Wed, Sep 16

Lecture 1B:

Introduction to Unix

Echinocactus platyacanthus
© Tom Incrocci, 2020

Practical Bioinformatics (Biol 4220)
Instructor: Michael Landis
Email: michael.landis@wustl.edu

Lecture 1B outline

1. What is Unix?
2. Unix features
3. Unix commands
4. Unix filesystem
5. Lab 1B overview

What is Unix?

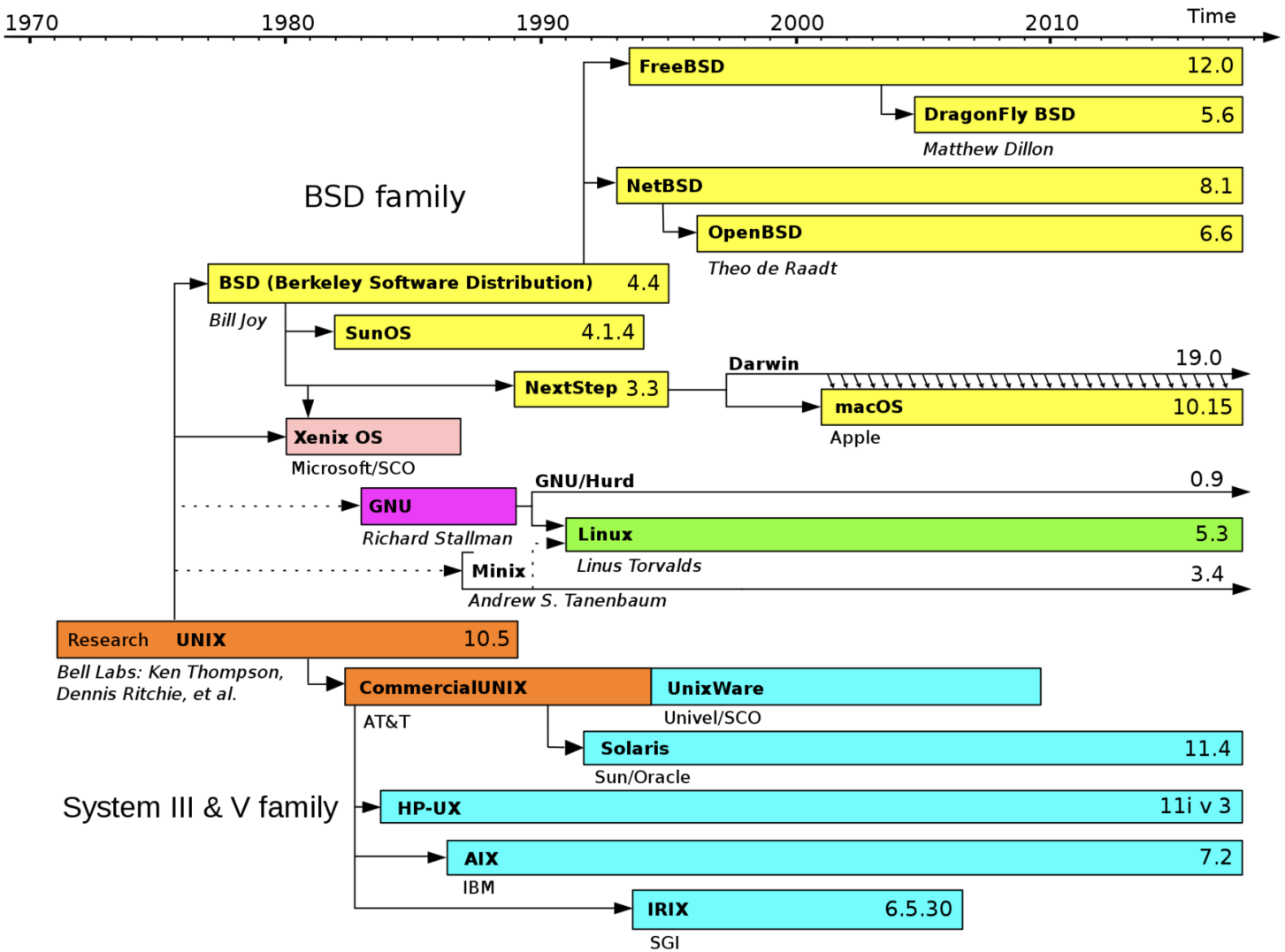
Unix is a family of operating systems

- created by Ken Thompson, Dennis Ritchie, and others at Bell Labs between 1971-1973
- numerous variants by late 1970s
- written in the C programming language

Examples:

- Linux is an open source Unix variant
- Mac OS X is a Unix variant for Apple hardware

Unix family tree



https://commons.wikimedia.org/wiki/File:Unix_timeline.en.svg

Unix features

Modern versions of Unix offer

Multiuser support	mulitiple user can interact with computer at once
Multitasking	OS can run multiple programs simultaneously
Flexible shell	users can issue commands by keyboard or with "script" files
Hierarchical filesystem	child folders and files "nest" inside parent folders; file permissions
Documentation	built-in help files (man pages)
Open source	users are free to read and modify OS code
Portability	OS can be installed on most hardware

Unix philosophy

Designed for complex, modular workflows

1. ***Make each program do one thing well.*** To do a new job, build afresh rather than complicate old programs by adding new "features."
2. ***Expect the output of every program to become the input to another, as yet unknown, program.*** Don't clutter output with extraneous information. Avoid stringently columnar or binary input formats. Don't insist on interactive input.
3. ***Design and build software***, even operating systems, ***to be tried early***, ideally within weeks. Don't hesitate to throw away the clumsy parts and rebuild them.
4. ***Use tools in preference to unskilled help to lighten a programming task***, even if you have to detour to build the tools and expect to throw some of them out after you've finished using them.

McIlroy, Pinson, Tague (*Bell System Technical Journal* 1978)

https://www.youtube.com/embed/JoVQTPbD6UY?enablejsapi=1

(video missing in pdf)

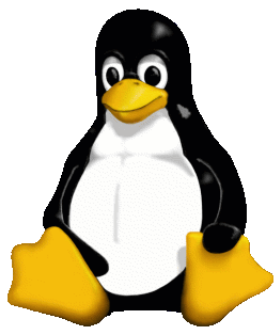
Linux kernel

Linux is a free and open source Unix-based kernel.

- Created by Linus Torvalds in 1991
- Popular both for desktops and servers
- Linux distributions (or ***distros***) are operating systems that package together a Linux kernel, a shell, and a selection of utility programs

Distros serve different needs. *Examples:*

- Debian: older, stable
- ArchLinux: minimalist, customizable
- Gentoo: targets power users
- RedHat: enterprise features, e.g. security, user mgmt



Tux
Linux mascot

Ubuntu

We'll use Ubuntu 20.04 LTS in this course

Features:

- Reliable testing + release cycles
- Excellent tutorials
  https://ubuntu.com/tutorials
- Extremely active support community
  https://ubuntuforums.org/
- ***modified Debian kernel*** for stability
- the popular ***bash shell*** by default

Unix design + open source = innovation

Unix is a family of operating systems

Linux is a Unix-inspired kernel

Debian is a Linux-based distribution

Ubuntu is a Debian-based distribution

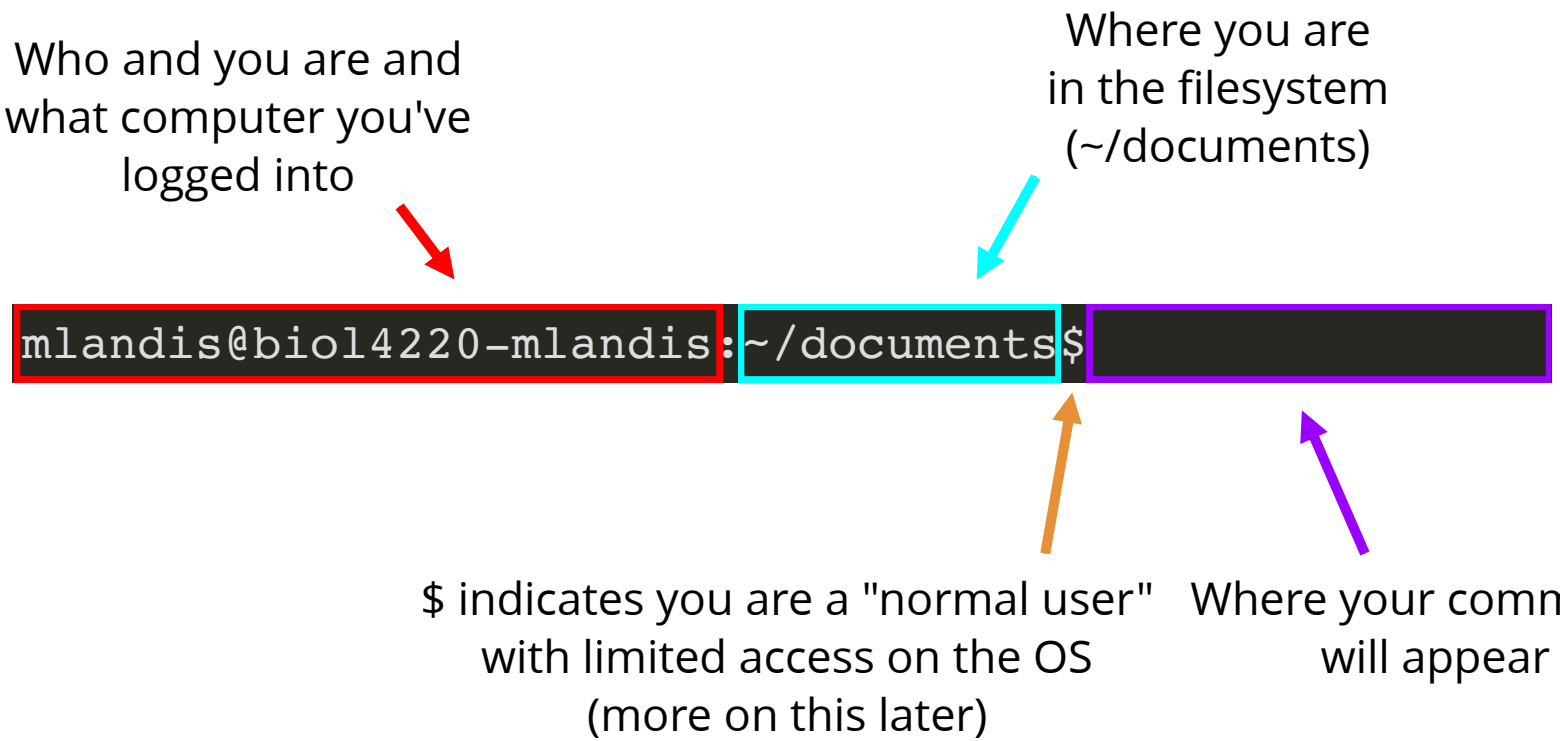
But how is Unix relevant to biological research?

(video missing in pdf)

Typical Unix command line

```
mlandis@biol4220-mlandis:~/documents$
```

Typical Unix command line

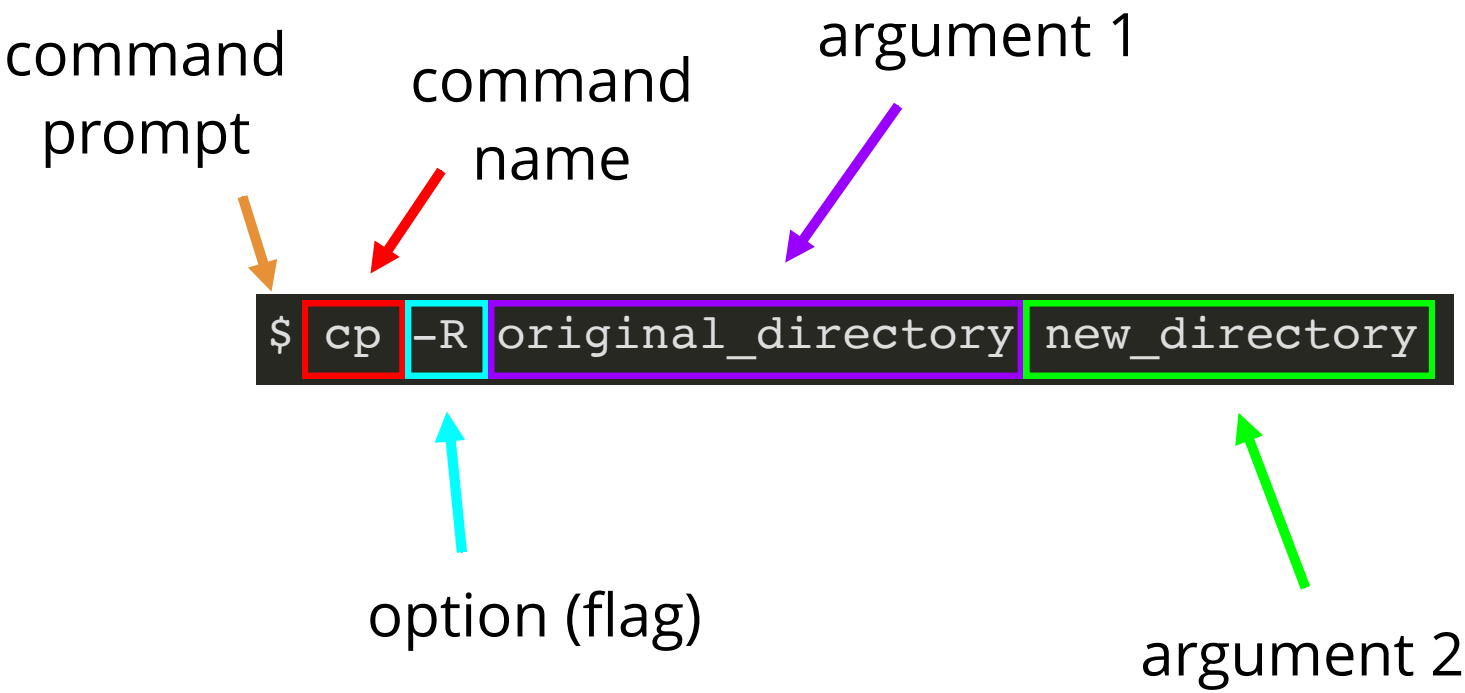


To simplify instructions, the command line prompt is often represented by $ alone

Anatomy of Unix command

```
$ cp -R original_directory new_directory
```

Anatomy of Unix command



Commands are applied against ***argument(s)*** and its behavior can be modified by ***option(s)***

Some commands require a specific number of arguments (0, 1, 2, ...) others are flexible

```
$ pwd
$ cd documents
$ cat file1.txt file2.txt file3.txt
```

Options are not necessary, and some can be grouped

```
$ ls
$ ls -l -a -r -t
$ ls -lart
```

Some options require arguments

```
$ ssh -p 22 128.252.89.47
```

A ***comment*** is a "non-command" that helps communicate intent to humans

```
$ # list files (and hidden files)
$ ls -a
```

Commands are ***executed*** one at a time, and in the order they are received

```
$ # first we'll create a folder
$ mkdir data
$ # then we'll enter the folder
$ cd data
```

A single line may contain multiple commands, when the commands are separated by ;

```
$ # first create then copy output.txt
$ touch output.txt; cp output.txt copy.txt
```

echo, print text

```
# `echo` prints its arguments to
# the standard output device (stdout)
$ echo Hello, world!
Hello, world!
$ echo GATTACA
GATTACA
```

ls, list files

```
# `ls` lists all contents in the current
# directory, printed to stdout
$ ls
labs lectures notes.txt
$
# `ls` accepts target directories as
# arguments, too
$ ls labs lectures
labs:
lab_01A  lab_01B
lectures:
lab_01A.pdf  lab_01B.pdf
```

cat, concatenate

```
# `cat` concatenates the contents
# of one or more files together
# then prints to stdout
$ cat hello.txt
Hello,
$ cat world.txt
 world!
$ cat hello.txt world.txt
Hello, world!
```

mv, move

```
# `mv` moves files and folders
# within the filesystem; one
# use is to rename files
$ ls
notes.txt old_notes.txt
$ mv old_notes.txt notes.txt
$ ls
notes.txt
$ mv notes.txt old_notes.txt
$ ls
old_notes.txt
```

```
# `mv` can rename folders, too
$ ls
old_lectures
$ mv old_lectures lectures
$ ls
lectures
```

cp, *copy*

```
# `cp` copies the first file
# to the second filepath
$ ls
notes.txt
$ cp notes.txt old_notes.txt
$ ls
notes.txt old_notes.txt
```

```
# `cp` cannot copy folders
$ cp lectures old_lectures
cp: lectures is a directory (not copied).
```

rm, remove

```
# `rm` removes a file from the
# filesystem; removed files
# cannot easily be restored!!
~$ ls
notes.txt
~$ rm notes.txt
~$ ls
~$
```

Unix filesystem

The ***filesystem*** organizes files and folders into a hierarchical structure

- ***files*** contain data*, e.g.*
  - text, programs, music
- ***folders*** contain files or other folders

The filesystem identifies read/write/execute ***user permissions*** for filesystem objects. More details in later lectures.

An example filesystem

```
$ tree
.
└── home
    ├── course_project.md
    ├── course_schedule.md
    ├── how_to_guide.md
    ├── labs
    │   ├── lab_01A.md
    │   └── lab_01B.md
    └── lectures
        ├── lect_01A.pdf
        └── lect_01B.pdf


3 directories, 7 files
```

labs is nested within home
as home/labs

Filesystem paths

A ***path*** is the address of a file or folder in the filesystem

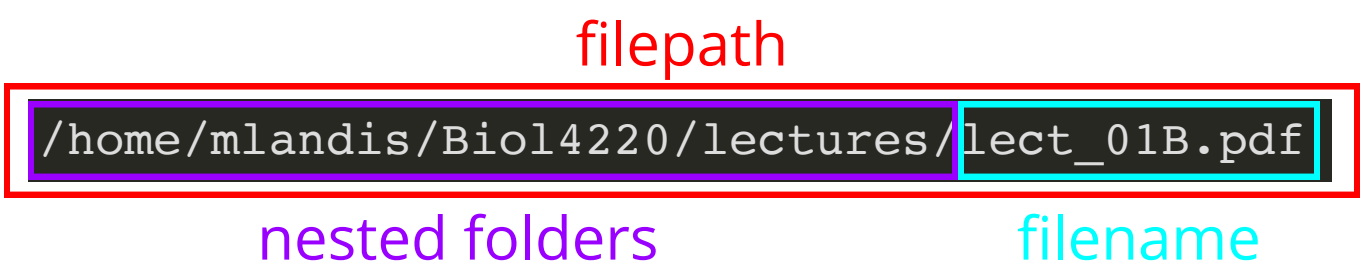
- the path is a string of text that lists the nested directories, from *deep* to *shallow*, locating the object
- directories in the path are separated by the `/` token
- file paths end with the ***filename***, e.g. *output.txt*
- paths for folders do not end in a filename

```
/home/mlandis/Biol4220/lectures/lect_01B.pdf
```

Filesystem paths

A ***path*** is the address of a file or folder in the filesystem

- the path is a string of text that lists the nested directories, from *deep* to *shallow*, locating the object
- directories in the path are separated by the `/` token
- file paths end with the ***filename***, e.g. *output.txt*
- paths for folders do not end in a filename



An ***absolute path*** specifies all nested folders, beginning with the deepest folder, ***the root directory***

A ***relative path*** specifies only those folders needed to locate a resource *relative* to your current location in the filesystem

If your current location is /home/mlandis/Biol4220, these paths refer to the same file:

```
/home/mlandis/Biol4220/lectures/lect_01B.pdf
```

and

```
lectures/lect_01B.pdf
```

Special directories

The ***home directory*** (~) is a shortcut for /home/username; most of your work will take place in ~

The ***current directory*** (.) is a shortcut for the currently occupied directory; can be useful to make relative paths explicit

The ***parent directory*** (..) is a shortcut for the directory that *contains* the current directory; useful for moving "up" one directory

These paths are equivalent (assuming you are located in /home)

```
# absolute
/home/mlandis/Biol4220/labs
# relative
mlandis/Biol4220/labs
# using home directory (~)
~/Biol4220/labs
# using current directory (.)
./mlandis/Biol4220/labs
# using parent directory (..)
/home/mlandis/Biol4220/lectures/../labs
```

mkdir, make directory

```
# `mkdir` makes a new directory
# specified by the path argument
~/labs$ ls
# relative path
~/labs$ mkdir lab_01A
~/labs$ ls
lab_01A
# using .. in path
~/labs$ mkdir ../lectures
~/labs$ ls ..
labs  lectures
# absolute path
~/labs$ mkdir /home/mlandis/labs/lab_01B
~/labs$ ls
lab_01A  lab_01B
```

cd, change directory

```
# `cd` changes into the directory
# specified by the path argument
~$ ls
labs
~$ ls labs
lab_01A  lab_01B
# relative path
~$ cd labs/lab_01B
# using .. in path
~/labs/lab_01B$ cd ../lab_01A
# absolute path
~/labs/lab_01A$ cd /home/mlandis
~$
```

rmdir, remove directory

```
# `rmdir` removes and empty directory
# specified by the path argument
~$ ls
labs  lectures
# `labs` isn't empty!
~$ rmdir labs
rmdir: labs: Directory not empty
~$ ls labs
lab_01A  lab_01B
~$ ls labs/lab_01A labs/lab_01B
lab_01A:

lab_01B:
# remove subdirectories
~$ rmdir labs/lab_01A
~$ rmdir labs/lab_01B
# now remove `labs`
~$ rmdir labs
~$ ls
~$
```

cp -R, copy folders

```
# `cp -R` will copy a directory and
# recursively copy all internal
# files and directories
~$ ls lectures
lect_01A.pdf  lect_01B.pdf
# `cp` cannot target directories by default
~$ cp lectures lectures_old
cp: lectures is a directory (not copied).
# add the `-R` flag
~$ cp -R lectures lectures_old
~$ ls
lectures  lecture_old
```

rm -rf, remove completely

```
# WARNING: this is a very dangerous command!
#
# `rm -rf` will remove a file or directory,
# along with all of its contents, without
# any warnings or user interactions
#
$ ls
labs
$ ls labs
lab_01A  lab_01B
$ rmdir labs
rmdir: labs: Directory not empty
$ rm labs
rm: labs: is a directory
$ rm -rf labs
$ ls
$
```

Suppose this lists all filesystem objects

- what directories are shown?
- what files are shown?
- which folder contains three files?
- which folder contains two directories?
- including the root directory, how many directories are in the absolute path for lect_01B.pdf?

```
/home/mlandis/Biol4220
/home/mlandis/Biol4220/notes.txt
/home/mlandis/Biol4220/labs
/home/mlandis/Biol4220/labs/lab_01A.pdf
/home/mlandis/Biol4220/labs/lab_01B.pdf
/home/mlandis/Biol4220/lectures
/home/mlandis/Biol4220/lectures/lect_01A.pdf
/home/mlandis/Biol4220/lectures/lect_01B.pdf
/home/mlandis/Biol4220/lectures/lect_02A_draft.pdf
```

How would you execute this series of commands, using only commands we learned in Lecture 01B?

1. change to the home directory
2. copy the notes.txt file into lectures
3. delete the lectures directory
4. copy the labs directory into your home directory

```
/home/mlandis/Biol4220
/home/mlandis/Biol4220/notes.txt
/home/mlandis/Biol4220/labs
/home/mlandis/Biol4220/labs/lab_01A.pdf
/home/mlandis/Biol4220/labs/lab_01B.pdf
/home/mlandis/Biol4220/lectures
/home/mlandis/Biol4220/lectures/lect_01A.pdf
/home/mlandis/Biol4220/lectures/lect_01B.pdf
/home/mlandis/Biol4220/lectures/lect_02A_draft.pdf
```