Introduction to Unix

Anthony Armstrong
Bioinformatics and Computational Biosciences Branch
Office of Cyber Infrastructure and Computational Biology

25 February 2021

Today's Instructor

Anthony Armstrong, PhD

Computational Structural Biology Specialist

- Bioinformatics and Computational Biosciences Branch (BCBB), NIAID
- National Institutes of Health, Bethesda, MD USA
- Contact our team:
  - Email: bioinformatics@niaid.nih.gov
  - Instructor: anthony.armstrong@nih.gov

NIH National Institute of Allergy and Infectious Diseases

NIAID

Outline

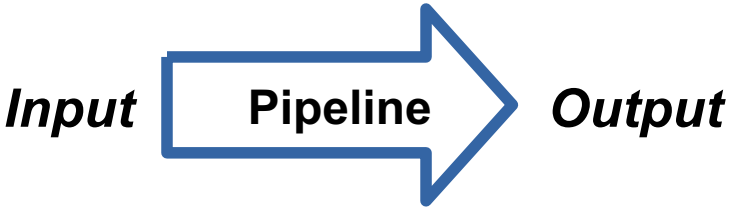
1) What is UNIX and why might you use it?
2) UNIX file structure
3) Introduction to basic UNIX commands with focus on “ls”
4) Aliases
5) Profile
6) Simple text editing on Unix with pico

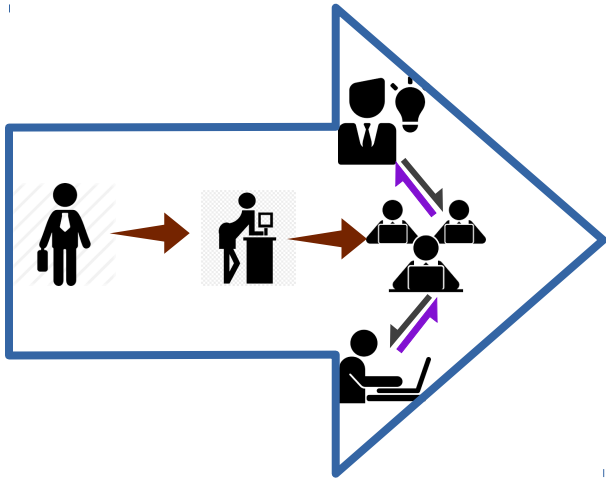
What is Unix

- Unix grew out of project in the 1960s to develop a time-sharing operating system, a software that functions as an interface between user and computer hardware. Time sharing refers to the simultaneous sharing of computer resources by multiple users. Today, Unix is commonly used operating system for scientific computing.
- Graphical user interfaces (GUIs) are available for Unix-like OSs, but their power lies in their command line functionality.
- Unix comprises:
  - Kernel: The “backend” of the OS. Handles time and memory allocation to programs, filesystem management, and responses to system calls
  - Shell: The shell is a special program that functions as a command-line interpreter serving as an interface between kernel and user
  - Programs: Sets of instructions for executing a particular task
- Another way in Unix can be viewed is as a set of files and processes:
  - File: A collection of byte data
  - Process: An executing program. Processes in Unix have an associated process ID (PID)

Why UNIX

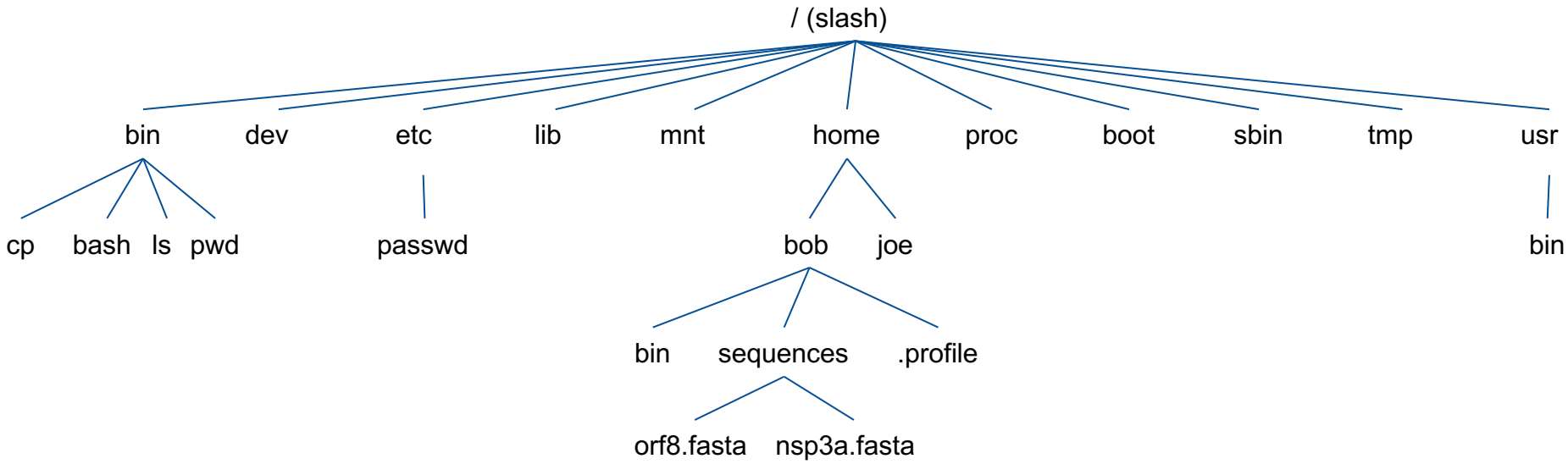
Data analysis often requires chaining together multiple tools that each perform a particular function to accomplish a more complex task (pipelining)





UNIX acts as a 'workbench' of small tools, each performing a small task expertly. Unix allows easy integration of different tools into a pipeline. Programs can run on background and do not need their own pop up window

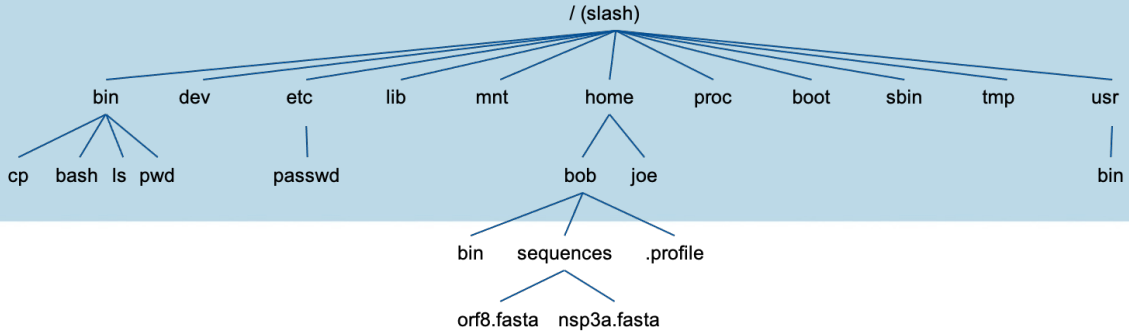
UNIX – File Structure



UNIX employs a hierarchical file system that is grossly consistent across Unix-like operating systems (Solaris, GNU/Linux, OS X). A basic understanding is necessary for working efficiently within a Unix environment.

The Filesystem Hierarchy Standard is maintained by the Linux Foundation. Linux distributions generally follow it, but not strictly.

UNIX – File Structure



/bin : Stands for “binaries” and contains certain fundamental utilities, such as ls or cp, which are generally needed by all users.

/boot : Contains all the files that are required for successful booting process.

/dev : Stands for “devices”. Contains file representations of peripheral devices and pseudo-devices.

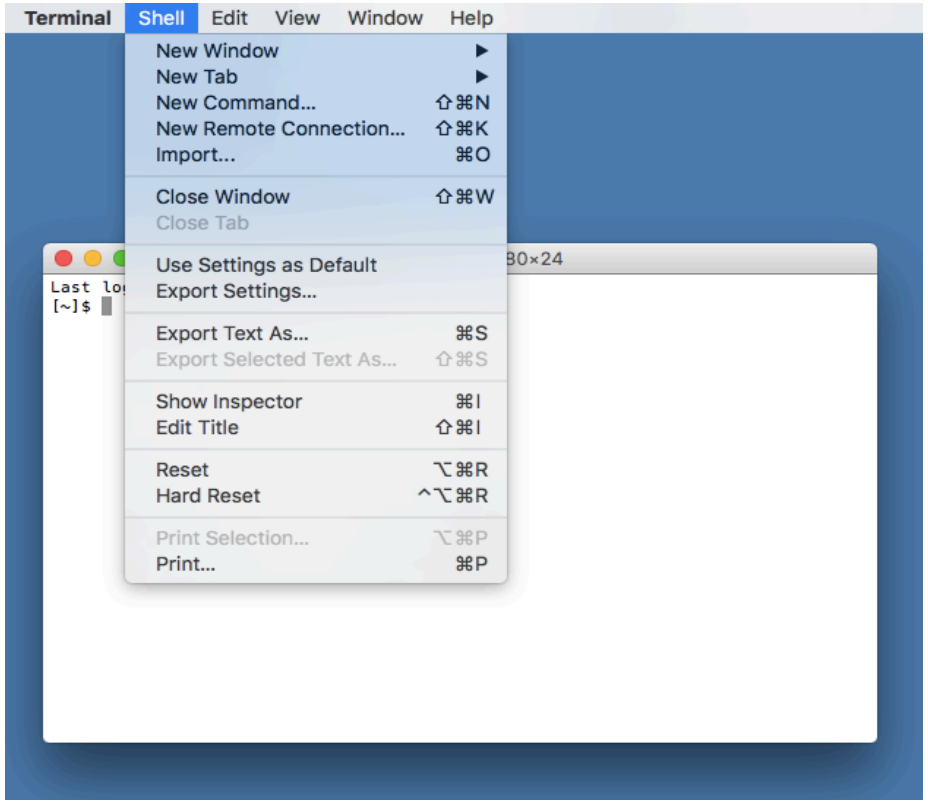
/home : Contains the home directories for the users. A home directory is a default location for all of a particular user's files. It gives the system a standardized location in which to find configuration and preference files.

/lib : Contains system libraries, and some critical files such as kernel modules or device drivers.

/mnt : Stands for “mount”. Contains filesystem mount points. These are used, for example, if the system uses multiple hard disks or hard disk partitions. It is also often used for remote (network) filesystems, CD-ROM/DVD drives, and so on.

/usr/bin : This directory stores all binary programs distributed with the operating system not residing in /bin, /sbin or (rarely) /etc.

UNIX Command Line



There are many kinds of tasks in data analysis that are much more difficult, or perhaps impossible to do by pointing and clicking, as compared to using a command line.

Unix has thousands of commands and makes it easy to integrate graphic interface based programs with text based command line programs.

Accessing the terminal

Linux

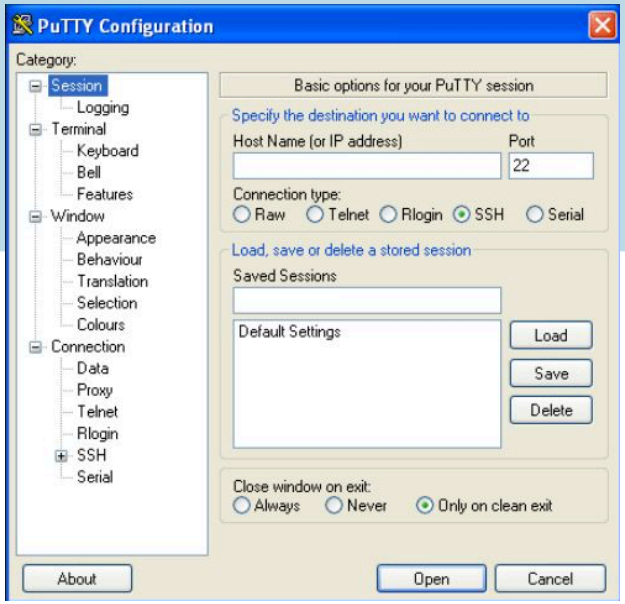
1) Alt + Ctrl + T

Mac

1) Command + Space (to open Spotlight search)
2) Type in “Terminal”.
3) You should see the Terminal application as the “Top Hit” in your results. Double-click it to open a Terminal.

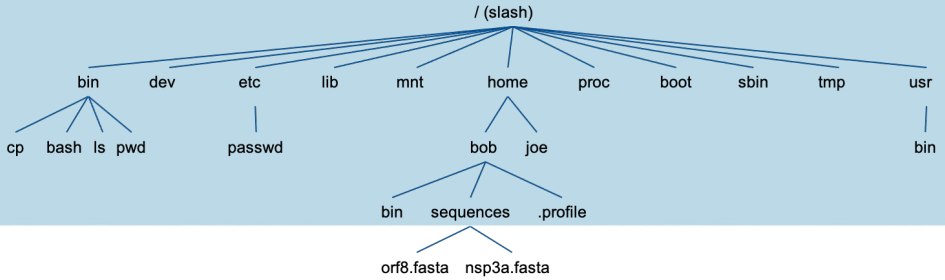
Windows (connecting to a server)

1) Download, install, and launch the Putty terminal emulator (putty.org)
2) Enter the hostname of the server to which you want to connect, set the port to “22”, and select SSH as the connection type. Click “Open”
3) Enter username and password to login



Putty Configuration Window

UNIX – a few essential tools



The following commands (programs) are essential to know in order to work within a unix-like environment. They will be used regularly and will soon become second-nature:

pwd (print working directory) – print the name of the directory in which you are currently working

ls (list) – list the contents of a directory

cd (change directory) – move out of your current working directory to a new location

mkdir (make directory) – create a new directory

touch – update timestamp information on a file or create a new (empty) file

mv (move) – change the location of a file or directory

cp (copy) – copy a file or directory

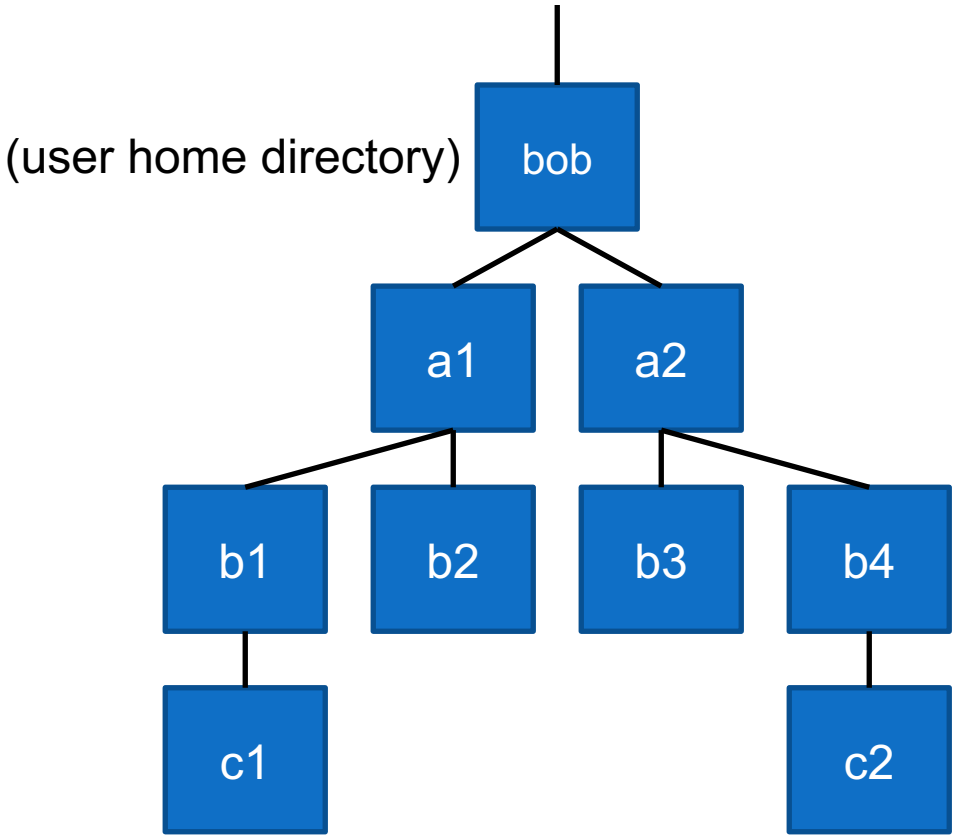
rmdir (remove directory) – delete a directory.

rm (remove) – delete a file or directory. **Caution**: Do not execute this or the next command while sleepy or distracted!

less / ***more*** – view the contents of a file

man (manual) – get help on usage and options associated with a particular unix command

UNIX – moving around in the filesystem



Absolute path: Always starts with a "/". Specifies a directory or file location irrespective of current location. E.g. regardless of my current working directory, I can always move to directory b4 by typing:

```
cd /home/bob/a2/b4
```

Relative path: Specifies a directory or file location relative to your current location. For this purpose a period (`.`) represents your current location, a double period (`..`) represents one directory up, and a tilde (`~`) represents your home directory. If I am located in directory b1 and wish to move to directory c1 I can type any of the following:

```
cd c1          or
cd ./c1        or
cd ../b1/c1    or
cd ~/a1/b1/c1
```

and so on …

National Institute of Allergy and Infectious Diseases

NIAID

UNIX – flags to alter command behavior

From output of `ls`, it is not often clear what is a directory and what is a file. We can provide the `ls` program with options or "flags" that will alter its behavior and provide more information.

`ls -l` give long form information for files and directories.
`ls -p` put a trailing slash at the end of a directory.

`rm -i` makes `rm` a little safer by asking for confirmation before deleting.

To learn what options are available for a particular command we can refer to the documentation for that command which resides on the system.

`man ls` prints to the terminal the documentation for the `ls` program

We will now switch to the hands on exercise unix_handson.sh to gain some practice with various Unix commands

Obtaining unix_hands.sh

- Using a terminal connect to your account on biocompace

```
ssh username@biocompace2.ace.ac.ug
```

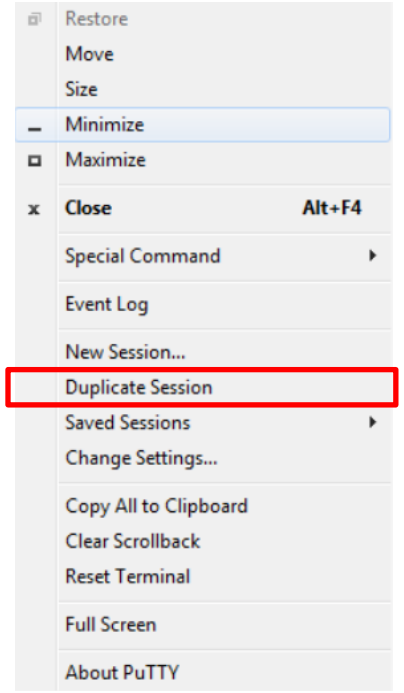
It is not necessary that you copy the training materials at this time; however, if you want to follow along with a local copy of the instructions you can do so.

- Issue the following command to copy the necessary files to your home directory

```
cp -a /home/aarmstrong/ACE_unix/intro_to_unix ~
```

- You can now view the tutorial file with the following:

```
less ~/intro_to_unix/unix_handson.sh
```



Putty Right Click Pop Up

- If you are using Putty and want to have two sessions open simultaneously (one in which to view the tutorial file and one in which to execute the commands), right click along the top of the Putty window, and select Duplicate Session from the pop up menu. Login again with username and password

UNIX – aliases and profiles

In the hands-on exercise we were introduced to aliases. However, aliases created from the command line only exist in the current terminal session. If you log out or use a new terminal window, then you would have to retype the alias to continue using it.

If we instead want to make the aliases persistent, we can create a file storing their definitions which is read and executed with each new session or terminal window instantiation. This usually requires a text editor, an application type we will begin to cover shortly. For now we can create such a file by issuing for example:

```
echo "alias ls='ls -p'" > my_profile
```

To use the alias we have just created, we "source" or execute the commands in the profile file

```
source my_profile
```

You can also write aliases directly to the .bashrc file in your home directory or source a profile containing the aliases from the .bashrc file. Take a moment to examine the contents of your .bashrc file:

```
less ~/.bashrc
```

UNIX – aliases and profile

If you name your file **.profile** in your home directory, then it will be loaded automatically every time you log in.

Files with a filename starting with a dot are hidden files in Unix. To see them, you can use `ls -a` which lists all files.

Pico: a simple Unix text editor

Our next two classes will be dedicated to a powerful text editor called vi whose functionality may be desirable for the manipulation of scripts and data file types regularly encountered in the area of bioinformatics.

The vi learning curve is steep, however, and so for now a simpler text editor called pico (or nano on most Linux systems) can be used for basic file editing.

To open a blank document, simply type:

```
pico
```

With pico open you can immediately begin typing text. Type the text

```
This is only a test
```

Exit the program by typing

```
<ctrl>-x
```

And then type “n” at the prompt to specify that we do not want to save the text we just entered.