Introduction to Linux

Wes Brashear

30 January 2026

High Performance
Research Computing
DIVISION OF RESEARCH

Course Outline

1. Introduction
2. Managing Directories & Files
3. More about Directories & Files
4. Useful Commands and Tools
5. Customizing Environment
6. Remote Access and File Transfer

Directives Used in this Class

Commands to type in will use the following:

- **Bold** words should be entered explicitly
- *Italicized* words are variable depending on the information that the utility needs
- commands for you to type in $
- command output in

$ = start of prompt, do not include this in your command

Introduction

What is Linux?

- Linux is a family of open-source **Unix-like** operating systems based on the Linux kernel
  - 1st Unix OS (**1969**), Macintosh OS (1979), DOS - Disk Operating System (1980)
  - Linux (**1991**, Linus Torvalds)
- A kernel is the lowest level of software that can interface with computer hardware.
- 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**
- Available for almost all hardware
- Common Linux Operating Systems
  - Centos, Red Hat, Ubuntu, Fedora Core, SUSE, etc

Linux Flavors

- RedHat/CentOS
- OpenSuse
- Rocky Linux

- Ubuntu
- Linux Mint
- Pop!_OS
- Kali Linux
- Deepin

- Arch Linux
- Manjaro OS
- Garuda Linux

Command Line Interface (CLI)

Linux systems rely on the **command line** much more than other operating systems despite the presence of numerous desktop environments (or GUIs, Graphical User Interfaces).

- Why use CLI?
  CLI works almost everywhere
  CLI is fast and powerful
- Where to type the command?
  Terminal: text input and output interface
  - A terminal is a wrapper program that runs a **shell** and allows us to enter commands

  Shell: command line interpreter
  - A shell is the program that actually processes the command and output results
  - Different types of shells are available in Linux

Bash Shell Control

- Shell - a program that lets the user communicate with the Linux kernel
  - Each shell has its own scripting language
  - **Bash shell (bash)** - most commonly used shell on Linux systems
  - Zsh - default shell for Mac
  - Bourne shell (sh) – often used for system administration
  - C shell (csh)
    - T-shell (tcsh) - historically, most commonly used shell on UNIX systems
  - Great information about shells: www.linfo.org/shell.html
- Prompting
  - An active prompt means that the shell is ready for you to type a command.
  - Bash prompt can be customized by the PS1 (Prompt String 1) variable:

    The prompt will display as: [*username@hostname folder*] :

Bash Commands

When a command is typed at the prompt, the Shell processes the command and sends it to the Linux kernel.

- Linux commands are case-sensitive
- Command line structure: Command [options] [arguments]
  - Example: [*netid*@grace1 ~] : **`ls -al /home/user/dir_name`**
    - [*netid*@grace ~] : is the prompt
    - **`ls`** is a command
      - list all the files in the current directory
    - **`-al`** are options
      - options typically starts with dash, changes the way commands work
    - **`/home/user/dir_name`** is an argument
      - arguments - input given to a command to process

Accessing Grace: via SSH

- SSH command is required for accessing Grace via terminal:
  - On campus: ssh userNetID@grace.hprc.tamu.edu
  - Off campus:
    - Set up and start VPN (Virtual Private Network): u.tamu.edu/VPnetwork
    - Then: ssh userNetID@grace.hprc.tamu.edu
  - *Two-Factor Authentication* enabled for CAS, VPN, SSH
- SSH programs for Windows:
  - MobaXTerm (preferred, includes SSH and X11)
  - PowerShell
  - PuTTY SSH
- SSH programs for MacOS:
  - Terminal

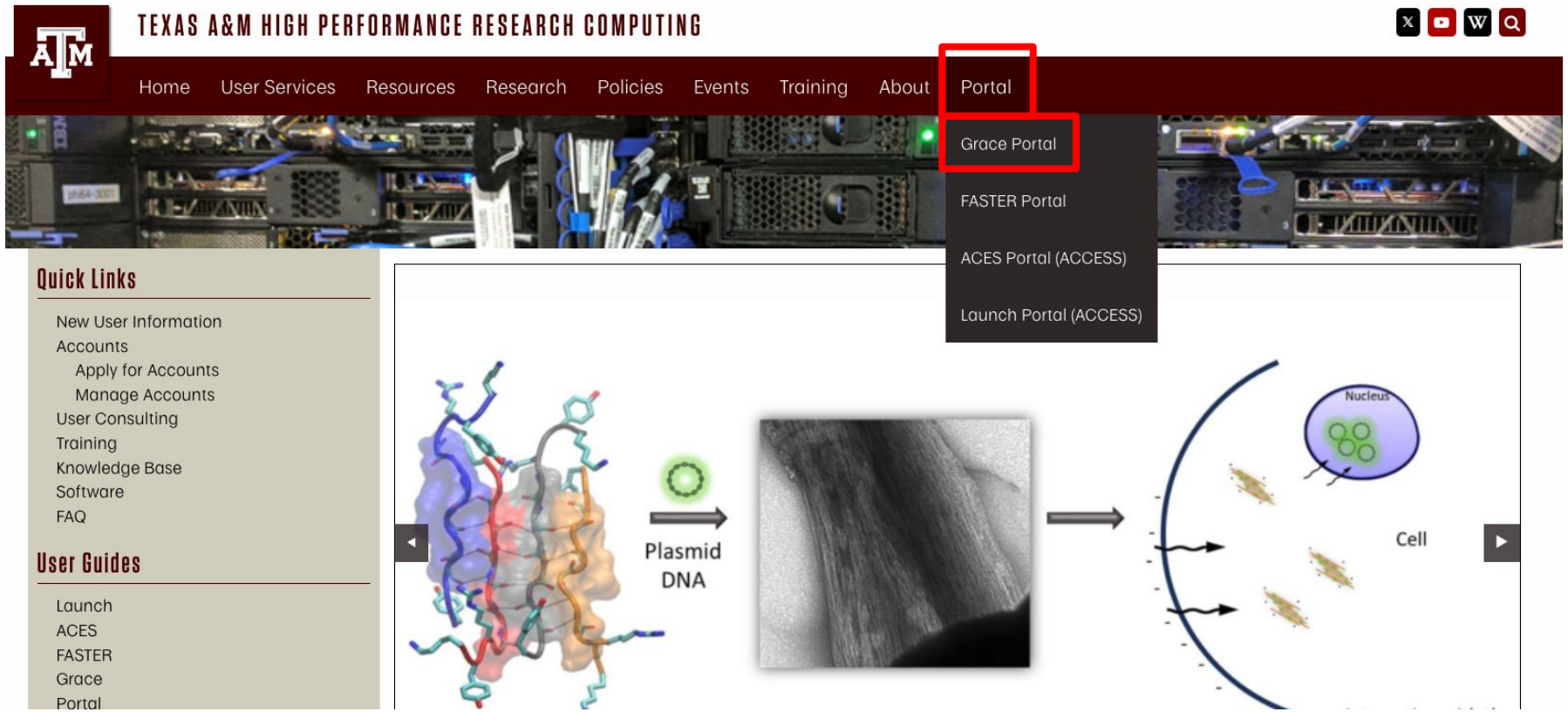
https://hprc.tamu.edu/kb/Helpful-Pages/#ssh

Login sessions that are idle for **60** minutes will be closed automatically
Processes run longer than **60** minutes on login nodes will be killed automatically.
Do not use more than 8 cores on the login nodes!

Do not use the sudo command.

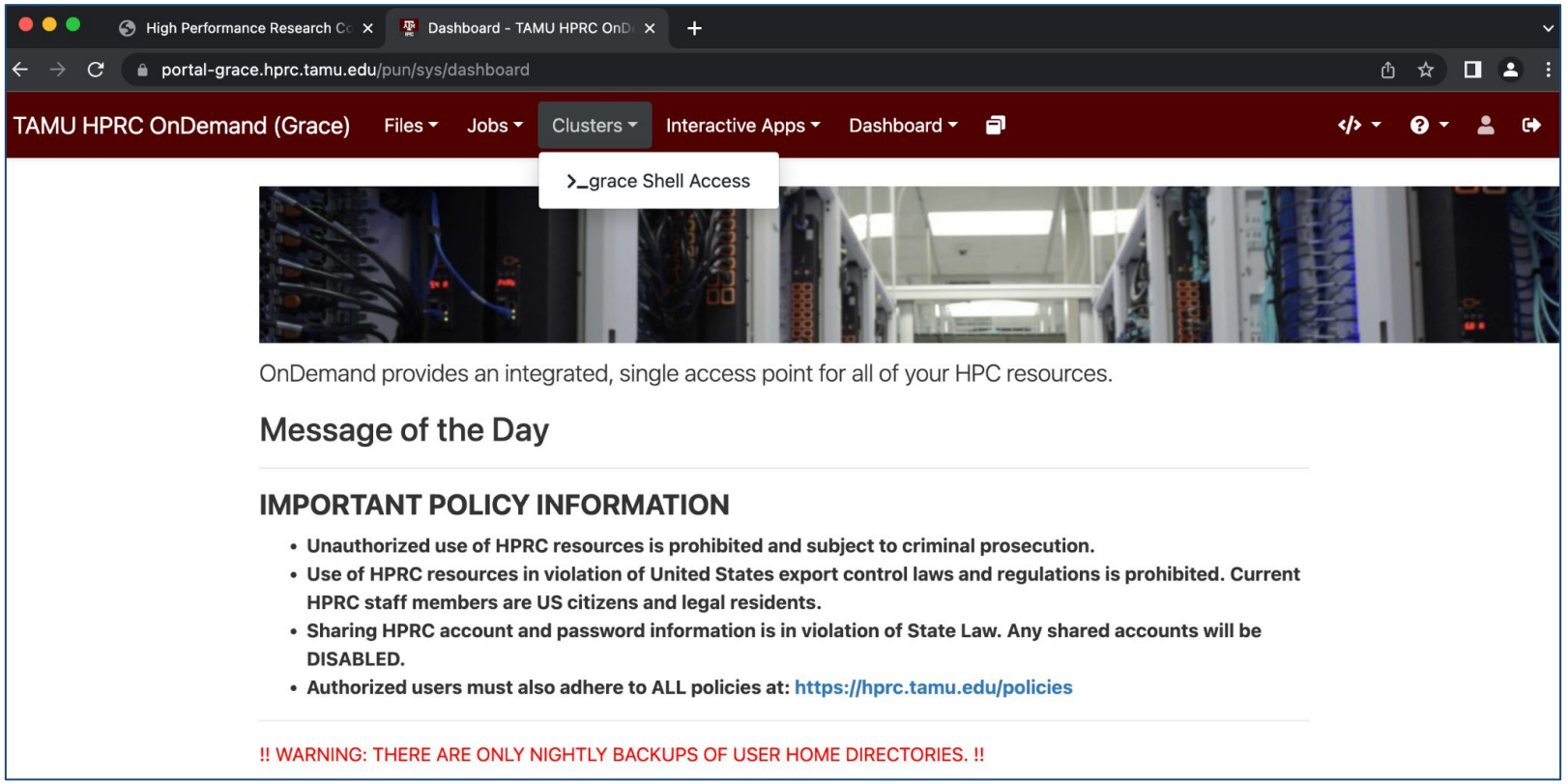
Accessing Grace: via HPRC Portal

- HPRC homepage: hprc.tamu.edu
- Select ‘Grace Portal’ in Portal tab dropdown:

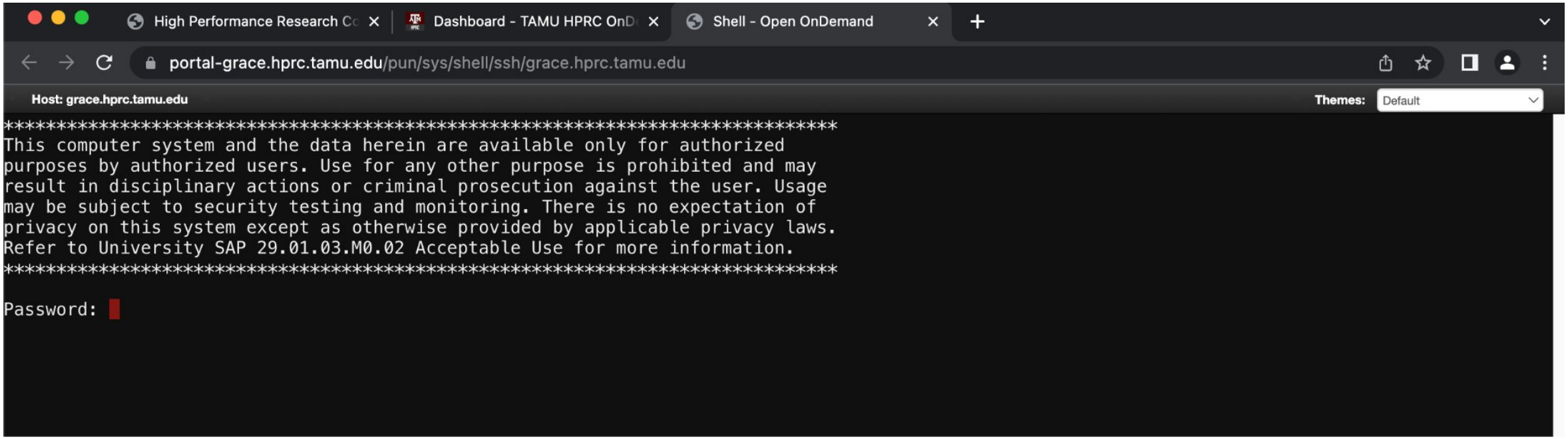


Accessing Grace: via HPRC Portal

- Log in to CAS
- Select '>_grace Shell Access' from Clusters dropdown:



Accessing Grace: via HPRC Portal



Follow Along

Short course material can be found on the short course page.

https://hprc.tamu.edu/training/intro_linux.html

And on disk on Grace

/scratch/training/introduction_linux

Content from our short courses are covered in the relevant Introduction and Primer videos on our Youtube Channel

Texas A&M HPRC YouTube Channel

Hands-on Session

We will demonstrate how to:

- Access Grace via SSH in your preferred way
- Display the current shell name: `echo $0`
- Display the name of the current user: `whoami`
- Remove input/output from previous commands: `clear`

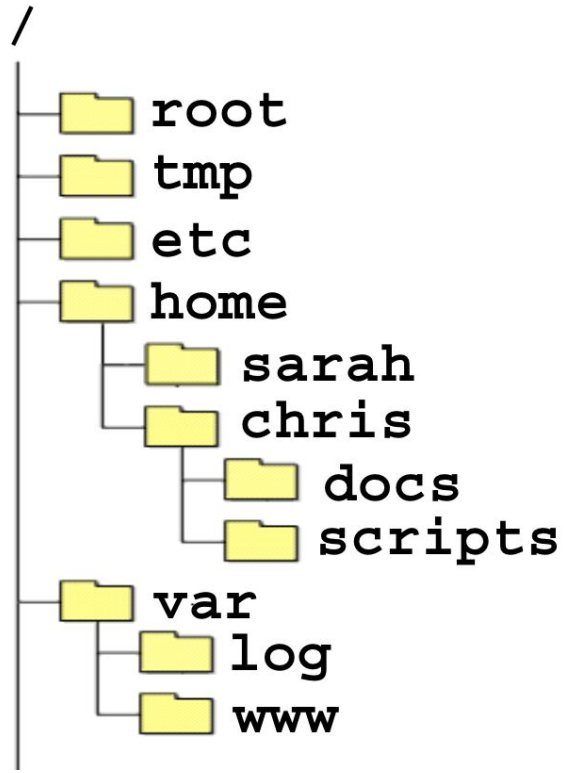
Hands-on Exercise:
Copy training files to your scratch directory

In your terminal, execute

```
$ cd $SCRATCH
$ cp -r /scratch/training/introduction_linux .
```

Managing Directories & Files

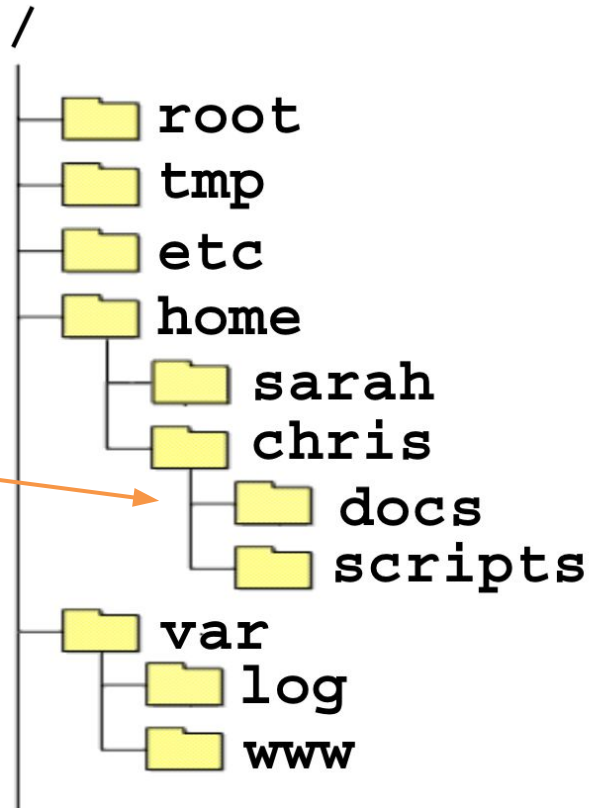
File Hierarchy Structure



/
/root
/tmp
/etc
/home
/home/sarah
/home/chris
/home/chris/docs
/home/chris/scripts
/var
/var/log
/var/www

Navigating the File System

- Most Linux file systems are case-sensitive.
- **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/chris/docs
  - relative pathnames (do NOT start with a /)
    - . current directory
    - .. parent directory
    - ~ home directory



Listing 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 all files including hidden files
    - -l print long listing
    - -C list entries by columns
    - -F print a special character after special files
    - to find all possible flags, use the command: man ls
- **tree** - recursive directory listing

File & Directory Names

```
Commonly used:
A-Z
a-z
0-9
.
-   dash
_   underscore
```

```
Do NOT use:
    spaces or tabs
() parenthesis
" ' quotes
?   Question mark
$   Dollar sign
*   Asterisk
\   back slash
/   forward slash
:   colon
; semi-colon
  ampersand
@ & [ ] ! < >
```

- Do NOT use spaces in the file name
  - ("**`my data file.txt`**" vs "**`my_data_file.txt`**").
- File and directory names are case sensitive
- Avoid creating files on your Windows computer and copying to Linux especially with spaces in the file name

Managing Files & Directories: mkdir

- Making a directory (dir)
  - **mkdir** *dirname* (creates a directory in the current dir)
  - **mkdir** scripts (creates the directory *scripts* in the current dir)
  - **mkdir** ~/scripts (creates the directory *scripts* in your home dir)
  - **mkdir** /home/*netid*/scripts (creates the directory *scripts* in /home/*netid*)

Managing Files and Directories: mv

- Rename a directory
  - **mv** *olddirname newdirname*
- 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*
- 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.

Managing Files and Directories: cp

- 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
- Safe copy
  - **cp -i** *oldfilename newfilename*
    - will prompt you if you are about to overwrite a file named *newfilename*

Managing Files and Directories: cp

- Copying a directory
  - **cp -R** *olddirname newdirname*
    - Makes a complete copy of the directory named *olddirname* including all of its contents, and names it *newdirname* in the current directory
    - the -R flag makes the copying of directories recursive
    - Note: *newdirname* cannot be the name of a directory that already exists

Managing Files and Directories: rm

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

Exercise: Directories & Files

- Change to your home directory
- Print your current working directory
- List contents of the current directory including hidden files
- Make two directories named **temp1** and **temp2** in your current directory
- Show the current directory hierarchy using the **tree** command

Solution: Directories & Files

- Change to your home directory

```
$ cd
```

- Print your current working directory

```
$ pwd
```

- List contents of the current directory including hidden files

```
$ ls -a
```

- Make two directories named **temp1** and **temp2** in your current directory

```
$ mkdir temp1
$ mkdir temp2
```

- Show the current directory hierarchy using the **tree** command

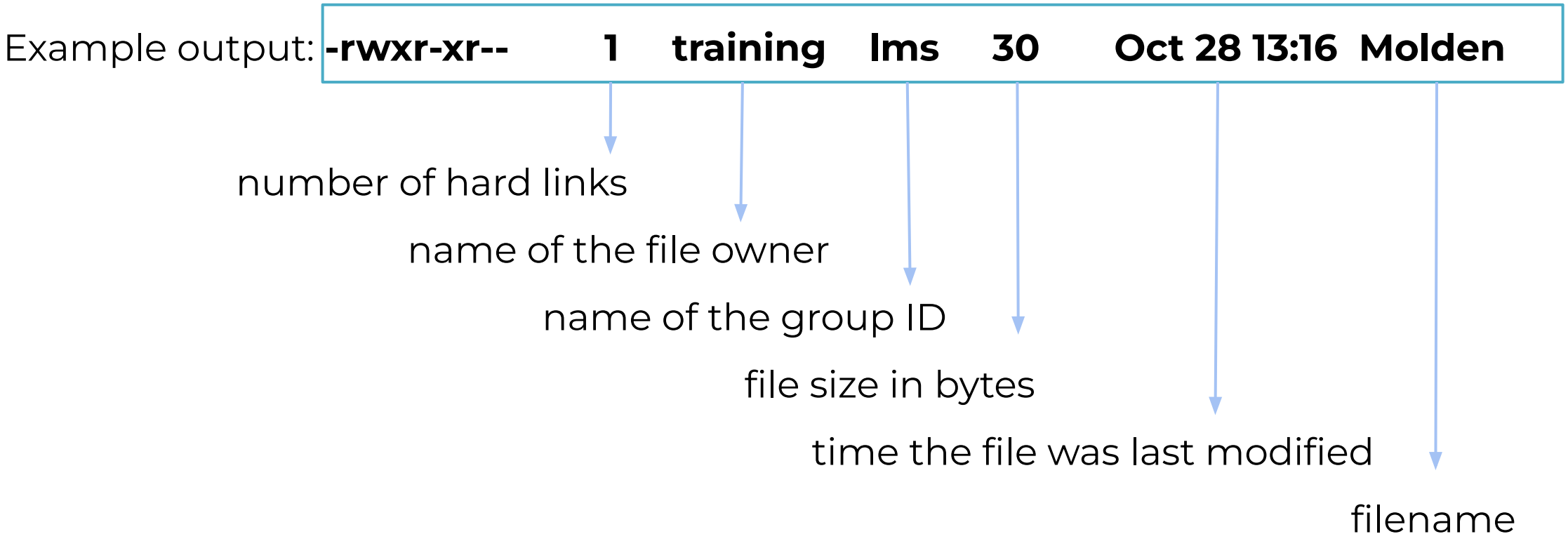
```
$ tree
```

More about Directories & Files

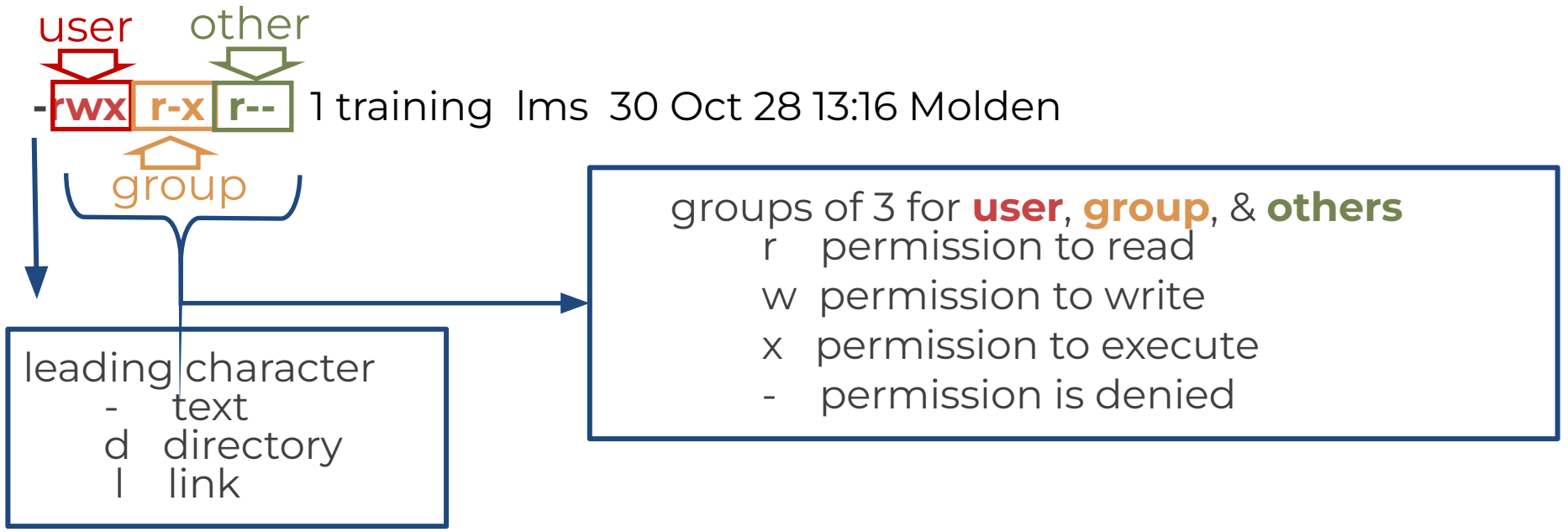
File Attributes

`$ ls -l` lists the files in the dir in **l**ong format

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



File Attributes



Example:
-rwxr-xr-- 1 training lms 30 Oct 28 13:16 Molden
User has read, write and executable permission
Group has read and executable permission but not write permission
Other has read permission but not write or executable permission

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*)
  - removes read permission for group and others
- **chmod -R a+rx** *dirname*
  - gives 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 permission 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

Exercise: Changing Permissions

- Move to the ‘introduction_linux’ directory you copied earlier
- Try to run the script ‘hello_world.sh’

```
$ ./hello_world.sh
```

- Change the permissions of the file to make sure you are able to execute it
- Run the script again

```
$ ./hello_world.sh
```

Solution: Changing Permissions

- Move to the ‘introduction_linux’ directory you copied earlier

```
$ cd /scratch/user/username/introduction_linux
```

- Try to run the script ‘hello_world.sh’

```
$ ./hello_world.sh
```

- Change the permissions of the file to make sure you are able to execute it

```
$ chmod u+x ./hello_world.sh
```

- Run the script again

```
$ ./hello_world.sh
```

Displaying the Contents of a File

Printing ASCII (text) file contents to the screen

- **less** *filename*
- **more** *filename*
- **cat** *filename*
- **cat -A** filename
  - shows hidden characters

- **head -***n filename*
  - n is an integer
  - displays the first *n* lines
- **tail -***n filename*
  - displays the last *n* lines
- **tail -f** *filename*
  - Displays the last 10 lines of a file and waits for new lines, ctrl-c (^c) to exit.

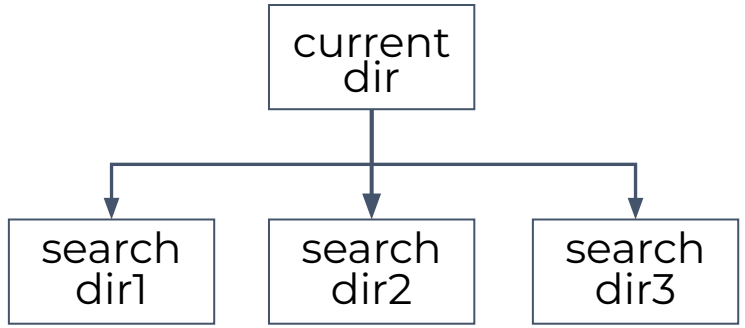
Wildcards (globbing)

*****	* matches any character(s)
?	? matches one character
[...]	matches a single character for a specified range of characters within 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
  - the dir Project1 must already exist 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.

Searching For A File Or Directory



- **find . -name** '*search string*'
  - find . -name '*test1*'
  - 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)

Exercise: Displaying Files, Searching, Wildcards

- Ensure you are still in the '`introduction_linux`' directory
- Use some of the commands we learned to display the text of '`bee_movie.txt`'
- Make a new folder named '`Macbeth`'
- Move all of the files starting with '`macbeth`' to the new directory
- Find all of the files with '`actI`' in the name in the introduction_linux directory

Solution: Displaying Files, Searching, Wildcards

- Make a new folder named '`Macbeth`'

```
$ mkdir Macbeth
```

- Move all of the files starting with '`macbeth`' to the new directory

```
$ mv macbeth* Macbeth/
```

- Find all of the files with '`actI`' in the name in the introduction_linux directory

```
$ find . -name "*actI*"
```

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 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

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 -A *N* Energy run1.out**

  Outputs *N* lines **after** each line containing ‘Energy’

- **grep -B *N* Energy run1.out**

  Outputs *N* lines **before** each line containing ‘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.

Exercise: Searching File Contents

- Find all of the instances of the word ‘honey’ in ‘`bee_movie.txt`’
- Find each line in ‘`macbeth_actIV.txt`’ that contains one or more of the following words: double, cauldron, snake
- Search for the word ‘hurlyburly’ in all files located in the ‘`Macbeth`’ directory. Print the 4 lines before each instance, as well as the 3 lines following each instance

Solution: Searching File Contents

- Find all of the instances of the word 'honey' in '`bee_movie.txt`'

```
$ grep honey bee_movie.txt
```

- Find each line in '`macbeth_actIV.txt`' that contains one or more of the following words: double, cauldron, snake

```
$ egrep 'double|cauldron|snake' Macbeth/macbeth_actIV.txt
```

- Search for the word 'hurlyburly' in all files located in the '`Macbeth`' directory. Print the 4 lines before each instance, as well as the 3 lines following each instance

```
$ grep -R -B 4 -A 3 hurlyburly Macbeth/
```

Get A File From A URL

Use the **wget** command to get a file from a URL

```
$ wget https://hprc.tamu.edu/files/training/DOS_script.sh
```

```
--2022-03-03 21:43:09--  https://hprc.tamu.edu/files/training/DOS_script.sh
Resolving hprc.tamu.edu (hprc.tamu.edu)... 165.91.16.14
Connecting to hprc.tamu.edu (hprc.tamu.edu)|165.91.16.14|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 47 [text/x-sh]
Saving to: 'DOS_script.sh'

100%[======================================================================
====================================>] 47          --.-K/s   in 0s

2022-03-03 21:43:09 (4.82 MB/s) - 'DOS_script.sh' saved [47/47]
```

File Type - CRLF Line Terminators

Windows editors such as Notepad will add hidden Carriage Return Line Feed (CRLF) characters that will cause problems with many applications

```
$ file DOS_script.sh
```

```
DOS_script.sh: ASCII English text, with CRLF line terminators
```

dos2unix command will convert the file to unix format

```
$ dos2unix DOS_script.sh
```

```
dos2unix: converting file DOS_script.sh to Unix format ...
```

```
$ file DOS_script.sh
```

```
DOS_script.sh: ASCII English text
```

Exercise: Directories & Files

- In your home directory, download a file with **wget** from https://hprc.tamu.edu/files/training/DOS_script.sh
- Check the file type with **file**
- Check the file contents
- Convert the file type with **dos2unix**
- Move the DOS_script.sh file to **temp1**
- Show the current directory hierarchy using the **tree** command

Solution: Directories & Files

- Download a file with **wget** from https://hprc.tamu.edu/files/training/DOS_script.sh

```
wget https://hprc.tamu.edu/files/training/DOS_script.sh
```

- Check the file type with **file**

```
file DOS_script.sh
```

- Check the file contents

```
cat -A DOS_script.sh
```

- Convert the file type with **dos2unix**

```
dos2unix DOS_script.sh
```

- Move the DOS_script.sh file to temp1

```
mv DOS_script.sh temp1
```

- Show the directory hierarchy using the **tree** command `tree`

Useful Commands & Tools

Redirecting Input and Output

- **>** 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`
- **<** Redirects input
  - *program***<***inputfile*
  - g16<run1.com
  - output would go to standard out (stdout)
- Redirecting input and output together and running in the background
  - *program***<***inputfilename***>***outputfilename***&**
  - g16<run1.com>run1.log&

Pipes

Pipes **|**

- takes the output of one command and sends it to another
- **ls | more**
- **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, !, ↑, ↓

- **history**
  - The history command will list your last n commands (n = integer).

    !! repeats your last command

    !*n* repeats the nth command

    !*name* repeats the last command that started with name

- You can use the up (↑) and down (↓) arrow keys to scroll through previous commands
- Examples:
  - **history | grep** *wget*
    search history commands that contains wget
  - **history | tail**
    see the last 10 commands

Using Tab for Autocompletion

Tab will try to complete the rest of the file/directory name you are typing

Example:

Type the first few characters of the file name

```
ls my
```

Then hit the **tab key** to autocomplete the file name

```
ls my_favorite_foods.txt
```

Then hit enter to see the command results

If the tab key did not complete the file name then either the file does not exist or there are two or more files that begin with the same characters in which case you need to hit tab twice then type a few more characters and hit tab again 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 -xpvf** *filename***.tar**
  - Extracts the contents of filename.tar
- **tar -cpvf** *filename***.tar** *filenames* (or *dirnames*)
  - Archives filenames and/or dirnames into the file filename.tar
- some of the tar flags
  - **-c** create a new archive
  - **-x** extract files and/or directories from the archive
  - **-p** preserve protection information
  - **-v** verbose
  - **-f** working with files
  - **-z** use the compress program when reading or writing the archive
  - **-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 hierarchy into the file *filename***.zip**
- **unzip** *filename***.zip**
  - Extracts the contents of *filename***.zip**

Customizing the Environment

Bash Environment Variables

- Environment variables store information that is used across different processes in a Linux system.

- Use all caps for Bash Environment variable. **A-Z 0-9 _**
- Use lowercase for the variables that you create. **a-z 0-9 _**

  - **HOME** Pathname of current user's home directory
  - **PATH** The search path for commands.

- Use the **echo** command to see the contents of a variable

```
echo $HOME
```

The Search PATH

- The shell uses the **PATH** environment variable to locate commands typed at the command line
- The value of PATH is a colon separated list of full directory names
- The PATH is searched from left to right. If the command is not found in any of the listed directories, the shell returns an error message
- If multiple commands with the same name exist in more than one location, the first instance found according to the PATH variable will be executed.

```
echo $PATH
```

```
/usr/lib64/qt-3.3/bin:/sw/local/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/usr/lpp/mmfs/bin:/home/netid/.local/bin
```

- Add a directory to the PATH for the current Linux session

```
export PATH=$PATH:/home/netid/bin
```

Customizing the Environment

Two important files for customizing your Bash Shell environment

- **.bashrc** (pronounced dot bashrc)
  - contains aliases, shell variables, paths, etc.
  - sourced (makes the contents file known in the shell) upon starting a non-login shell.
- **.bash_profile** (dot bash_profile)
  - also can contain aliases, shell variables, paths, etc
  - 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**)
  - Re-sources the commands in the .bashrc file

.bash_profile file contents

```
# Get the aliases and functions
        if [ -f  ~/.bashrc ]; then
                .  ~/.bashrc
        fi

# User specific environment and startup programs
        PATH=$PATH:$HOME/.local/bin:$HOME/bin
        export PATH

# Personal aliases
        alias h="history|more"
        alias m="more"


# User specific functions
function  cc()  { awk  -f  cc.awk  "$@".log>"$@".cc  ;  }
```

A line that begins with a # is a comment

Enable settings in .bashrc

Syntax to set a global variable:
export *var_name*=*value*
Specify PATH for all sessions

Add personal aliases

Syntax to create a function:
function *name*() { *command* ; }

If you type **cc test** at the prompt, the following command will be executed:
awk -f cc.awk test.log > test.cc

Editing a text (ASCII) file

There are many editors available under Linux (if they are installed):

- Text mode
  - nano (simple)
  - vi or vim (more advanced)
  - emacs (more advanced)
- Graphic mode (requires remote graphics support - X11)
  - gedit
  - xemacs
  - gvim
- TAMU HPRC options: File editor in the HPRC Portal
- Be aware that a text file edited under Windows editors will most likely add CRLF characters. Use **`dos2unix`** to convert a DOS/Windows edited text file to Unix format.

vi Editor

vi is a text editor that is entirely keyboard-driven (no response to mouse clicks)

- **vi** *filename* - opens (creates) a file using vi
- **vi -R** *filename* or **view** *filename* - opens a file using vi in read-only mode
- To exit a file or save
  - **:q** or **:q!** -quit without saving
  - **ZZ** or **:wq** or **:x** - save the file and exit
- Two modes
  - insert mode
    - **i** one of the commands that initiates insert mode
    - for typing in text
    - all keystrokes are interpreted as text
  - command mode
    - hit the **Esc** key on keyboard to return the user to command mode
    - for navigating the file and editing
    - all keystrokes are interpreted as commands
- More about vi: https://vimhelp.org/

Remote Access and File Transfer

Remote Access

- You need an account on the remote machine (with username and password) to be able to log in as a regular user
- SSH (secure shell) Client: the most common way of remote access
  - Encrypted communication
  - Windows:
    - https://hprc.tamu.edu/kb/Helpful-Pages/#mobaxterm-recommended
  - MacOS:
    - https://hprc.tamu.edu/kb/Helpful-Pages/#macos
- Portal: web platform
  - Example: https://portal.hprc.tamu.edu/
  - login with your HPRC account
  - (Apply for Accounts on TAMU HPRC clusters if you are eligible TAMU applicants)

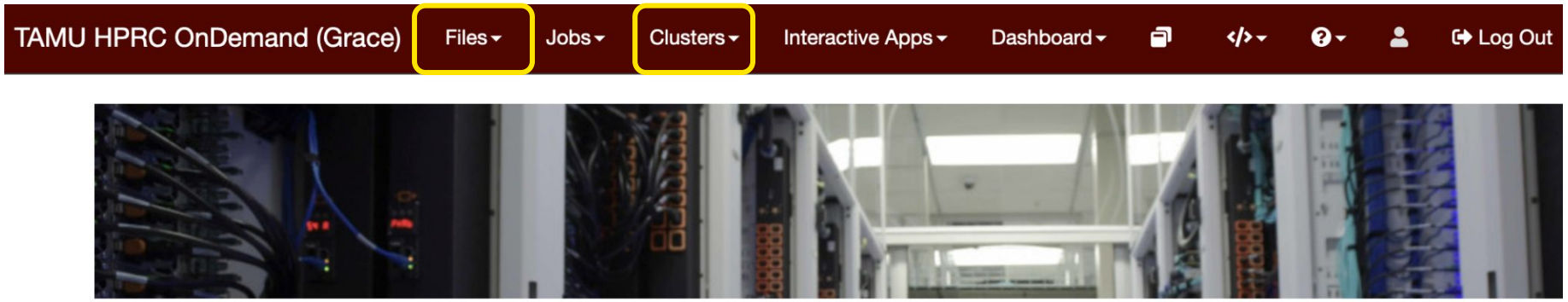
Computer Networking: SSH

Secure Shell (ssh) - Access a remote machine through a secure encrypted protocol

- **ssh** *username***@***remotehostname*
  - **ssh** *remotehostname* (username can be omitted if it is the same on the local and remote machines)
  - 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
- For remote graphics, you will need to ssh with the -X or -Y flag
  - **ssh -X** *netid***@***remotehostname*
- Examples:
  - ssh *netid*@grace.hprc.tamu.edu
  - ssh *netid*@grace
  - ssh grace.hprc.tamu.edu
  - ssh grace

Using the TAMU HPRC Portal

portal.hprc.tamu.edu



OnDemand provides an integrated, single access point for all of your HPC resources.

- **Files** > copy and edit files on the cluster's file systems
- Jobs > submit and monitor cluster jobs
- **Clusters** > open a shell terminal (command line) on a login node
- Interactive Apps > start graphical software on a compute node
- Dashboard > view file quotas and computing account allocations

File Transfer Options

- Command line
  - **scp**
  - **sftp**
  - **rclone**
  - **rsync**
- Use SCP/SFTP Clients with GUI from your computer
  - Examples: MobaXterm, WinSCP, FileZilla
  - Good for small files of less than 2GB
- FTP transfer
- Globus Connect
- Additional options on TAMU HPRC: Portal, HPRC Galaxy for bio-researchers

For more details and options on TAMU HPRC clusters, please visit
https://hprc.tamu.edu/kb/Helpful-Pages/File-Transfer/

Secure copy (scp)

- **scp *local_file username@remote_host:remote_path***
  - Makes a copy of local_file located in the current directory on the local machine to the remote path on the remote host
  - scp will ask you for your password for the remote host
  - Not specifying username will assume that your username is the same on both machines
- **scp *username@remote_host*:remote_path/*remote_file localpath***
  - Copies a file from the remote path on the remote host to the current directory on the local machine
- Useful flags:
  - **-r** recursively copy an entire directory (not suggested)
    - Copies the entire directory hierarchy
    - Links (ie shortcuts) will cause problems
  - **-v** debugging/verbose printing
  - **-p** preserve modification time, access times and modes

Secure File Transfer 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 will put (copy) all of the files that end in .out automatically.
  - **ls** - lists the contents of the remote machine directory
  - **lls** - lists the contents of the local machine directory
  - **pwd** - prints the working directory of the remote machine
  - **lpwd** - prints the working directory of the local machine

Secure File Transfer Protocol (sftp)

- Commands used in the sftp session (continued)
  - **cd** *dirname* - changes the remote machine directory
  - **lcd** *localdir* - changes the local machine directory
  - **mkdir** *dirname* - makes a dir *dirname* on the remote machine
  - **lmkdir** *dirname* - makes a dir *dirname* on the local machine
  - **bye** or **quit** - exits an sftp session.
  - **!***command* - executes a local shell command (i.e. hostname)

[*netid*@grace1 ~]$ **sftp faster.tamu.edu**

sftp> **pwd** — Remote working directory: /general/home/*netid*

sftp> **lpwd** — Local working directory: /home/*netid*

sftp> **bye**

Conclusion

1. Overview
   - what is Linux; bash
2. Managing Directories & Files
   - Relative path and absolute path
   - pwd, cd, ls, mkdir, cp, rm, mv
3. More about Directories & Files
   - File attributes: rwx, chmod; Wild cards
   - more, less, cat, echo, whereis, find, grep, wget, file, vi
4. Useful Commands and Tools
   - redirecting operators > <, pipes |, history, gzip, tar
5. Customizing Environment
   - .bashrc, .profile, $PATH, $HOME, alias
6. Remote Access and File Transfer
   - ssh (-X)
   - scp, sftp

Future Linux Course

Intermediate Linux

Instructor: Wes Brashear

Time: Friday, February 6th, 10:00AM-12:30PM

Location: Blocker 220

Need Help? Contact the HPRC Helpdesk

Website: hprc.tamu.edu
Email: help@hprc.tamu.edu
Phone: (979) 845-0219

Help us, help you -- we need more info

- Which Cluster (Grace, FASTER, Launch, ACES)
- NetID (NOT your UIN)
- Job id(s) if any
- Location of your jobfile, input/output files
- Application used if any
- Module(s) loaded if any
- Error messages
- Steps you have taken, so we can reproduce the problem

Course Survey

https://u.tamu.edu/hprc_shortcourse_survey

https://u.tamu.edu/hprc_shortcourse_survey

HPRC Survey