w**orking **d**irectory
- **cd**↵ - changes to your home directory (**c**hange **d**irectory)
- **cd** *name*↵ - change directory to name
  - absolute pathnames (start with a forward slash /)
    - cd /home/mouse/Project1↵
  - relative pathnames (do NOT start with a /)
    - . current directory
    - .. parent directory
    - ~ home directory
      - cd ../../tmp↵
      - cd ~↵
      - cd ~/Project1↵
      - cd ~training↵



Managing Files & Directories

- Editors
  - Graphical text editors
    - gedit, nedit, xemacs, kedit, ….
  - Command driven (non-graphical) text editors
    - vi, emacs, **…**
    - powerful and fast editors that may be used at any interface, but they are not user friendly.
- File and Directory Names
  - Do **NOT** use spaces (use _ or - instead but don't start a filename with -), meta, special or reserved characters
  - No, no, no, and no: * ? $ / \ { } [ ] ; : ' ` " & Tabs ! @ ( ) < >
  - A file cannot have the same name as the directory where it resides.
- searching for a file or directory
  - **whereis** *filename*↵
  - **locate** *filename*↵
  - **find . -name '***search string***' -print**↵
    - find . -name '*test1*' -print↵
    - searches for any file or directory with the string test1 in it from the current directory and down the hierarchy (-iname makes the search case insensitive)

Managing Files & Directories

- Printing directory contents to the screen
  - **ls**␣↵ - lists contents of working directory
  - **ls**␣*dirname*↵ - lists the contents of the directory specified by dirname
  - ls␣-aCFl↵ (flags)
    - -a print hidden files
    - -l print long listing
    - -F print a special character after special files
    - use man␣ls to find all possible flags
  - **tree**↵ - recursive directory listing
- Printing a files contents to the screen
  - **cat**␣*filename*↵, **less**␣*filename*↵, **more**␣*filename*↵, **page**␣*filename*↵
  - **head**␣**-***n*␣*filename*↵ (where n is an integer)
    - displays the first *n* lines
  - **tail**␣**-***n*␣*filename*↵
    - displays the last *n* lines
  - **tail**␣**-f**␣*filename*↵
    - Display the last 10 lines of a file and waits for new lines – ctrl-c (^c) to exit.

Managing Files & Directories/Folders

- Making a directory (dir)
  - **mkdir**␣*dirname*↵ (creates a directory in the current dir)
  - mkdir␣tmp↵ (creates the directory tmp in the current dir)
  - mkdir␣~/tmp↵ (creates the directory tmp in your home dir)
  - mkdir␣/home/mouse/tmp↵ (created the directory tmp in /home/mouse)
- Renaming a file
  - **mv**␣*oldfilename*␣*newfilename*↵ (note: new **cannot** be a directory name) You need to specify the location of *oldfilename* and *newfilename*. This command specifies the *oldfilename* and *newfilename* are in the current directory because there is nothing in front of the names.
- Move a file into a new directory
  - **mv**␣*filename*␣*dirname*↵ (note: *dirname* must be a directory that already exists.)
  - retains the filename but moves it to the directory *dirname*
  - You can rename the file while moving it to a new directory: **mv**␣*oldfilename*␣*dirname/newfilename*↵
- Rename a directory
  - **mv**␣*olddirname*␣*newdirname*↵
- Safe mv
  - **mv**␣**-i**␣*oldfilename*␣*newfilename*↵
  - -i is a flag that modifies the way mv behaves. In this case –i tells the command to prompt you for permission if you are about to overwrite a file.

Wildcards (globbing)

- ***** matches any number of characters
- **?** matches any single character
- **[ ]** matches a single character for a specified range of characters given in the brackets
- {} matches a list of patterns separated by a comma within the curly brackets
- examples
  - mv␣proj1*␣~/Project1↵
    - moves all files beginning with proj1 into dir Project1
      - Note: the dir Project1 must already exists in your home dir
    - ls␣proj?.log↵
    - lists all files where ? can be any one character
  - mv␣enzyme[12].com␣enzyme↵
    - moves enzyme1.com and enzyme2.com into dir enzyme
  - mv project{*.com,*.log,*.txt}␣project1-5
    - moves all files that start with project and end with .com, .log, or .txt to the directory project1-5 that already exists.

Managing Files & Directories

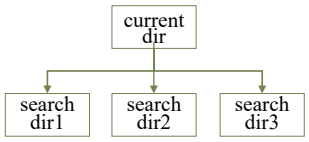
- Making a copy of a file
  - **cp**␣*oldfilename*␣*newfilename*↵
    - Makes a copy of the file named *oldfilename* and names it *newfilename* in the current directory
    - Note: *newfilename* cannot be the name of a directory
- Copying a file to a new directory
  - **cp**␣*filename*␣*dirname*↵
    - Makes a copy of the file named *filename* to the directory named *dirname*
    - Note: *dirname* must already exist
- Copying a directory
  - **cp**␣**-R**␣*olddirname*␣*newdirname*↵
    - Makes a complete copy of the directory named *olddirname* including all of its conents, and names it *newdirname* in the current directory
    - Note: *newdirname* cannot be the name of a directory that already exists
- Safe copy
  - **cp**␣**-i**␣*oldfilename*␣*newfilename*↵
    - will prompt you if you are about to overwrite a file named *newfilename*

Managing Files & Directories

- Deleting a file
  - **rm** *filename*↵
    - Deletes the file named *filename* **forever!**
- Deleting a directory
  - **rmdir** *dirname*↵
    - Deletes an empty directory named *dirname*
  - **rm -r** *dirname*↵
    - removes the directory named *dirname* and all of its contents **forever!**
- Safe delete
  - **rm -i** *filename*↵
    - will prompt you for confirmation before deleting *filename*
- Warning! Once a file is deleted or overwritten it is gone. Be VERY careful when using wildcards. rm -r *↵ will remove everything from that directory and down the directory hierarchy!

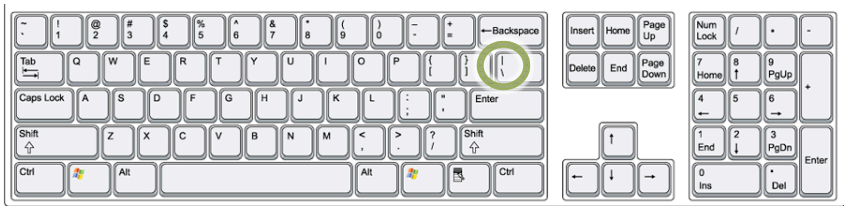
Searching File Contents

- **grep** *search-pattern filename*↵ - searches the file *filename* for the pattern *search-pattern* and shows the results on the screen (prints the results to standard out).
  - grep Energy run1.out↵
    - searches the file run1.out for the word Energy
    - grep is case sensitive unless you use the -i flag
  - grep Energy *.out↵
    - searches all files in that end in .out
  - grep "Total Energy" */*.out↵
    - You must use quotes when you have blank spaces. This example searches for Total Energy in every file that ends in .out in each directory of the current directory
  - grep –R "Total Energy" Project1
  - Searches recursively all files under Project1 for the pattern Total Energy



Searching File Contents

- **egrep␣'***pattern1*|*pattern2*|*etc*'␣*filename*↵
  - searches the file filename for all patterns (*pattern1*, *pattern2*, etc) and prints the results to the screen.
  - The | character is called a pipe and is normally located above the return key on the keyboard.
  - egrep␣'Energy|Enthalpy'␣*.out↵
    - searches for the word Energy or Enthalpy in every file that ends in .out in the current directory.



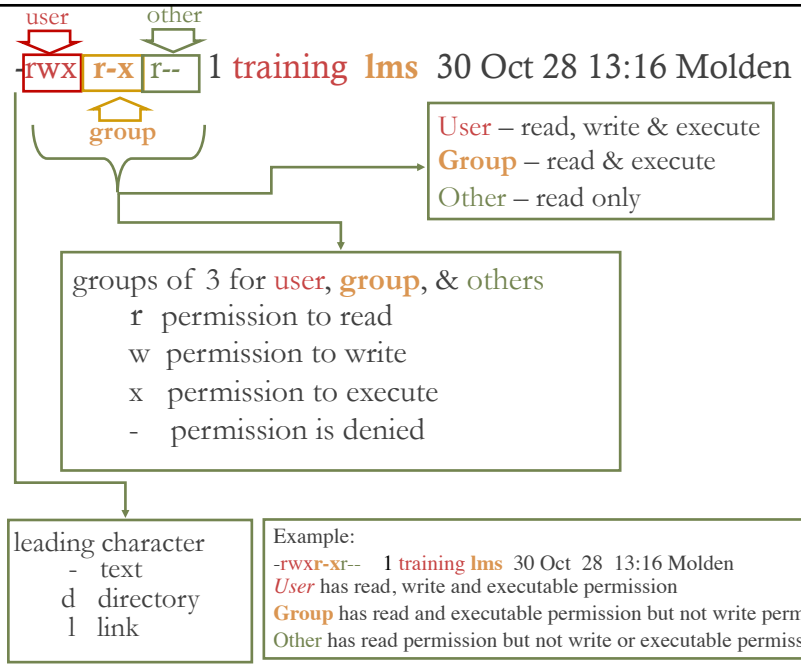
File Attributes

ls␣-l↵ lists the files in the dir in long format

Note: the flag is the letter l and not the number 1

-rwxr-xr-- 1 training lms 30 Oct 28 13:16 Molden

1	hard link count
training	file owner
lms	group ID
30	file size
Oct 28 13:16	time the file was last modified
Molden	filename



Permissions

- To change the read, write and executable permission for users (u), group (g), others (o) and all (a)
- **chmod␣u+x␣***filename*↵ (or *dirname*↵)
  - adds executable permission for the user
- **chmod␣og-r␣***filename*↵ (or *dirname*↵)
  - remove read permission for group and others
- **chmod␣-R␣a+rx␣***dirname*↵
  - give everyone read and executable permission from *dirname* and down the hierarchy
- chmod␣u=rwx␣*filename*↵
  - sets the permission to rwx for the user
- chmod␣g=␣*filename*↵
  - sets the permision to --- for the group
- You can also use numbers
- r = 4, w = 2, and x = 1, - = 0
  - chmod␣755␣*filename*↵ (result -rwxr-xr-x)
  - chmod␣600␣*filename*↵ (result -rw-------)

---	0
--x	1
-w-	2
-wx	3
r--	4
r-x	5
rw-	6
rwx	7

Ownership/Groups

- To change the group
  - **chgrp** *groupname filename*↵ (or *dirname*↵)
    - Changes the group for *filename* or for *dirname* but not for the files contained within *dirname*
  - **chgrp -R** *groupname dirname*↵
    - Changes the group for all of the files and directories down the hierarchy from *dirname*
    - Example: **chgrp -R lms training**↵
- Change owner
  - **chown** *username filename*↵ (or *dirname*↵)
    - Changes the owner of *filename* or *dirname* but not for the files contained within *dirname*
  - **chown -R** *username dirname*↵
- The chown and chgrp command is not allowed for users by default on many Linux OS's
- Change owner and group
  - **chown** *username*:*groupname filename*↵
  - **chown** *username*.*groupname filename*↵

Managing Disk Usage

- Most large computer systems impose a quota system for users
  - displays your disk allotment and usage: **quota -v**↵ and/or **mmlsquota**↵
- **df -h**↵
  - displays the available file systems in the easiest readable unit.

```
mouse@lms12[1000]: df -h
   Filesystem         Size  Used Avail Use% Mounted on
   /dev/nvme0n1p3      50G   12G   39G  23% /
   devtmpfs           7.8G     0  7.8G   0% /dev
   /dev/nvme0n1p7     152G  346M  152G   1% /scratch
   /dev/nvme0n1p5      10G   33M   10G   1% /tmp
   /dev/nvme0n1p6      10G  2.5G  7.6G  25% /var
   /dev/sda1          233G   51G  183G  22% /work
   /dev/nvme0n1p2     497M  277M  221M  56% /boot
   /dev/nvme0n1p1     200M   18M  183M   9% /boot/efi
   license:/home       20T   13T  7.6T  62% /home
   license:/apps/lms  392G  347G   26G  94% /apps
   tmpfs              1.6G   16K  1.6G   1% /run/user/42
```

- disk starting with /dev should indicate that it is "local" disk (reasonably fast access disk)
- disk starting with a network address indicates that it is network mounted disk (slow/slower access)
- **du -sh** prints your disk usage from the current directory and down (may take some time to complete)

Compressing Files

- Compressing files
  - **gzip** *filename*↵
    - zips-up filename and creates filename.gz
  - **gzip -v** *filename*↵
    - zips-up filename in a verbose manner (tells you % compression)
  - **gzip -r** *dirname*↵
    - zips-up all files down the hierarchy from dirname
  - **gunzip** *filename***.gz**↵
    - unzips filename.gz and creates filename
  - **bzip2** *filename*↵
    - zips-up (compresses) filename and creates filename.bz2 (or .bz or .bzip2)
  - **bunzip2** *filename***.bz2**↵
    - unzips filename

Archiving Files/Directories

- **tar -cpvf** *filename***.tar** *filenames*↵ (or *dirnames*↵)
  - Archives filenames and/or dirnames into the file filename.tar
  - It is best to zip-up your files before archiving them.
- **tar -xpvf** *filename***.tar**↵
  - Extracts the contents of filename.tar
- some of the tar flags
  - -c creates a new archive
  - -x extract files and/or directories form the archive
  - -p preserve protection information
  - -v verbose
  - -f working with files
  - -t lists the table of contents for an archive

ZIP command

- **zip** *filename*.**zip** *filenames*↵
  - Zips and archives filenames into the file *filename*.**zip**
- **zip -r** *filename*.**zip** *dirname*↵
  - Zips and archives files in *dirname* and down the heirarchy into the file *filename*.**zip**
- **unzip** *filename*.**zip**↵
  - Extracts the contents of *filename*.**zip**
- some of the tar flags
  - -v verbose
  - -l lists the table of contents for a zip file
  - -m delete the original files

Redirecting Input and Output

- Redirecting output
  - **>** symbol redirects output
    - *command*>*outputfilename*↵
    - ls -al>list-of-files.txt↵
    - >> symbol appends to the end of the file instead of overwriting it.
    - ls -al>>list-of-files.txt↵
- Redirecting input
  - **<** symbol redirects input
    - *program*<*inputfile*↵
    - g16<run1.com↵
    - output would go to standard out (ie monitor)
- Redirecting input and output together and running in the background
  - *program*<*inputfilename*>*outputfilename*&↵
  - g16<run1.com>run1.log&↵

File Type

- **file** *filename*
  - Returns the file type

DOS format

```
mouse@lms1 [1004]: file sample.txt
sample.txt: ASCII text, with CRLF line terminators
```

UNIX format

```
mouse@lms1 [1006]: file sample.txt
sample.txt: ASCII text
```

Mac format

```
mouse@lms1 [1008]: file sample.txt
sample.txt: ASCII text, with CR line terminators
```


File Type Conversion

- **dos2unix** *filename*
  - Convert dos line terminator to unix

```
mouse@lms1 [1016]: file sample.txt
sample.txt: ASCII text, with CRLF line terminators

mouse@lms1 [1017]: dos2unix sample.txt
dos2unix: converting file sample.txt to Unix format ...

mouse@lms1 [1018]: file sample.txt
sample.txt: ASCII text
```

  - mac2unix
  - unix2dos
  - unix2mac

Pipes

- Pipes |
  - takes the output of one command and sends it to another
  - ls|more↵ or ls|less↵
    - List the files one page at a time
  - grep␣Energy␣run1.out|grep␣HF↵
  - grep␣Energy␣run1.out|grep␣HF > HF_output.txt↵
    - Searches a file named run1.out for the word Energy and then searches for the word HF in the lines that have the word Energy. The resulting information is then sent to a file named HF_output.txt

history, !, ↑, ↓, & tab completion

- **history**↵ (usually aliased to h↵)
  - The history command will list your last n commands (n = integer).
- **!!**↵ - repeats your last command
- **!***n*↵ - repeats the nth command
  - You can find the number of the command using history
- **!***name*↵ - repeats the last command that started with name
- You can use the up (↑) and down (↓) arrow keys to scroll through previous commands
- Tab - will try to complete the rest of the file/directory name you are typing
  - If you have three files that start with x (xrun15 xrun16 and xrun17) then typing x and then tab will result in xrun1 at the prompt and you would have to type in the last character. On some systems, if you hit the tab key twice it will result in xrun1 at the prompt and list the 3 files that match.

Managing Processes

- **top**↵ shows all processes in a table
  - **q**↵ will exit the top process

```
top - 22:26:17 up 89 days,  5:25,  1 user,  load average: 8.00, 8.01, 6.92
Tasks: 279 total,   9 running, 270 sleeping,   0 stopped,   0 zombie
Cpu(s): 98.9%us,  1.1%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  24728124k total,  2533328k used, 22194796k free,   166776k buffers
Swap:  4095996k total,        0k used,  4095996k free,  1549420k cached
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
26469 mouse 20   0 15.9g 363m 6196 R 100.4  1.5   3:08.78 l502.exe
25404 mouse  20   0 15.9g 363m 6196 R 100.1  1.5  30:27.45 l502.exe
26468 mouse  20   0 15.9g 363m 6196 R 100.1  1.5   3:08.77 l502.exe
26470 mouse  20   0 15.9g 363m 6196 R 100.1  1.5   3:08.72 l502.exe
26471 mouse  20   0 15.9g 363m 6196 R 100.1  1.5   3:08.77 l502.exe
26472 mouse  20   0 15.9g 363m 6196 R 100.1  1.5   3:08.74 l502.exe
26473 mouse  20   0 15.9g 363m 6196 R 100.1  1.5   3:08.64 l502.exe
26474 mouse  20   0 15.9g 363m 6196 R  99.7  1.5   3:08.40 l502.exe
26621 mouse  20   0 15160 1380  952 R   0.0  0.0   0:00.02 top
```


Managing Processes

- **ps␣-u␣***username*↵ (list all of the processes for username)
- [mouse@vici ~]$ ps -u mouse

```
  PID TTY          TIME CMD
26780 ?        00:00:00 sshd
26781 pts/6    00:00:00 bash
27756 pts/6    00:00:00 nedit
28362 pts/6    00:00:00 ps
32432 ?        00:00:00 sshd
32433 pts/3    00:00:00 bash
32626 pts/3    00:00:00 vim
```

- **kill␣***pid*↵ kills the process with pid (process id number (PID)) nicely
- **kill␣-9␣***pid*↵ kills the process with pid without remorse – not nice or clean…
- To kill the nedit process: kill␣27756
- Check to see if it is gone (ps –u mouse) and if it is not, use: kill␣-9␣27756

Computer Networking

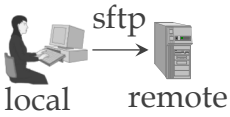
- Secure Shell (ssh) - Used to access a remote machine through a secure protocol
  - **ssh␣***username@remotehostname*↵ (username is different on the remote machine)
  - **ssh␣***remotehostname*↵ (username is the same on the local and remote machines)
    - ssh␣ada.tamu.edu↵
    - ssh␣ada↵
    - ssh␣mouse@ada.tamu.edu↵
    - ssh␣mouse@ada↵
    - The first time that you ssh to a machine from the local host, it will ask you for permission. You must type yes↵ to continue (y↵ will not work).
    - You will be prompted for your password

Graphics Across the Network

- If you are using ssh and need to run a graphical application remotely, you may need to try the commands and/or settings in the following order until one of them works.
  - **ssh␣-X␣***username@remotehostname*↵
  - On older machines, you may need to use:
    - **ssh␣-Y␣***username@remotehostname*↵

Secure File Tranfer Protocol (sftp)

- **sftp** is used to transfer files between unix/linux machines
- **sftp** *remotehostname*↵ or **sftp** *username@remotehostname*↵
  - sftp will ask you for your password and the first time you sftp to a machine it will ask you for permission. You must type **yes**↵ to continue (y↵ will not work).
- commands used in the sftp session
  - **get** *filename*↵ - copies *filename* from the remote machine to the local machine.
    - Wildcard usage: get *.out↵ get all of the files that end in .out automatically.
  - **put** *filename*↵ - copies *filename* from the local machine to the remote machine.
    - Wildcard usage: mput *.out↵ put all of the files that end in .out automatically.
  - **ls**↵ - list the contents of the remote machine directory
  - **lls**↵ - list the contents of the local machine directory
  - **cd** *dirname*↵ - changes the remote machine directory
  - **lcd** *localdir*↵ - change the local machine directory
  - **mkdir** *dirname*↵ - makes a dir *dirname* on the remote machine
  - **lmkdir** *dirname*↵ - makes a dir *dirname* on the local machine
  - **pwd**↵ - prints the working directory of the remote machine
  - **lpwd**↵ - prints the working directory of the local machine
  - **bye**↵ or **quit**↵ - exits an sftp session.
  - **!***command*↵ - executes a local shell command (i.e. hostname)



Free graphical sftp programs:

Windows: Mobaxterm, Filezilla, Winscp
Mac: Fetch (free for academics), Filezilla
Linux: Filezilla

Secure copy (scp)

- **scp** *filename username@remotehostname***:***remotepath*↵
  - scp run1.out mouse@ada.tamu.edu:↵
    - Makes a copy of run1.out located on the local machine to your home directory on ada
  - scp run1.out ada.tamu.edu:/scratch/mouse/↵
    - Makes a copy of run1.out to /scratch/mouse instead of the home directory. This syntax assumes that your username is the same on both machines
- **scp** *username@remotehostname***:***filename localpath*↵
  - Copies a file from the home directory on the remote host to the current directory on the local machine
- -r recursively copy an entire directory (not suggested)
  - **scp -r** *dirname remotehostname***:**↵
    - copies the entire directory heirarchy of *dirname* to the home directory on the remote machine. Links (ie shortcuts) will cause problems.
- Useful flags:
  - -v debugging/verbose printing
  - -p preserve modification time, access times and modes

rsync

- rsync finds files that need to be transferred using a "quick check" for files that have changed in certain attributes (size, last-modified time, etc).
- **rsync␣-avu␣-e␣ssh␣***dirorfilename(s)*␣*username@remotehostname***:***remotepath*↵
  - rsync␣Project1␣mouse@ada.tamu.edu:↵
    - Makes a copy of the directory Project1 located on the local machine to your home directory on ada with the name Project1
  - -a archive (preserve permissions, group, owner, and time stamp, copy symbolic links, and copies recursively)
  - -v verbose
  - -u doesn't transfer a file if it is newer on the receiving end
  - -e␣ssh use ssh protocol

vi editor

- **vi␣***filename*↵ - opens (creates) a file using vi
- **vi␣-R␣***filename*↵ - opens a file using vi in read-only mode
- **view␣***filename*↵ - same as **vi␣-R␣***filename*↵
- Two modes
  - insert mode
    - for typing in text
    - all keystrokes are interpreted as text
    - **i** one of the commands that initiates insert mode
  - command mode
    - for navigating the file and editing
    - all keystrokes are interpreted as commands
    - **Esc** returns the user to command mode

%vi filename↵

|

~

~

~

"filename"

starts in command mode

Typing :set␣showmode↵ while in command mode will display in the lower right hand corner what mode you are in (it doesn't always work properly…)

vi commands

- To exit a file or save
  - **ZZ** or **:wq**↵ or **:x**↵ - save the file and exit
  - **:w**␣*filename*↵ - save the file with the name filename
  - **:w!**↵ - force save
  - **:q**↵ or **:q!**↵ -quit without saving
    - **:q**↵ quits a file when there have been no changes
    - **:q!**↵ quits the file regardless of changes
- Moving around in the file
  - **h**, **l** (or **space**), **j** and **k** - left, right, down and up
  - **G** Move to end of file
  - **^f** (^ = Ctrl-key) Scroll down a full screen
  - **^b** Scroll up a full screen
  - **0** (zero) Move to start of current line
  - *n***G** Go to line *n*

vi commands

- **w** move forward one word
- **b** move back one word
- **e** move to the end of the word

- Text Editors
  - Commands that take you into insert mode
    - **i** insert text to the left of the cursor
    - **I** inserts text at the beginning of the line
    - **a** insert text to the right of the cursor
    - **A** insert text at the end of the line
    - **o** open a line below the cursor
    - **O** open a line above the cursor
    - **R** overwrite text to the right of the cursor

vi commands

- Editor commands that keep you in command mode
  - **x** deletes a character (the character the cursor is on)
  - **dd** deletes a line (the line the cursor is on)
  - ***n*dd** deletes *n* lines
  - **dw** deletes a word
  - **dG** deletes to the end of the file
  - **D** deletes to the end of the line
  - **r**␣*a* replaces current character with *a* (a = character, number, etc.)
  - **u** undo last command (only 1 undo on most unix machines. Most new versions of vi (vim) have multiple undo and redo (Ctrl-r) capability)
  - ***n*yy** yank *n* (*n* is a number) lines to memory
  - **p** put the yanked lines below the cursor
  - **P** put the yanked lines above the cursor
- Editor commands that put you in insert mode
  - **cw** changes a word to the text that you type it - you have to have the cursor at the beginning of the word

vi commands

- Miscellaneous commands
  - **/***name*↵ search forward for name
  - **?***name*↵ search backward for name
  - **:1,$␣s/***pattern1***/***pattern2***/g**↵
    - from line 1 to the bottom find and substitute *pattern1* for *pattern2*
    - you could also use **:%␣s/***pattern1***/***pattern2***/g**↵
      - % and 1,$ mean the entire file
    - the **g** means that all occurrences of *pattern1* will be substituted in a line and not just the first one
  - **:e␣***filename*↵ exits to the file *filename*
  - **ma** - marks that line and stores the position in the variable a
  - **:'a,.␣y␣x**↵ yanks the lines between the mark a and where the cursor is (.) and stores it in the variable x
  - **:pu␣x**↵ puts the lines stored in x into the file where the cursor is
  - **:r␣***filename*↵ insert the file filename into the current file.
  - **:set␣all**↵ lists all of the settings
  - **:set␣number**↵ displays line numbers

Obtaining an Account

- Contact the System Administrator
  - TAMU University Machines – High Performance Research Computing
    - Terra
    - Ada
    - Curie
    - https://hprc.tamu.edu/
  - LMS and/or Chemistry Departmental machines
    - https://lms.chem.tamu.edu/
  - Brazos Cluster
    - http://brazos.tamu.edu

Useful Websites and Books

- http://lms.chem.tamu.edu/
- http://hprc.tamu.edu/
- http://www.lynda.com/
- Safari Books Online
  - You must be behind the TAMU Firewall or logged in through the TAMU Library system.
  - http://proquest.safaribooksonline.com/
- Books
  - The O'Reilly series of books are quite good http://linux.oreilly.com/
    - Linux in a Nutshell
    - Essential System Administration
    - sed & awk
    - Programming in Perl
    - and many more

