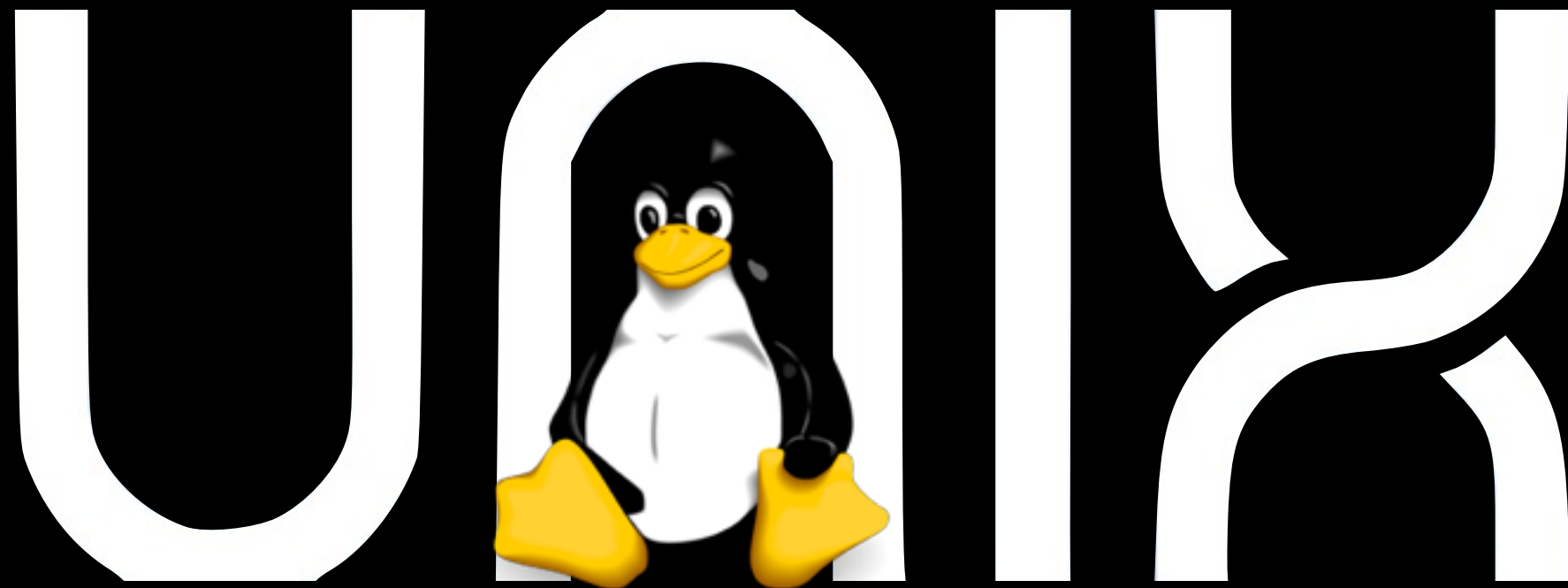


# Introduction To



**Ryan Mills**  
**remills@umich.edu**

# Introduction to Biocomputing

<http://bioboot.github.io/web-2017/>

Monday	<b>Introduction to UNIX*</b>
Tuesday	<b>Introduction to Programming</b>
Wednesday	<b>Data Analysis and Graphics with R</b>
Thursday	<b>Version Control &amp; Cluster Computing*</b>
Friday	<b>Group Projects</b>

HELLO  
my name is

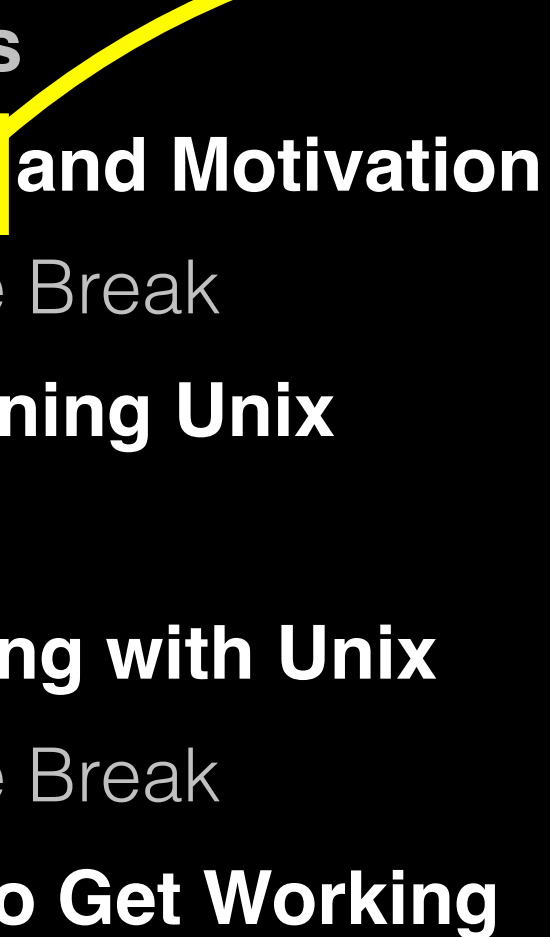
Fndb

HELLO  
*HIS* my name is

<i>]

# Today's Menu

	Time	Topics
I	9:00-10:15 AM	<b>Setup</b> and Motivation
	10:15-10:30 AM	Coffee Break
II	10:30-12:00 AM	<b>Beginning Unix</b>
	12:00-1:00 PM	Lunch
III	1:00-2:15 PM	<b>Working with Unix</b>
	2:15-2:30 PM	Coffee Break
IV	2:30-4:00 PM	<b>How to Get Working</b>

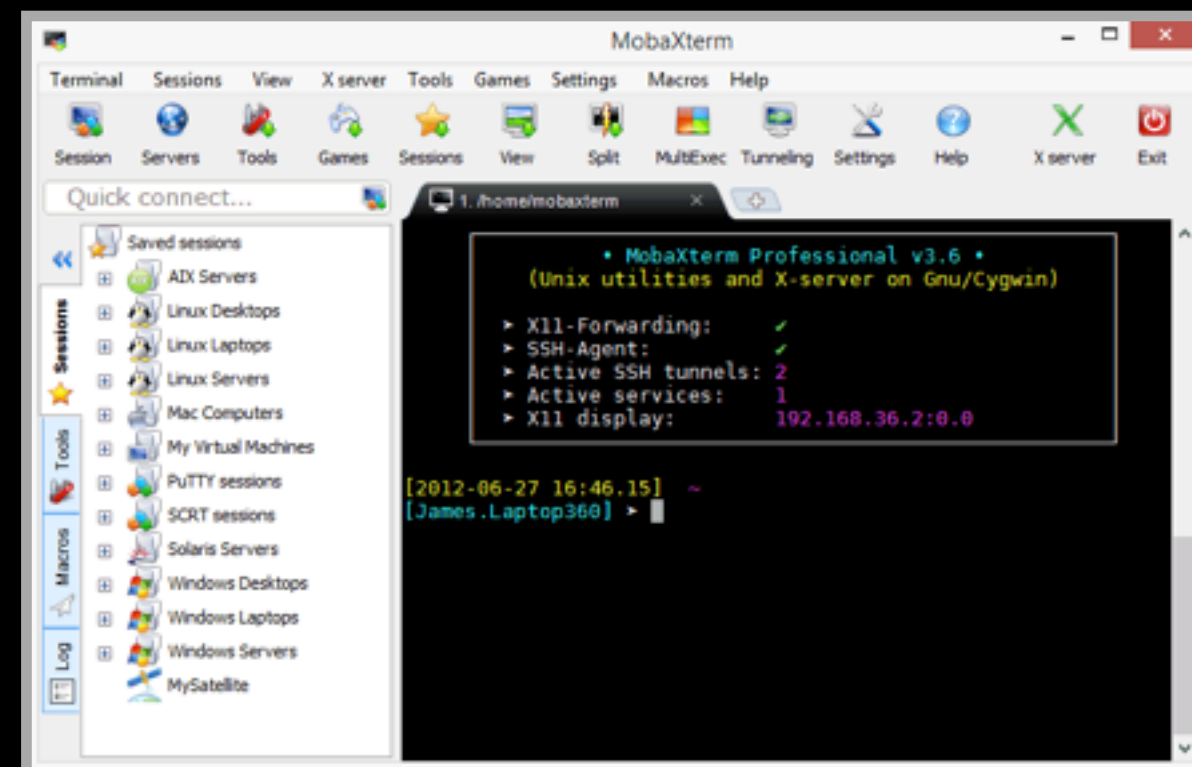
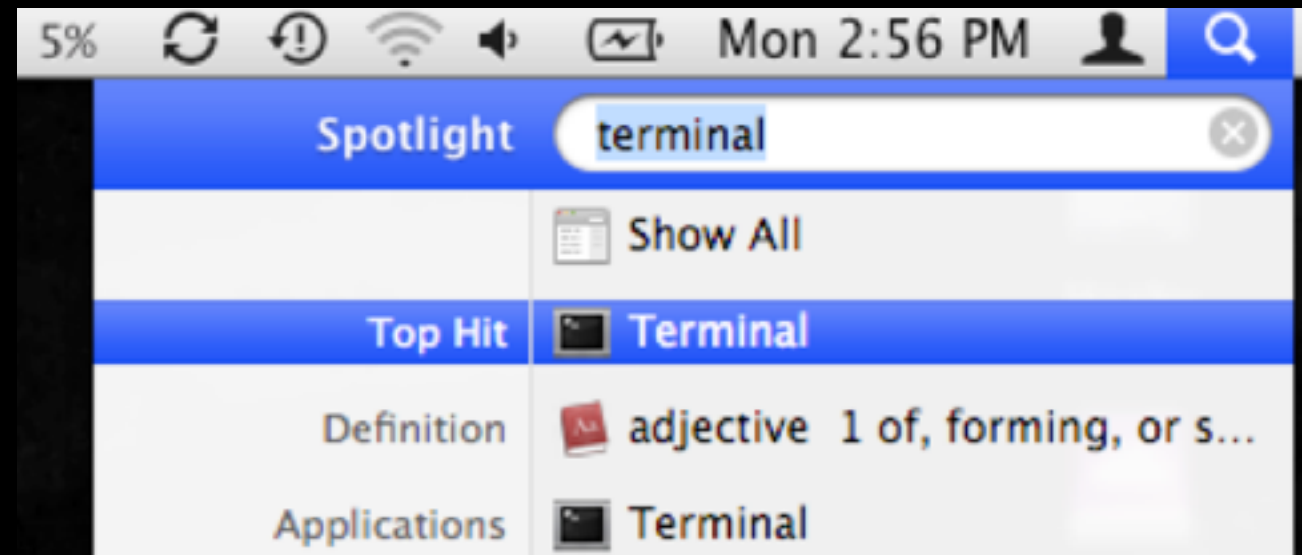


<https://dcmb-courses.github.io/bioinf606-2018/setup/>

# Lets get started....

Do it Yourself!

**Mac**  
Terminal



**PC**  
MobaXterm

# Setup Checklist

<https://dcmb-courses.github.io/bioinf606-2018/setup/>

- ☑ **Mac:** Terminal *or* **PC:** MoblXterm
- ☑ **Mac:** Git install *or* **PC:** MoblXterm git & CygUtils plugins
- ☑ Python Anaconda install
- ☑ R and RStudio install
- ☑ Flux access form submitted and Duo mobile app obtained
- ☑ Example data downloaded: <http://tinyurl.com/day1-unix>

*# In your terminal type*  
> which git  
> git --version

# Motivation

Why do we use Unix?

<b>Modularity</b>	Core programs are modular and work well with others
<b>Programmability</b>	Best software development environment
<b>Infrastructure</b>	Access to existing tools and cutting-edge methods
<b>Reliability</b>	Unparalleled uptime and stability
<b>Unix Philosophy</b>	Encourages open standards



<b>Modularity</b>	Core programs are modular and work well with others
<b>Programmability</b>	Best software development environment
<b>Infrastructure</b>	Access to existing tools and cutting-edge methods
<b>Reliability</b>	Unparalleled uptime and stability
<b>Unix Philosophy</b>	Encourages open standards

# Modularity

The Unix shell was designed to allow users to easily build complex workflows by interfacing smaller **modular programs** together.



An alternative approach is to write a **single complex program** that takes raw data as input, and after hours of data processing, outputs publication figures and a final table of results.



Which would you prefer and why?



Modular

vs



Custom

# Advantages/Disadvantages

The 'monster approach' is customized to a particular project but results in massive, fragile and difficult to modify (therefore inflexible, untransferable, and error prone) code.

With **modular workflows**, it's easier to:

- Spot errors and figure out where they're occurring by inspecting intermediate results.
- Experiment with alternative methods by swapping out components.
- Tackle novel problems by remixing existing modular tools.

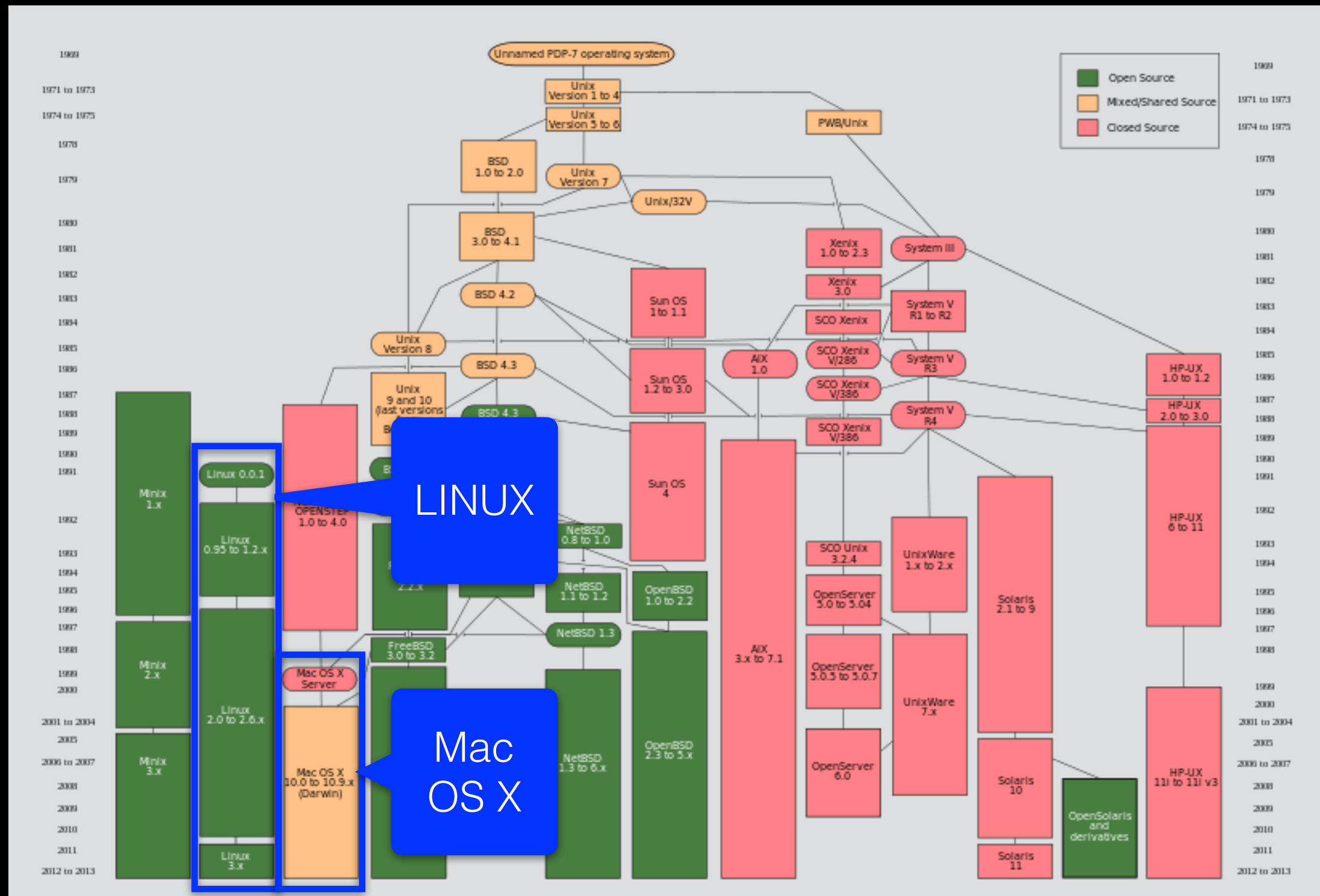
# Unix ‘Philosophy’

“Write programs that do one thing and do it well. Write programs to work together and that encourage open standards. Write programs to handle text streams, because that is a universal interface.”

— Doug McIlory



# Unix family tree [1969-2010]



Source: [https://commons.wikimedia.org/wiki/File:Unix\\_history-simple.svg](https://commons.wikimedia.org/wiki/File:Unix_history-simple.svg)

Basics	File Control	Viewing & Editing Files	Misc. useful	Power commands	Process related
ls	mv	less	chmod	grep	top
cd	cp	head	echo	find	ps
pwd	mkdir	tail	wc	sed	kill
man	rm	nano	curl	uniq	Ctrl-c
ssh	 (pipe)	touch	source	git	Ctrl-z
	> (write to file)		cat	R	bg
	< (read from file)		tmux	python	fg

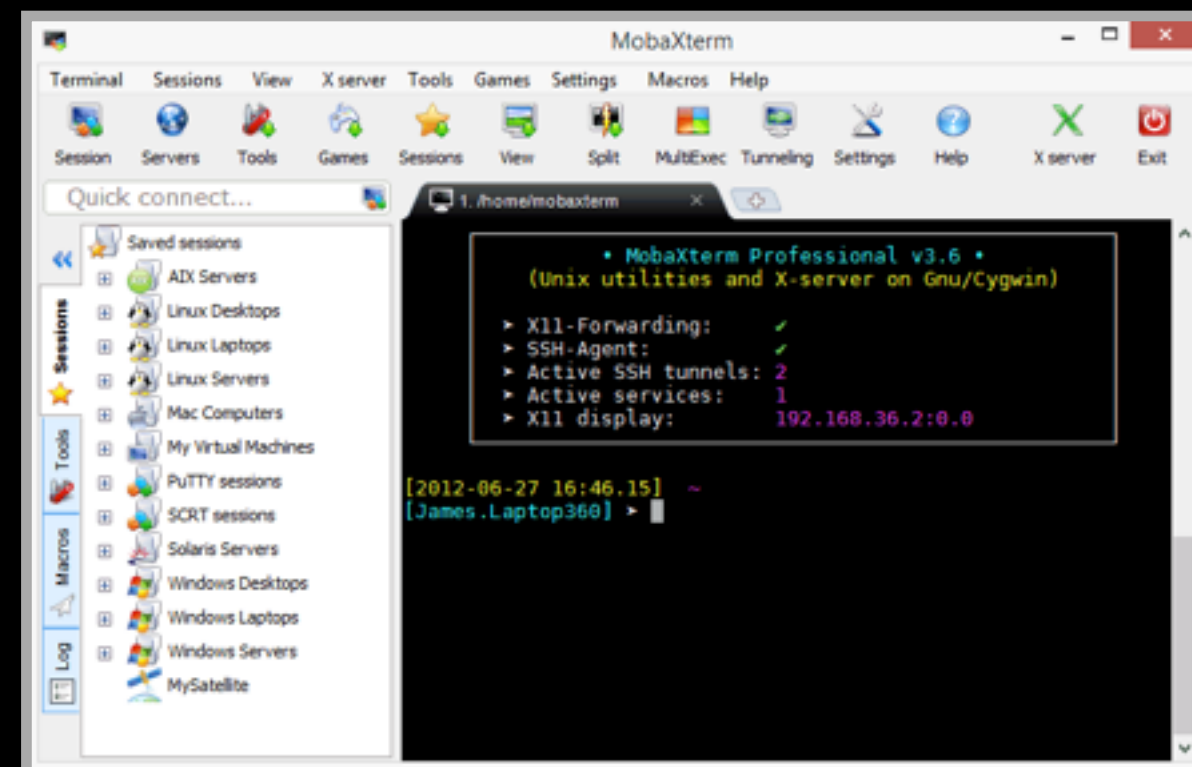
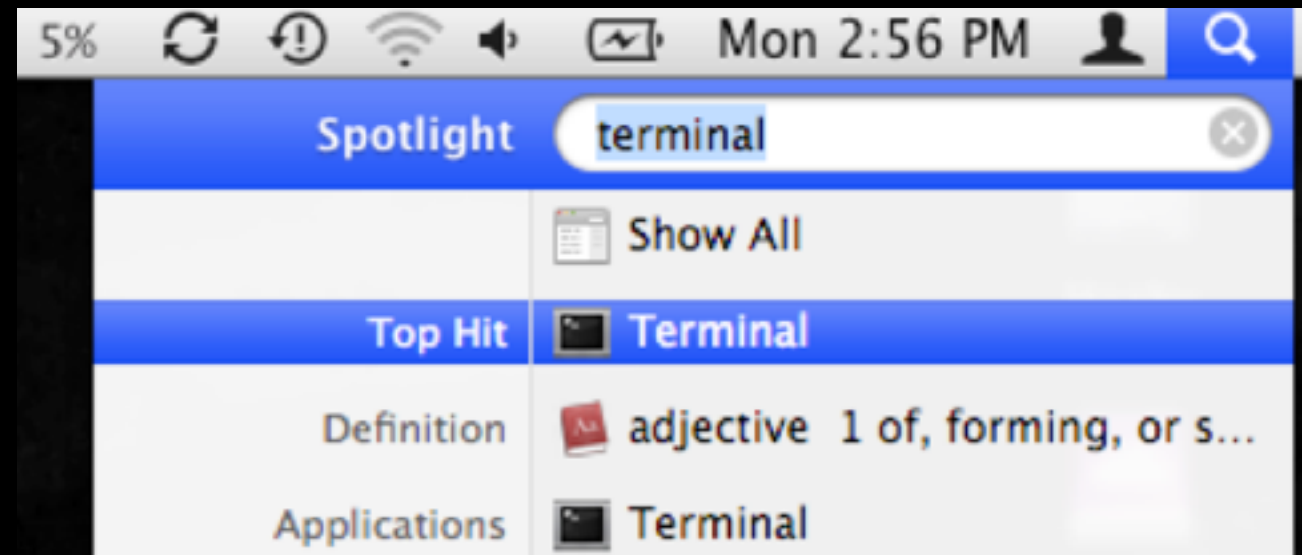
Basics	File Control	Viewing & Editing Files	Misc. useful	Power commands	Process related
<b>ls</b>	<b>mv</b>	<b>less</b>	<b>chmod</b>	<b>grep</b>	<b>top</b>
<b>cd</b>	<b>cp</b>	<b>head</b>	echo	find	<b>ps</b>
<b>pwd</b>	<b>mkdir</b>	<b>tail</b>	wc	sed	<b>kill</b>
<b>man</b>	<b>rm</b>	<b>nano</b>	curl	uniq	Ctrl-c
<b>ssh</b>	 (pipe)	touch	source	<b>git</b>	Ctrl-z
	> (write to file)		cat	<b>R</b>	bg
	< (read from file)		tmux	<b>python</b>	fg



# Lets get started....

Do it Yourself!

**Mac**  
Terminal



**PC**  
MobaXterm

# **Test:** Connecting to remote machines (with **ssh**)

- Most high-performance computing (HPC) resources can only be accessed by **ssh** (Secure SHell)
  - > ssh [user@host.address]
  - > ssh remills@scs.gpcc.itd.umich.edu
  - > ssh -X remills@flux-login.arc-ts.umich.edu

# Test: Your software versions

- We will use the **which** command to locate your versions of the major software we will be using this week.

```
> which R  
> R --version
```

Now do the same for **python** and **git** , *i.e.*

```
> which git  
> git --version
```

- If you get an 'error' or 'not found' msg let us know!

HELLO  
my name is

Fndb

HELLO  
*HIS* my name is

<i>]