UT Southwestern Medical Center | BioHPC

Basics of Linux I

The Linux Command Line Interface

[web] portal.biohpc.swmed.edu
[email] biohpc-help@utsouthwestern.edu

Updated for 2021-02-10

So, you wish to learn the art of Linux-Fu…

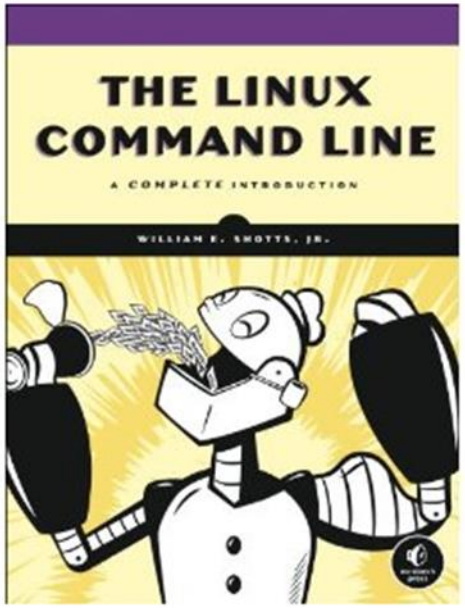
Study Resources: A Free Book

Free, Creative-Commons PDF

On the portal

Training -> Slides & Handouts

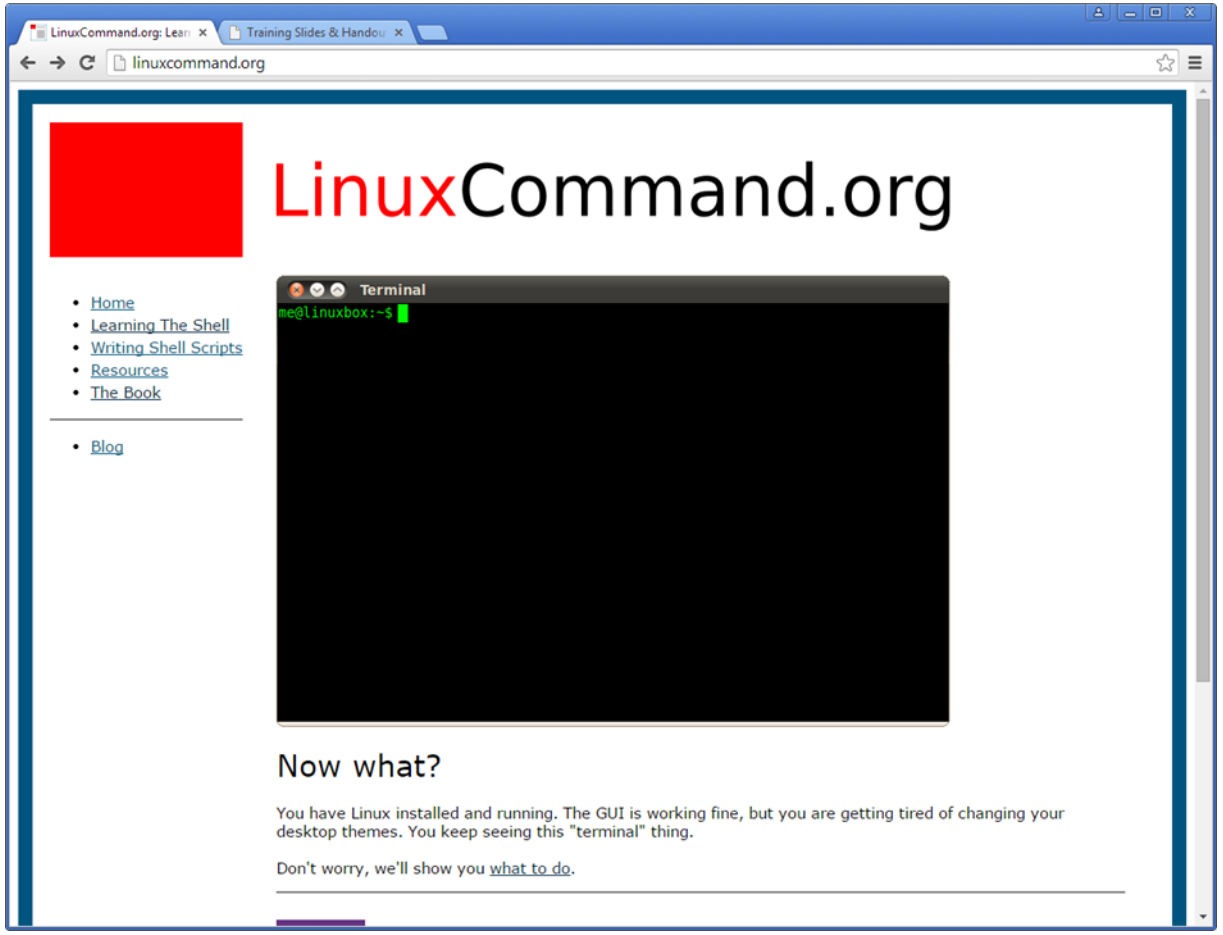
http://linuxcommand.org/tlcl.php



500+ pages

*Some of the materials covered in today's training is from this book

Study Resources: tutorial website



This is a good place to start…

Study Resources: websites

https://linoxide.com/

An all-time favorite. Has all sorts of resources.

https://itsfoss.com/

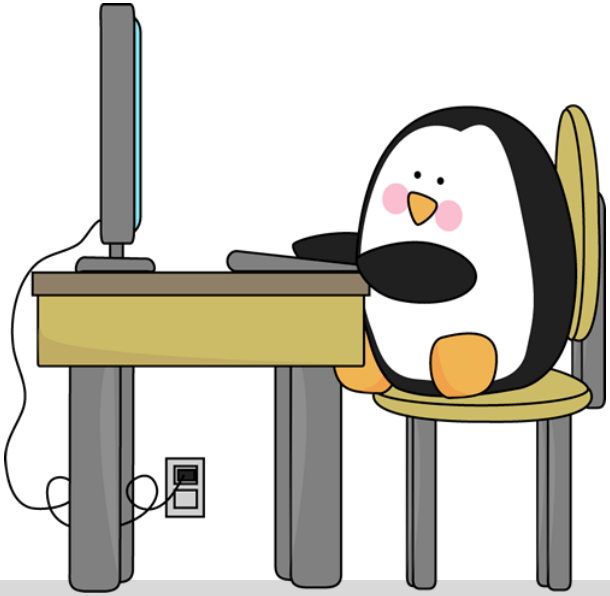
"…an award-winning web-portal that focuses on Open Source in general and Linux in particular."

https://linuxjourney.com/

Great for learning the basics of Linux!

https://wizardzines.com/comics/

Simply awesome!



Study Resources: follow along…

You can follow along using:

1. The Nucleus Web terminal on the BioHPC portal (VPN required):

   https://portal.biohpc.swmed.edu/terminal/ssh/

2. PuTTY, WSL, MobaXterm or any other SSH client* from your PC

3. Terminal from your MacBook

   ssh <username>@nucleus.biohpc.swmed.edu

*https://www.smarthomebeginner.com/best-ssh-clients-windows-putty-alternatives/

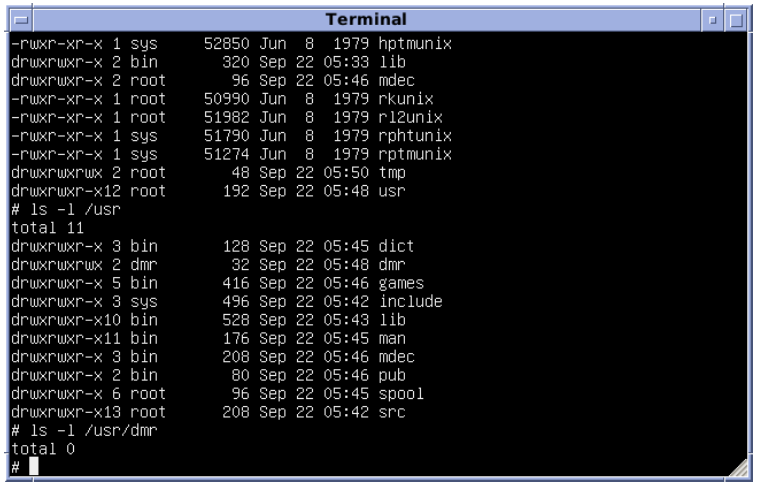
The Terminal

Not too long ago (30+ years ago)…

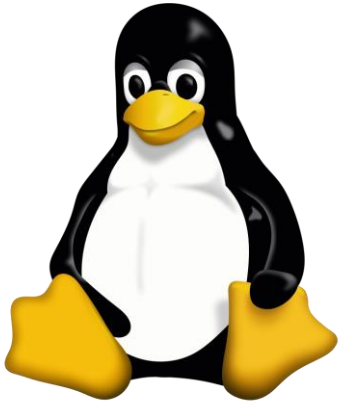
Computers were primarily found in research centers, business, educational institutions, and libraries.

Access points to these computers were called **terminals**:

- Simple **keyboard** and **monitor** interface;
- Computer may be a small, single unit or part of a larger network;
- Many of these computers ran a licensed **UNIX** operating system developed by AT&T.



Modern Unix Descendants



GNU/Linux

Created by:
Linus Torvalds
and
community developers

1991-today

macOS

macOS

Created by:
Apple, Inc.

2001-today

Android

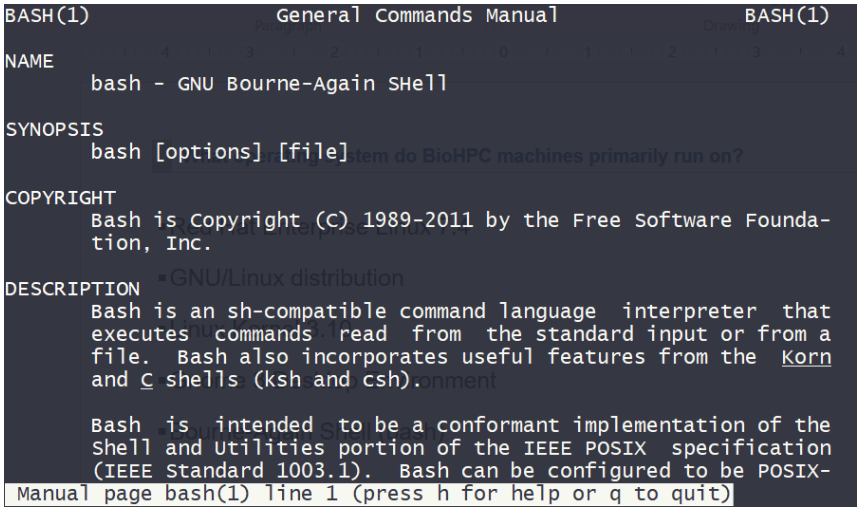
Created by:
Google

Forked from Linux

2008-today

What operating system do BioHPC machines primarily run on?

- Red Hat Enterprise Linux (RHEL) 7.6
- GNU/Linux distribution
- Linux Kernel 3.10
- Gnome 3 Desktop Environment
- Bourne-Again Shell (bash)
- Modular environment
- Slurm Workload Manager



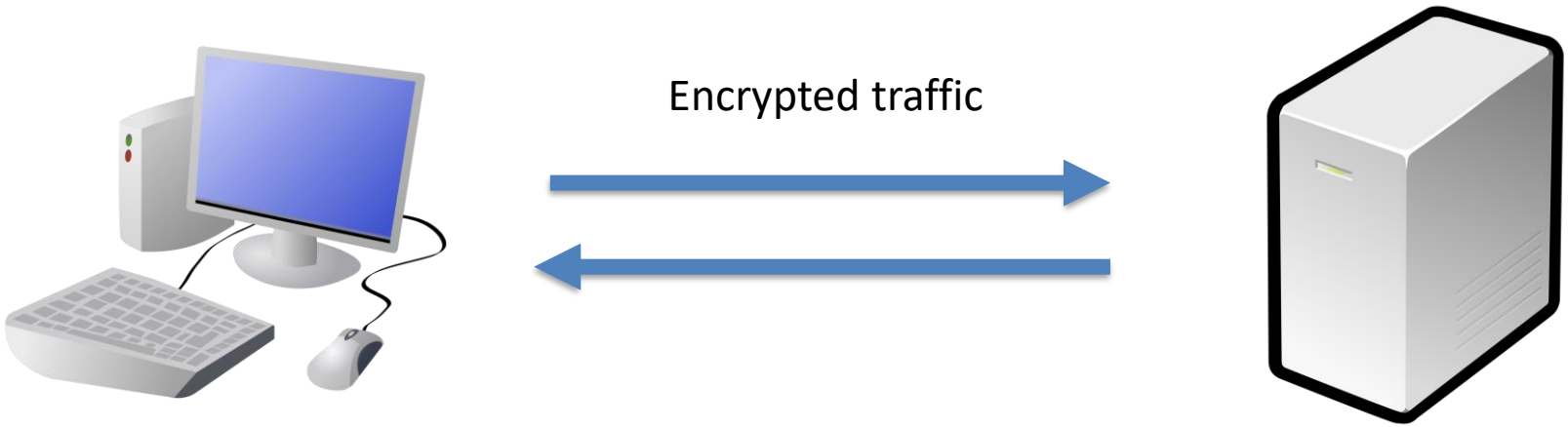
SSH – Secure Shell

Most of your interactions with the **Nucleus** cluster will likely be through SSH.

Most modern GNU/Linux distributions have an **OpenSSH** client installed by default. Mac OS X also has SSH. **PuTTY** is recommended for MS Windows.

Another option on Windows: use **Windows Subsystem for Linux (WSL)**.

```
$ ssh s191529@nucleus.biohpc.swmed.edu
```



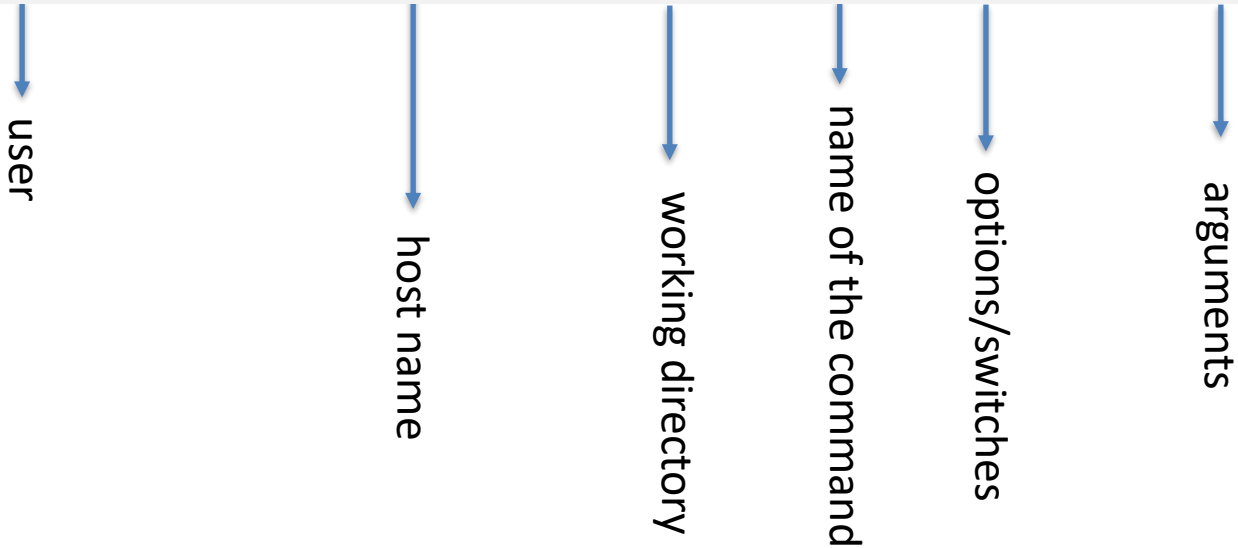
The Text (Command-Line Interface) Shell

The interaction between user and the operating system is provided by a **shell.**

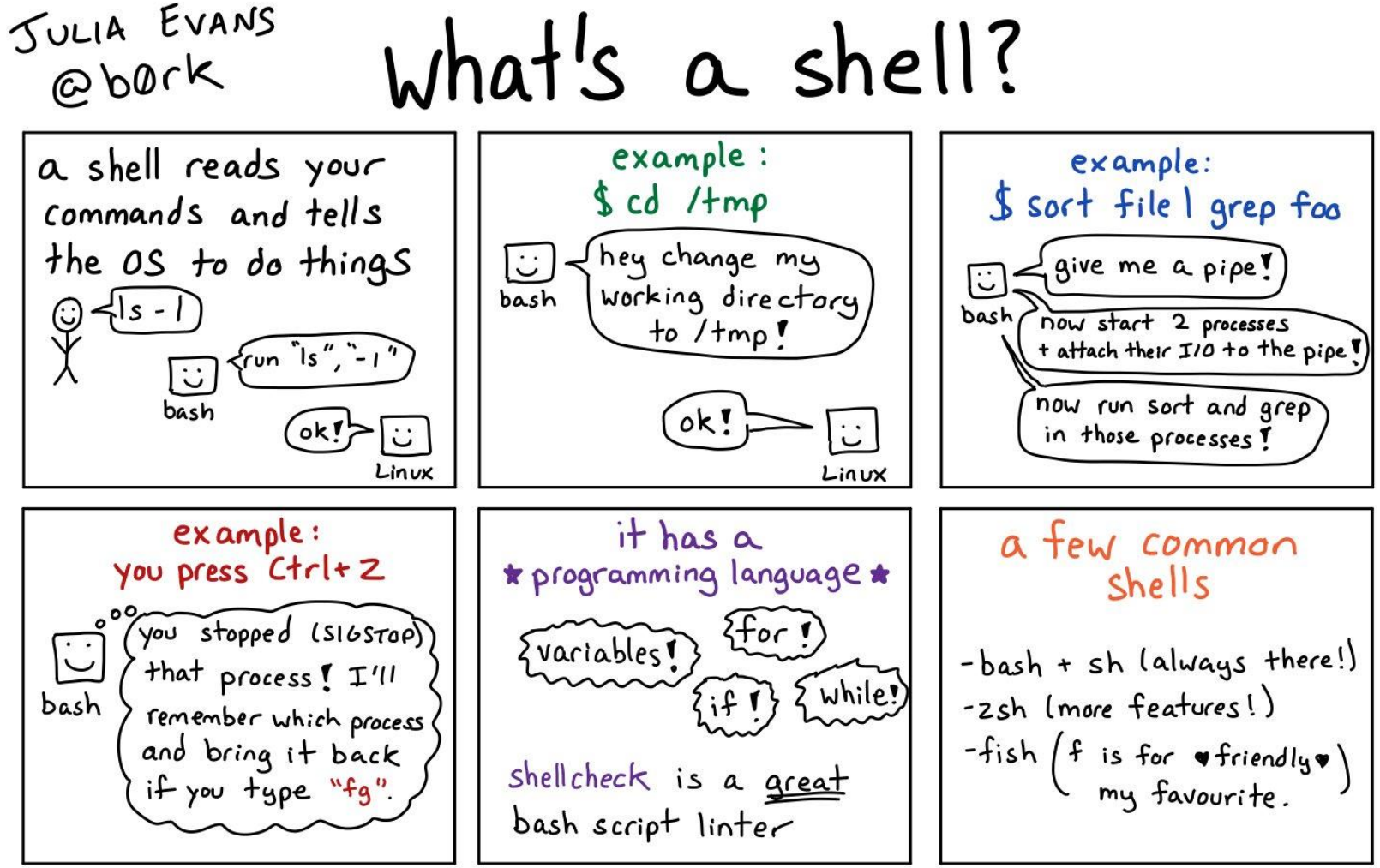
The shell accepts keyboard commands and hands them off to the operating system.

The BioHPC default shell is *bash* – the *Bourne-Again Shell*.

```
[s191529@Nucleus005 ~]$ echo -n Hello, world!
```



About the shell



https://twitter.com/b0rk/

Logging into Nucleus – Where Am I?

```
[s191529@Nucleus005 ~]$ pwd
```

pwd – print working directory

```
/home2/s191529
```

ls – list contents of a directory

```
[s191529@Nucleus005 ~]$ ls
```

```
[s191529@Nucleus005 ~]$ ls /home2/s191529
```

```
[s191529@Nucleus005 ~]$ ls ~
```

```
[s191529@Nucleus005 ~]$ ls .
```

Study Resources: man pages

Get a command's help page: **man <command>**

```
[s191529@rhel7vm ~]$ man ls
```

Press **q** to exit the man page

Filenames that start with **.** are hidden. You can view them however with the **ls** command and pass the **-a** flag to it (**a** for all).

Try some other Linux commands and see what they output:

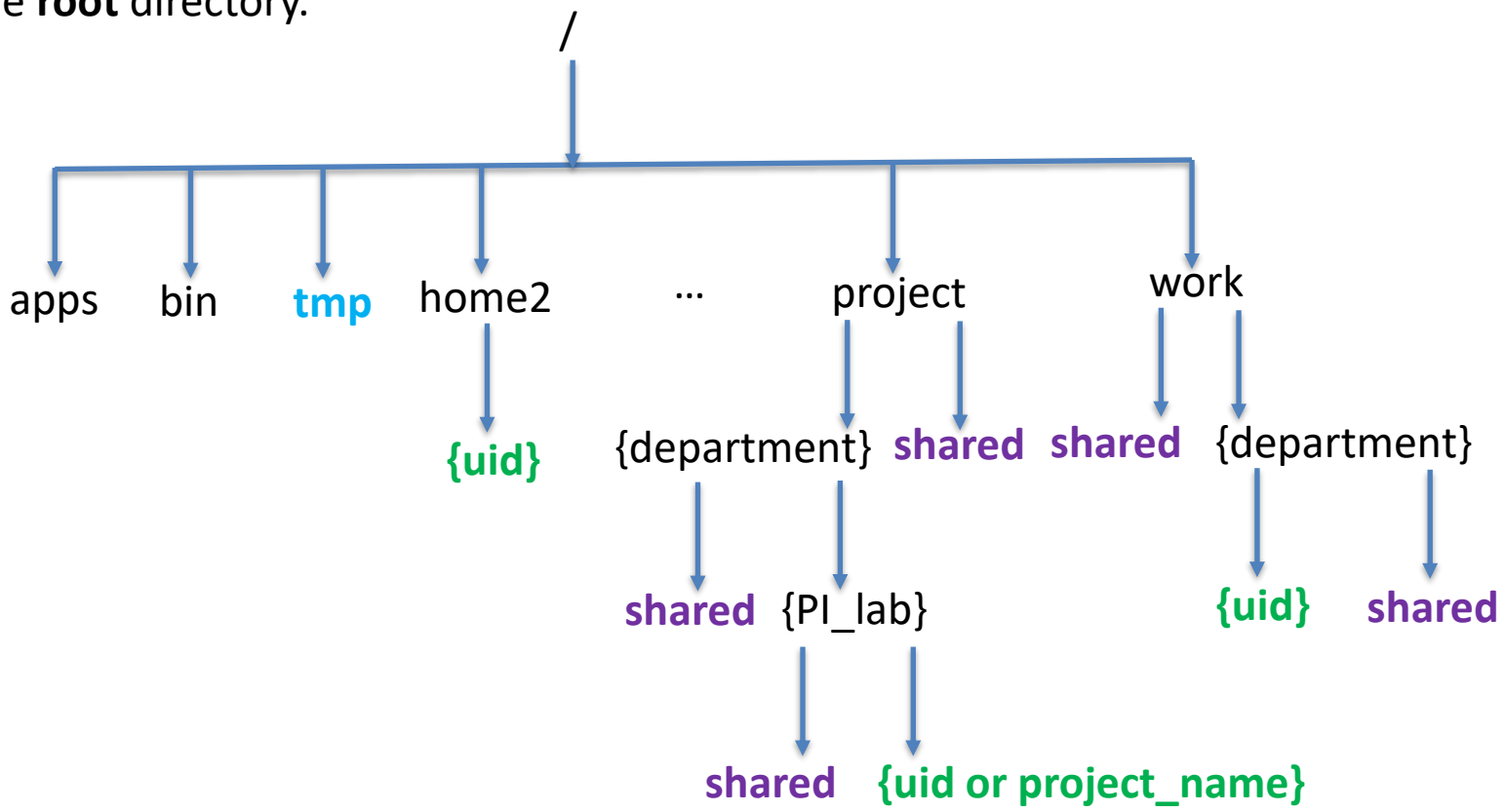
```
[s191529@rhel7vm ~]$ date
```

```
[s191529@rhel7vm ~]$ whoami
```

```
[s191529@rhel7vm ~]$ echo Hello World!
```

Linux Basics: The File System

Everything in Linux is a file. Keep this in mind. Files on a Linux system are arranged in a **hierarchical directory structure**. The first directory in the filesystem is named the **root** directory.



Navigating the file system

How does one change his/her working directory?

cd – change directory

```
[s191529@Nucleus005 ~]$ cd /work/biohpcadmin/s191529/
```

```
[s191529@Nucleus005 s191529]$ cd ..
```

```
[s191529@Nucleus005 biohpcadmin]$ cd s191529/
```

Shortcuts to help you out:

- **.** This is the directory you are currently in.
- **..** Takes you to the directory above your current one.
- **~** This directory defaults to your home directory.
- **-** This will take you to the previous directory you were just at.
- Finally, the **Up Arrow** brings the last command you hit.

Linux Command Line: Files and Directories

Files and directories may be referenced by an absolute or relative path

- **Absolute path**—specify the location of a file or directory from / (the root directory)

```
[s191529@Nucleus005 ~]$ cd /project/biohpcadmin
```

Pros: you know exactly where you are going!

Cons: tedious if there are many nested folders.

- **Relative path**— paths relative to your working directory.

```
[s191529@Nucleus005 biohpcadmin]$ cd s191529
```

```
[s191529@Nucleus005 s191529]$ cd ..
```

Determining your storage quota

```
[s191529@Nucleus005 ~]$ quota -s
```

```
Disk quotas for user s191529 (uid 191529):
    Filesystem  space  quota  limit  grace  files  quota  limit  grace
lysosomehome:/home2
             21581M  51200M  71680M          153k      0      0
```

```
[s191529@Nucleus005 ~]$ lfs quota -g 1001 /project -h
```

```
Disk quotas for grp 1001 (gid 1001):
    Filesystem   used  quota  limit  grace  files  quota  limit  grace
      /project  17.59T      0k     0k      - 12021829      0      0      -
```

How does one find the **gid**?

```
[s191529@Nucleus005 ~]$ id 191529
```

How much storage is a directory occupying?

```
[s191529@Nucleus005 ~]$ ls -l Documents/misc/
```

How much space does this directory, and all its contents use?
du – disk usage (**-h** – human readable; **-s** – summarize)

```
[s191529@Nucleus005 ~]$ du -hs Documents/misc/
```

How can I create a new (empty) file?
touch

```
[s191529@Nucleus005 misc]$ touch myfile.txt
```

The command **touch** can also be used to change timestamps.

What kind of file a file is?
file

```
[s191529@Nucleus005 misc]$ file myfile.txt
```

In Linux, file extensions aren't required.

Exploring the file system

```
[s191529@Nucleus005 ~]$ cd /project/shared/biohpc_training
```

Let's concatenate the contents of a file to the standard output of the terminal.
In other words, let's print to the terminal:

```
[s191529@Nucleus005 biohpc_training]$ cat c475_r0ck_4m_1_r16h7.txt
```

Notice: not all files have an extension:

```
[s191529@Nucleus005 biohpc_training]$ file RJ_WS
```

Wish to clear the terminal?

```
[s191529@Nucleus005 biohpc_training]$ clear
```

```
[s191529@Nucleus005 biohpc_training]$ reset
```

Bash has a very useful auto-completion shortcut for typing commands more quickly.

Give it a try!

Type:

```
cd /project/shared/biohpc_training
cat c475_r0ck_4m_1_r16h7.txt
```

Viewing large text files

A file does not have to be very large before concatenating them to the standard output becomes unhelpful. The file extension has a **.fastq.gz** file extension, but what does **file** produce?

```
[s191529@Nucleus005 biohpc_training]$ file HD728.R1.fastq.gz
```

The file is a compressed file – its contents are unreadable to us. Let's decompress the file first using **gzip**.

```
[s191529@Nucleus005 biohpc_training]$ file HD728.R1.fastq.gz
```

```
[s191529@Nucleus005 biohpc_training]$ gzip -cd HD728.R1.fastq.gz > HD728.R1.fastq
```

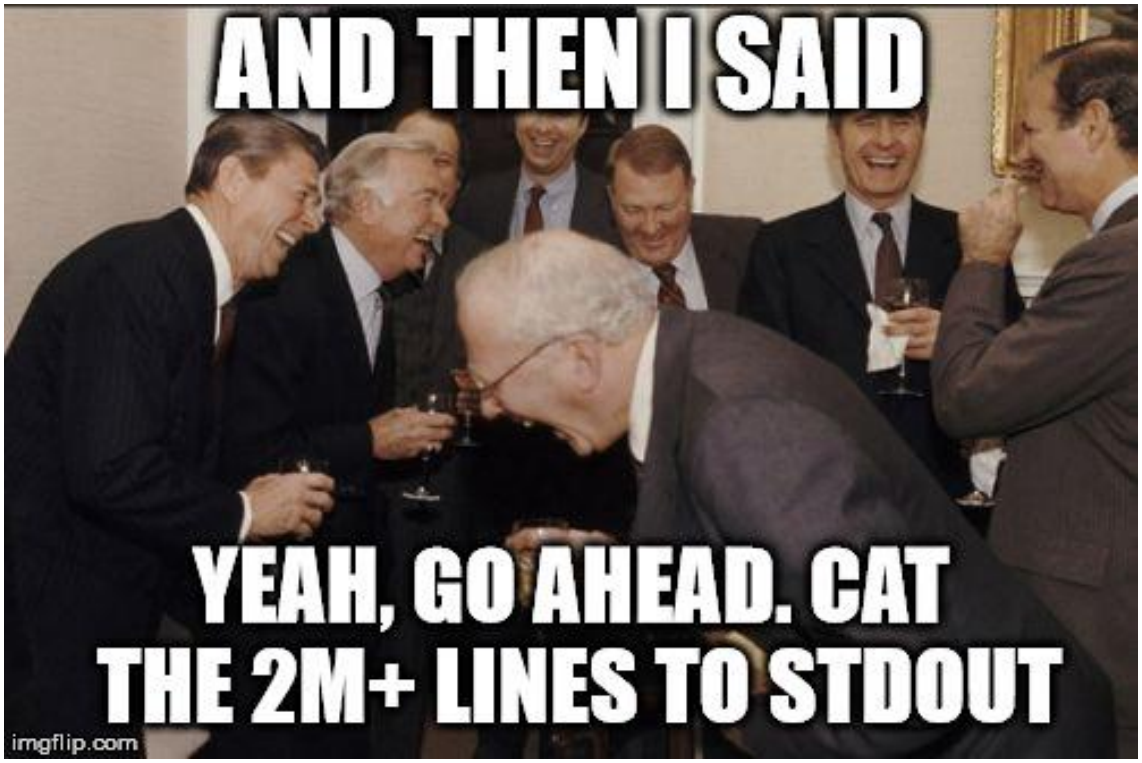
Exercise
Using the program **wc**, count how many lines of text are inside **HD728.R1.fastq**? How would one access information on how to use this program?

Viewing large text files

```
[s191529@Nucleus005 biohpc_training]$ wc -l HD728.R1.fastq
```

Keep in mind that **cat** it's only meant for short content. Trying to **cat** 2M+ lines to the standard output is going to be a bit problematic. Let's try it anyway…

Interrupting a Running Program



What happens if I need to kill a program that is running?

Pressing **CTRL + C** will send an interruption signal (SIGINT) to the program which usually kills it. If not…

```
[s191529@Nucleus005 biohpc_training]$ man kill
```

Head, Tail, More, Less

Not always practical to print an entire file to the shell. Use these commands:
head – print the first 10 lines of each file to the standard output
tail – print the last 10 lines of each file to the standard output

```
[s191529@Nucleus005 biohpc_training]$ head HD728.R1.fastq
```

Exercise
Print the first 50 lines of **HD728.R1.fastq**! Hint: **man head**

You can navigate through a text file page by page with **less**:

```
[s191529@Nucleus005 biohpc_training]$ less HD728.R1.fastq
```

To navigate through **less**:

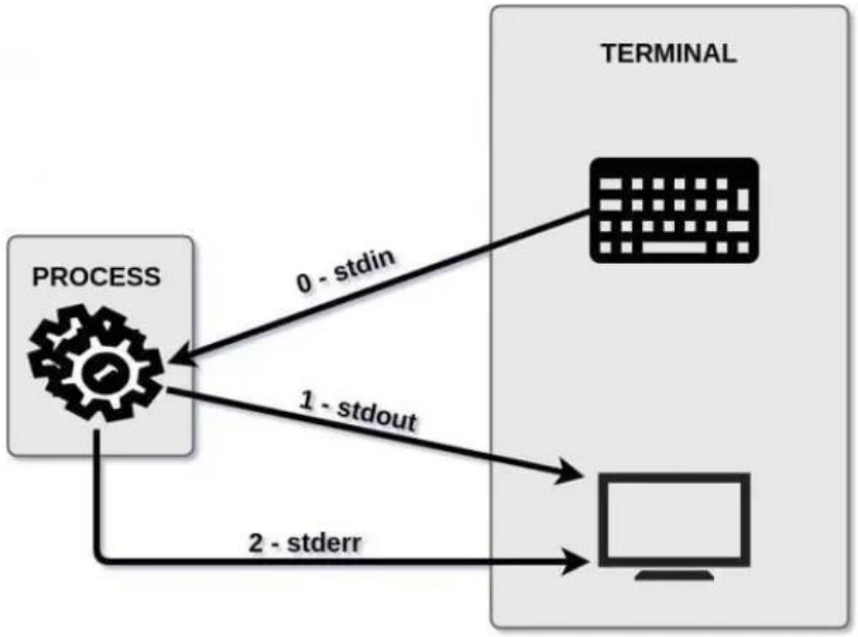
- **q** – to quit out of less
- **Page up/down**, **Up/Down** – to navigate
- **g/G** - moves to the beginning/end of the text file
- **/text** - search for specific **text**
- **h** – If you need a little help about how to use less while you're in less, use help.

Standard Streams

Streams are usually connected to the terminal in which they are executed, but that can be changed using **redirection operators** and/or **pipes**.

Redirection operators are a subset of control operators. They allow you to direct the input or output (stream) of your command.

The **pipe** operator is used to pass the output of a command to the input of another command. The vertical bar (**|**) represents this operator.



Redirection Operators

A simple example of a program that uses **standard input** is the **cat** command. Standard input can also come from an input file:

```
[s191529@Nucleus005 biohpc_training]$ cat ~/.bashrc
```

You can use **input redirection** (represented by **<**) to achieve the same results as above:

```
[s191529@Nucleus005 biohpc_training]$ cat  < ~/.bashrc
```

You can redirect **standard output** to a file (represented by **>**). This is useful if you want to save the output for later use, or as a log of a script:

```
[s191529@Nucleus005 biohpc_training]$ cat  ~/.bashrc > bashrc.txt
```

Use the **output append operator** (represented by **>>**) if you want to append to an existing file:

```
[s191529@Nucleus005 biohpc_training]$ stat  ~/.bashrc >> bashrc.txt
```

File Descriptors

Linux often represents the three standard streams as file descriptors:

File Descriptor	Name	Standard Stream
0	Standard Input	stdin
1	Standard Output	stdout
2	Standard Error	stderr

Let's try the **standard error**:

```
[s191529@Nucleus005 biohpc_training]$ ls -l /bin/usr
```

We can redirect the **standard error** to a file:

```
[s191529@Nucleus005 biohpc_training]$ ls -l /bin/usr 2> error.txt
```

You can redirect **stderr** and **stdout** to a single file (two ways):

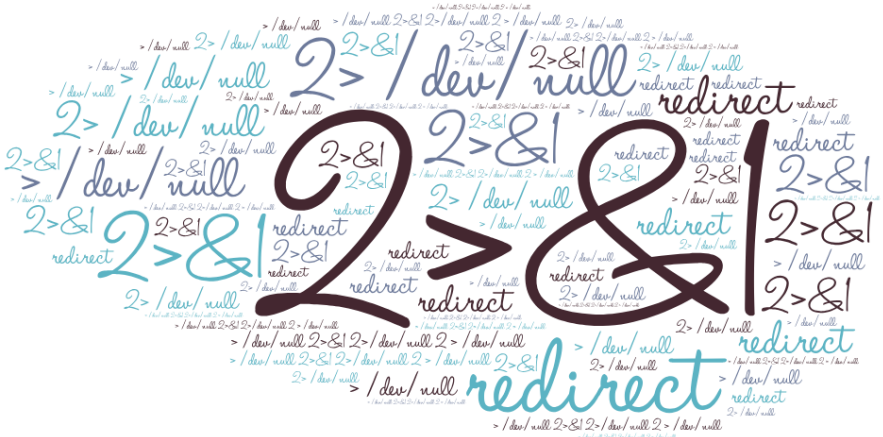
```
[s191529@Nucleus005 biohpc_training]$ ls -l /bin/usr > error.txt 2>&1
[s191529@Nucleus005 biohpc_training]$ ls -l /bin/usr &> error.txt
```

Redirecting to /dev/null

What if **I don't care at all** about the **stdout** and **stderr**?

```
[s191529@Nucleus005 biohpc_training]$ ls -l /bin/usr > /dev/null 2>&1
[s191529@Nucleus005 biohpc_training]$ ls -l /bin/usr &> /dev/null
```

*"To begin, **/dev/null** is a special file called the null device in Unix systems. Colloquially it is also called the **bit-bucket** or the **blackhole** because it immediately discards anything written to it and only returns an end-of-file (**EOF**) when read."*



Text Editors

Vim

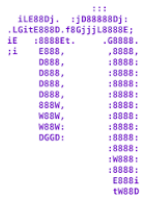
Cryptic commands! Cheat sheet on the portal.

Quick tutorial: http://www.washington.edu/computing/unix/vi.html

Emacs

An extensible, customizable text editor.

Quick tutorial: http://www.gnu.org/software/emacs/tour/



nano

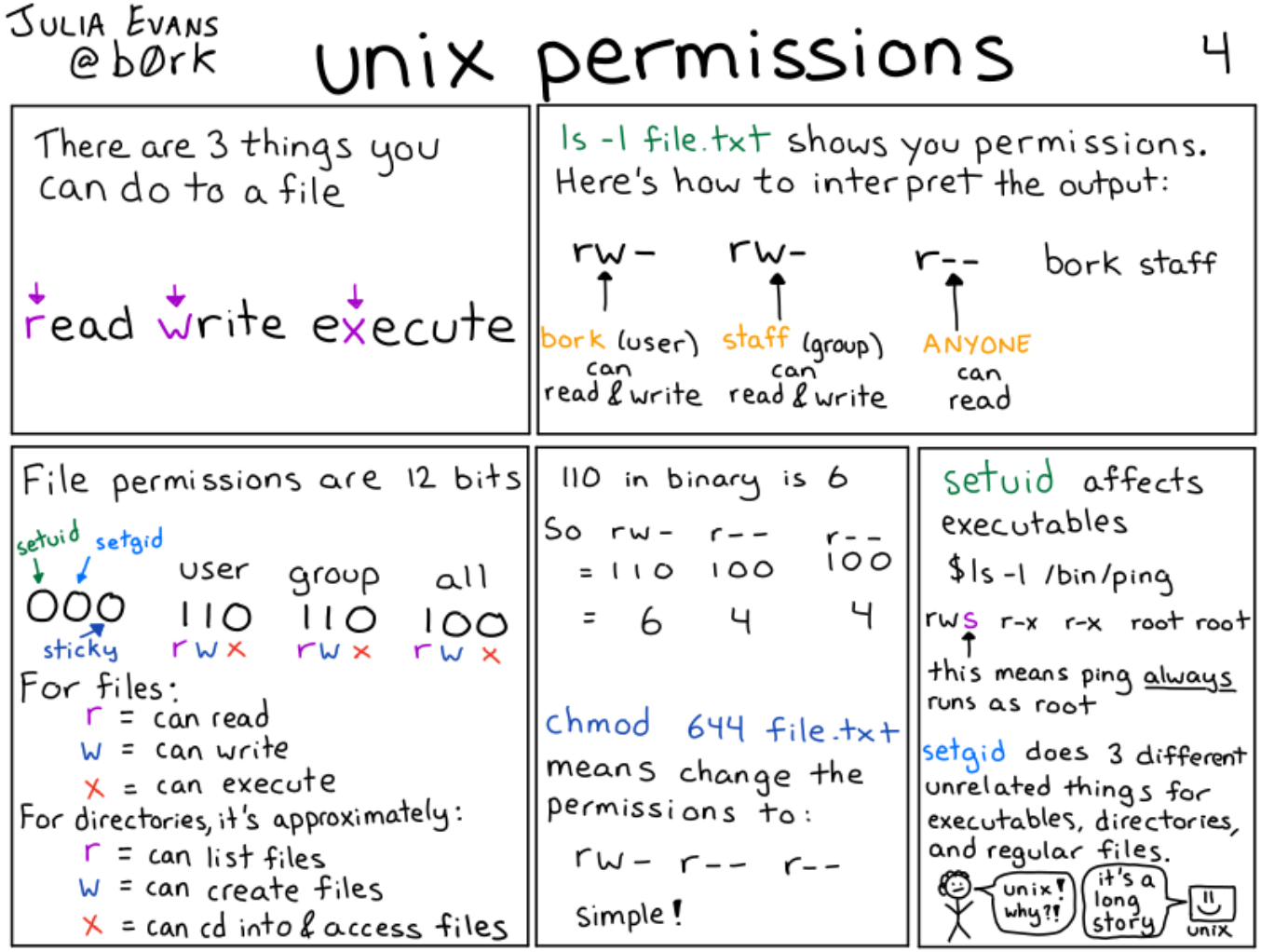
Easier to use.

Quick tutorial: http://mintaka.sdsu.edu/reu/nano.html

Any text editor from your PC or Mac.

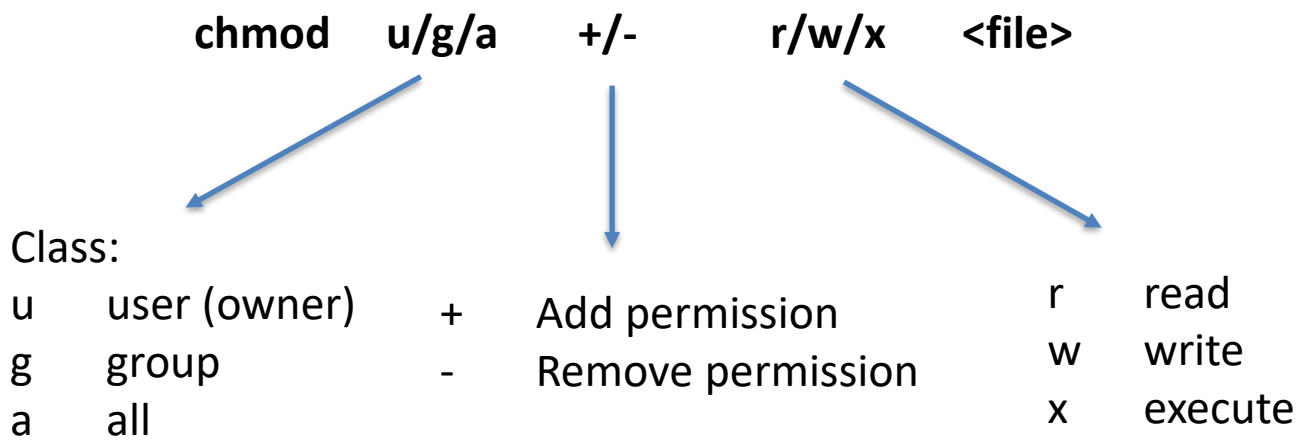
Mount your directories as network drives:
https://portal.biohpc.swmed.edu/content/guides/biohpc-cloud-storage/

Permissions



https://wizardzines.com/comics/permissions/

Permissions



Examples:

```
chmod g+rw script.sh     # Add read/write permissions for the group
chmod a+x script.sh      # Add execute permission for everyone
chmod g-x script.sh      # Remove execute permission for the group

chmod 700 script.sh      # ?
chmod 640 script.sh      # ?
```

Copying data

First, let's create an empty directory with **mkdir**:

```
[s191529@Nucleus005 biohpc_training]$ mkdir -p
/project/biohpcadmin/shared/cuda_samples
```

Copy everything recursively (**-r**) from **source** to **destination**:

```
[s191529@Nucleus005 biohpc_training]$ cp -r  ~/cuda_samples/*
/project/biohpcadmin/shared/cuda_samples/
```

We can copy the entire folder recursively:

```
[s191529@Nucleus005 biohpc_training]$ cp -r  ~/cuda_samples
/project/biohpcadmin/shared/cuda_samples
```

If you copy a file over to a directory that has the same filename, the file will be overwritten with whatever you are copying over. You can use the **-i** flag (interactive) to prompt you before overwriting a file. By default, **cp** will apply your ownership and primary group to files.

Moving data

Very similar to the copy command. You can rename a file (or a directory) with **mv**:

```
[s191529@Nucleus005 biohpc_training]$ mv foo.txt blah.txt
```

And of course, we can move things from **source** to **destination**:

```
[s191529@Nucleus005 biohpc_training]$ mv  blah.txt foo.bar /somedir
```

If you don't want to overwrite anything:

```
[s191529@Nucleus005 biohpc_training]$ mv -i foo.txt blah.txt
```

Note that **mv** will attempt to preserve original permissions. You can also make a backup of that file and it will just rename the old version with a **~**:

```
[s191529@Nucleus005 biohpc_training]$ mv -b /somedir /newdir
```

Deleting Files

- Be very cautious of your ability to destroy files!
- There is **no Recycling Bin** to restore your files.
- Once files are deleted by the CLI, it is generally very difficult to recover them.
- Make sure important data is backed up! The command to remove things is **rm**, and it's very similar to **cp** and **mv**.

To delete everything in a folder:

```
[s191529@Nucleus005 biohpc_training]$ rm somedir/*
```

To delete a folder recursively:

```
[s191529@Nucleus005 biohpc_training]$ rm -r somedir
```

Try deleting things interactively (recommended):

```
[s191529@Nucleus005 biohpc_training]$ rm -i somedir/*
```

Wildcards

***** Match any number of characters:

```
ls *                Any file
ls notes*           Any file beginning with notes
ls *.txt            Any file ending in .txt
ls *2019*           Any file with 2015 somewhere in its name
```

? Match a single character:

```
ls data_00?.txt     Matches data_001, data_002, data_00A, etc.
```

[] Match a set of characters (bracket expression):

```
ls data_00[0123456789].txt
ls data_00[0-9].txt     Matches data_001 – data_009, not data_00A
```

History

There is a history of the commands that you previously entered. This is useful as you can look through these commands:

```
[s191529@Nucleus005 biohpc_training]$ history
```

To run the previous command without typing it again, hit **!!**. Another history shortcut is **Ctrl-R**, this is the reverse search command, if you hit **Ctrl-R** and you start typing parts of the command you want it will show you matches and you can just navigate through them by hitting the **Ctrl-R** key again. Once you found the command you want to use again, just hit the **Enter** key.

To find out what a command does, try using **whatis**:

```
[s191529@Nucleus005 biohpc_training]$ whatis cat
```

Environmental Variables – Controlling the behavior of the Shell

Several variables control the behavior of the shell. You can print all these variables with:

$ env

Or print them individually:

$ echo $SHELL
/bin/bash

$ echo $HOME
/home2/s191529

$ echo $USER
s191529

$PATH variable is one of the most important and tells the shell where your programs are:

$ echo $PATH
/home2/s191529/.local/bin:/cm/shared/apps/slurm/16.05.8/sbin:/cm/shared/apps/slurm/16.05.8/bin:/usr/local/bin

The module system on BioHPC modifies this **$PATH** so that programs are made available to the user. One can also manually edit their **$PATH**

$ export
PATH=/home2/s191529/bin:$PATH

Overview of commands used

Command	Full Name	Description
man	manual	man <command> opens manual for a command
ssh	secure shell	opens a remote shell on a server
echo	echo	prints statement to standard output
pwd	print working directory	prints current working directory
cd	change directory	change to specified directory
ls	list	list contents of a directory
file	file	determines type of file
cat	concatenate	concatenates files to standard output
head	head	prints the top n-lines of a text file
tail	tail	prints the bottom n-lines of a text file
history	command history	outputs previously hit commands
less	less	like more, but allows backwards traversal of a file
du	disk usage	calculate disk usage of a file or folder
vi	vi text editor	simple text editor
cp	copy	copy a file or directory from a source to a destination
mv	move	moves a file from a directory from a source to a destination
rm	remove	deletes a file or a directory
chmod	change mode	modifies permissions of a file or directory

That's it for today! Questions?

KEEP
CALM
AND
LEARN
LINUX