STATS 700-002
Data Analysis using Python

Lecture 6: the UNIX/Linux Command Line
UNIX/Linux: a (very) brief history

1960s: Multics (Bell Labs, MIT, GE), a time-sharing operating system

1970s: UNIX developed at Bell Labs


1990s: GNU/Linux emerges

2000s: MacOS developed based on UNIX

The Unix philosophy: do one thing well

1. Write programs that do one thing and do it well.
2. Write programs to work together.
3. Write programs to handle text streams, because that is a universal interface.


These three design principles, articulated in the concise form above long after Unix was written, go a long way toward explaining how to approach the command line. For nearly any task you wish to accomplish, there almost certainly exists a way to do it (reasonably) easily by stringing together several different programs.
Connecting to other machines: ssh

ssh (Secure Shell) network protocol allows secure communication machines
    Allows remote access to resources on, e.g., a server or compute cluster

UNIX/Linux/MacOS: open a terminal, type “ssh user@machine”, and you’re off!

Windows: ssh does not come standard.
    PuTTY: https://en.wikipedia.org/wiki/PuTTY
    Cygwin: https://en.wikipedia.org/wiki/Cygwin
Typical 

Secure shell (\textit{ssh}) login to Fladoop, from the command line on my Mac (term)

I cropped a few security-related things out of here.

And now I have a command line prompt on the Fladoop cluster!
Typical ssh session

If you’re using a Mac or UNIX/Linux machine, you can pretty much copy what I just did. On Mac, use the app Terminal. On UNIX/Linux systems, you should be able to pull up a terminal using a shortcut like `ctrl+alt+t`, depending on what distribution of UNIX/Linux you’re using.

On Windows, you can use cygwin to run a command line on your own machine, or use PuTTY to open an ssh connection to another machine like I did in this slide.

If you have trouble with any of this, please post to the discussion board and come to office hours and get assistance promptly so that you can do the command line portion of the homework!

And now I have a command line prompt on the Fladoop cluster!
Parts of the command line prompt

Username
Hostname
Current directory
Prompt/delimiter

Note: details of this will vary from one computer to the next (and it can be customized by the user), but this is the default on the Fladoop cluster. For information on customizing the command line prompt, see https://linuxconfig.org/bash-prompt-basics
Basic concepts

**Shell**: the program through which you interact with the computer. Provides the command line and facilitates typing commands and reading outputs. Popular shells: bash (Bourne Again Shell), csh (C Shell), ksh (Korn Shell)

Redirect: take the output of one program and make it the input of another. We’ll see some simple examples in a few slides.

 stdin, stdout, stderr: three special “file handles” for reading inputs from the shell (stdin) and writing output to the shell (stderr for error messages, stdout other information).
Basic commands for navigating

**pwd**: “print/present working directory”. Print the directory that you are currently in.

**ls**: list the contents of the current directory.

**cd dirname**: change the working directory to dirname.

**Try it out!** Type `pwd` or `ls` in your shell (either in terminal/cygwin or on Fladoop).

**cd dirame**: change the working directory to dirame.

Some special directory symbols:

~ : your home directory. `cd ~` will take you back to your home.

. : the current directory. `cd .` will take you to where you are right now.

.. : the directory above the current directory.

If you’re in `/home/klevin/stats`, then `cd ..` will take you to `/home/klevin`. 
Example: `pwd`, `ls` and `cd`

```
keith@Steinhaus:~$ ssh -X klevin@flux-hadoop-login.arc-ts.umich.edu
Password:

[...]

[klevin@flux-hadoop-login2 ~]$ pwd
/home/klevin
[klevin@flux-hadoop-login2 ~]$ ls
Myfile.txt  stats700f17
[klevin@flux-hadoop-login2 ~]$ cd stats700f17/
[klevin@flux-hadoop-login2 stats700f17]$ pwd
/home/klevin/stats700f17
[klevin@flux-hadoop-login2 stats700f17]$ ls .
hw1.tex  hw2.tex  hw3.tex
[klevin@flux-hadoop-login2 stats700f17]$ ls ..
myfile.txt  stats700f17
[klevin@flux-hadoop-login2 stats700f17]$ ls ~
myfile.txt  stats700f17
```
Getting help: man pages

When in doubt, the shell has built-in documentation, and it tends to be good!

```
man cmdname : brings up documentation about the command cmdname
```

This help page is called a man (short for manual) page. These have a reputation for being terse, but once you get used to reading them, they are extremely useful!

Some shells also have a command `apropos`:

```
apropos topic : lists all commands that might be relevant to topic.
```

Let’s read some of the `ls` man page and see if we can make sense of it.
Relevant xkcds
Basic commands: actually doing things

In the next few slides, we’ll look at some commands that actually let you do things like creating files and directories, reading files, and moving them around.

Follow along with the examples in your terminal, if you like (highly recommended).
Basic commands: `echo`

`echo string`: prints string to the shell.

```bash
keith@Steinhaus:~$ echo "hello world."
hello world.
keith@Steinhaus:~$ echo "hello world!"
-bash: !": event not found
keith@Steinhaus:~$ echo "hello world\!"
hello world!
keith@Steinhaus:~$ echo 'hello world!'
hello world!
keith@Steinhaus:~$ echo "hello\tworld."
hello	world.
keith@Steinhaus:~$ echo -e "hello\tworld."
hello world.
```

The shell tries to interpret the exclamation point as referencing a previous command rather than as text. Escaping doesn’t do the trick here. Instead, use single-quotes to tell the shell not to try and process the string.

To print special characters (tabs, newlines, etc), use the flag `-e`, without which `echo` just prints what it’s given.
Aside: redirections using `>`

What if I want to send output someplace other than the shell?

```
keith@Steinhaus:~$ echo -e "hello\tworld." > myfile.txt
keith@Steinhaus:~$
```

Note: the other redirect, `<`, has a somewhat similar function, but is beyond our purposes here (stay tuned for command-line workshop at end of semester, perhaps?)

Redirect tells the shell to send the output of the program on the “greater than” side to the file on the “lesser than” side. This creates the file on the RHS, an overwrites the old file, if it already exists!
Basic commands: `cat`

cat filename: prints the contents of the file filename.

```
keith@Steinhaus:~$ cat myfile.txt
hello    world
keith@Steinhaus:~$
```

So cat is like echo but it takes a filename as argument instead of a string.
Basic commands: head

head filename: prints the first 10 lines of filename.
head -n X filename: prints the first X lines of filename.

```
keith@Steinhaus:~$ head ~/Teaching/Homeworks/HW1/homework1.tex
\documentclass[11pt]{article}
\usepackage{enumerate}
\usepackage{amsmath}
\usepackage{amsfonts}
\usepackage{hyperref}
\oddsidemargin 0mm
\evensidemargin 5mm
\topmargin -20mm
keith@Steinhaus:~$
```
Basic commands: more/less

more and less are two (very similar) programs for reading ASCII files.

```
[klevin@flux-hadoop-login2 stats700f17]$ less hw1.tex
[less takes up the whole screen]

This is just a dummy file that I wrote as an example.
An actual tex file wouldn't look like this.
It would have a bunch of stuff like
\begin{definition}
An integer $p > 1$ is called \textit{prime}
is its only divisors are $1$ and $p$.
\end{definition}
and it would have a preamble section declaring its document type and a bunch of other stuff.
```

Note: press “q” to quit less/more and return to the command line.
Basic commands: mkdir

mkdir dirname: creates a new directory called dirname, if it doesn’t exist

[kleinv@flux-hadoop-login2 stats700f17]$ ls
hw1.tex  hw2.tex  hw3.tex
[kleinv@flux-hadoop-login2 stats700f17]$ mkdir hadoop_stuff
[kleinv@flux-hadoop-login2 stats700f17]$ ls
hadoop_stuff  hw1.tex  hw2.tex  hw3.tex
[kleinv@flux-hadoop-login2 stats700f17]$
Basic commands: \texttt{mv}

\texttt{mv file1 file2}: \textbf{moves} \texttt{file1} to \texttt{file2}, \textbf{overwriting} \texttt{file2}.

If \texttt{file2} is a directory, this places \texttt{file1} inside that directory, again replacing any existing file with the same \texttt{basename} as \texttt{file1}. \\
/path/to/file/basename.txt
Basic commands: \texttt{cp}

\texttt{cp} \texttt{file1} \texttt{file2}: similar to \texttt{mv}, but creates a copy of \texttt{file1} with name \texttt{file2}

So \texttt{cp} is like \texttt{mv} but \texttt{file1} is copied instead of being renamed

\begin{verbatim}
[klevin@flux-hadoop-login2 stats700f17]$ cat homework2.tex
This is the second homework!
[klevin@flux-hadoop-login2 stats700f17]$ cp homework2.tex HW2.tex
[klevin@flux-hadoop-login2 stats700f17]$ cat homework2.tex
This is the second homework!
[klevin@flux-hadoop-login2 stats700f17]$ cat HW2.tex
This is the second homework!
[klevin@flux-hadoop-login2 stats700f17]$ ls
hadoop_stuff homework2.tex HW2.tex hw3.tex
\end{verbatim}

\textbf{Note:} to copy a directory, you must include the \texttt{-r} flag to \texttt{cp}: \texttt{cp -r dirname otherdirname}
Basic commands: \texttt{rm}

\texttt{rm filename}: deletes the file \texttt{filename}. \textbf{Be very very careful with this!}

```
[klevin@flux-hadoop-login2 stats700f17]$ ls
hadoop_stuff homework2.tex HW2.tex hw3.tex
[klevin@flux-hadoop-login2 stats700f17]$ rm HW2.tex
[klevin@flux-hadoop-login2 stats700f17]$ ls
hadoop_stuff homework2.tex hw3.tex
[klevin@flux-hadoop-login2 stats700f17]$
```
Basic commands: `logout`

`logout`: close connection to the current machine

```
[klevin@flux-hadoop-login2 stats700f17]$ logout
Connection to flux-hadoop-login.arc-ts.umich.edu closed.
keith@Steinhaus:~$
```

**Note:** depending on the type of shell session in use, you may need to use `exit` or `ctrl-D` to log off.
Moving files between machines: scp (Secure copy)

scp localfile username@hostname:path/to/file

Copy a file from your machine to some other machine via ssh

scp username@hostname:path/to/file localfile

Copy a file from another machine to your machine via ssh

```
keith@Steinhaus:~$ scp myfile.txt
klevin@flux-hadoop-login.arc-ts.umich.edu:~/stats700f17/myfile.txt
Password:
[authentication]
myfile.txt                   100%   14   0.0KB/s   00:00
keith@Steinhaus:~$ ssh -X klevin@flux-hadoop-login.arc-ts.umich.edu
Password:
[authentication]
[klevin@flux-hadoop-login1 ~]$ ls stats700f17/
.hadoop_stuff  homework2.tex  hw3.tex  myfile.txt
```
We’ve only scratched the surface!

The UNIX command line is extremely powerful!

Offers numerous tools for working with text and general data wrangling: grep, sed, awk, tr, cut, ...

Ability to use the command line is crucial to being a good “data scientist”
  ● Command line, once you’re good at it, makes things VERY fast!
  ● 2-3 lines of shell script to do what would take an entire Python program!

If time allows, we’ll come back to some of these tools at the end of the course.
Readings (this lecture)

Required:

Introduction to Unix commands: https://kb.iu.edu/d/afsk
Includes all the commands we discussed today, and a few more that you don’t need to know well, but are worth being aware of.

Recommended:

Survival guide for Unix newbies: http://matt.might.net/articles/basic-unix/
More thorough discussion, including advanced commands like grep

“GNU/Linux Command-Line Tools Summary” by Gareth Anderson
Comprehensive introduction to the command line and the UNIX/Linux design philosophy in general.
Readings (next lecture)

Required:

https://research.google.com/archive/mapreduce.html
This is the paper that originally introduced the MapReduce framework, and it’s still, in my opinion, an excellent place to start. Don’t worry too much about understanding every bit of the paper-- it’s written for computer systems engineers!

Recommended:
“Introduction to HDFS” by J. Hanson