The Command Line: Automation and Productivity for Scientists

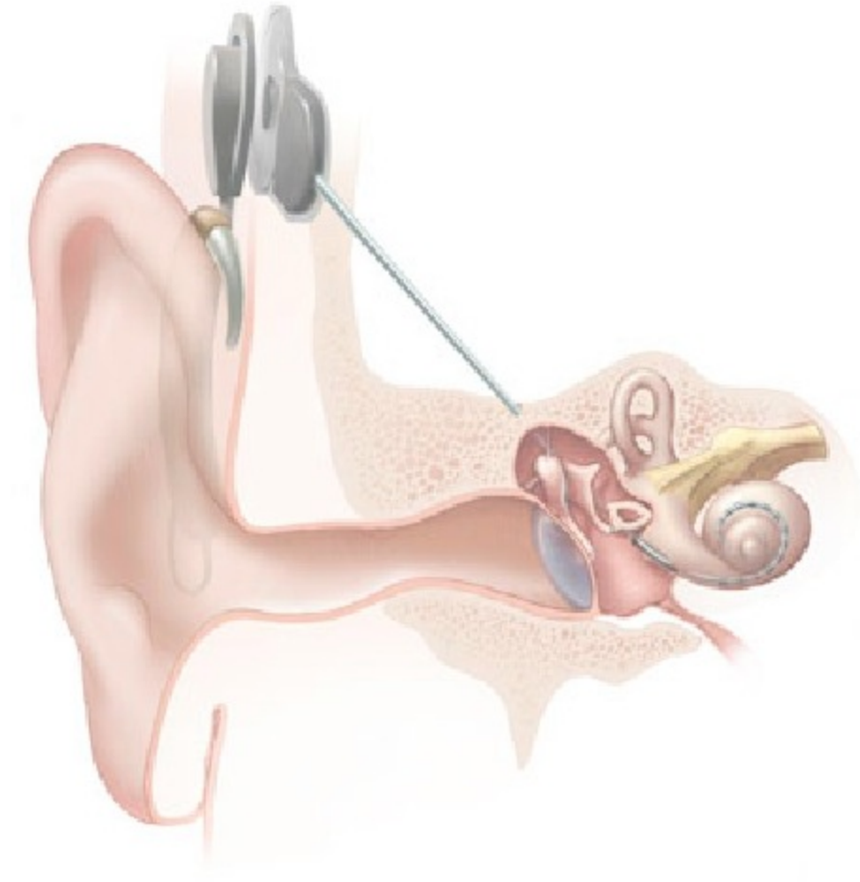
Jonathan Dursi, SciNet

SciNet

Cochlear Implants

A cochlear implant is a small electronic device that is surgically implanted in the inner ear to give deaf people a sense of hearing. More than a quarter of a million people have them,but there is still no widely-accepted benchmark to measure their effectiveness. In order to establish a baseline for such a benchmark,our supervisor got teenagers with CIs to listen to audio files on their computer and report:

- the quietest sound they could hear
- the lowest and highest tones they could hear
- the narrowest range of frequencies they could discriminate



http://en.wikipedia.org/wiki/File:Cochlear_implant.jpg

SciNet

Cochlear Implants

To participate,subjects attended our laboratory and one of our lab techs played an audio sample,and recorded their data - when they first heard the sound,or first heard a difference in the sound. Each set of test results were written out to a text file,one set per file.

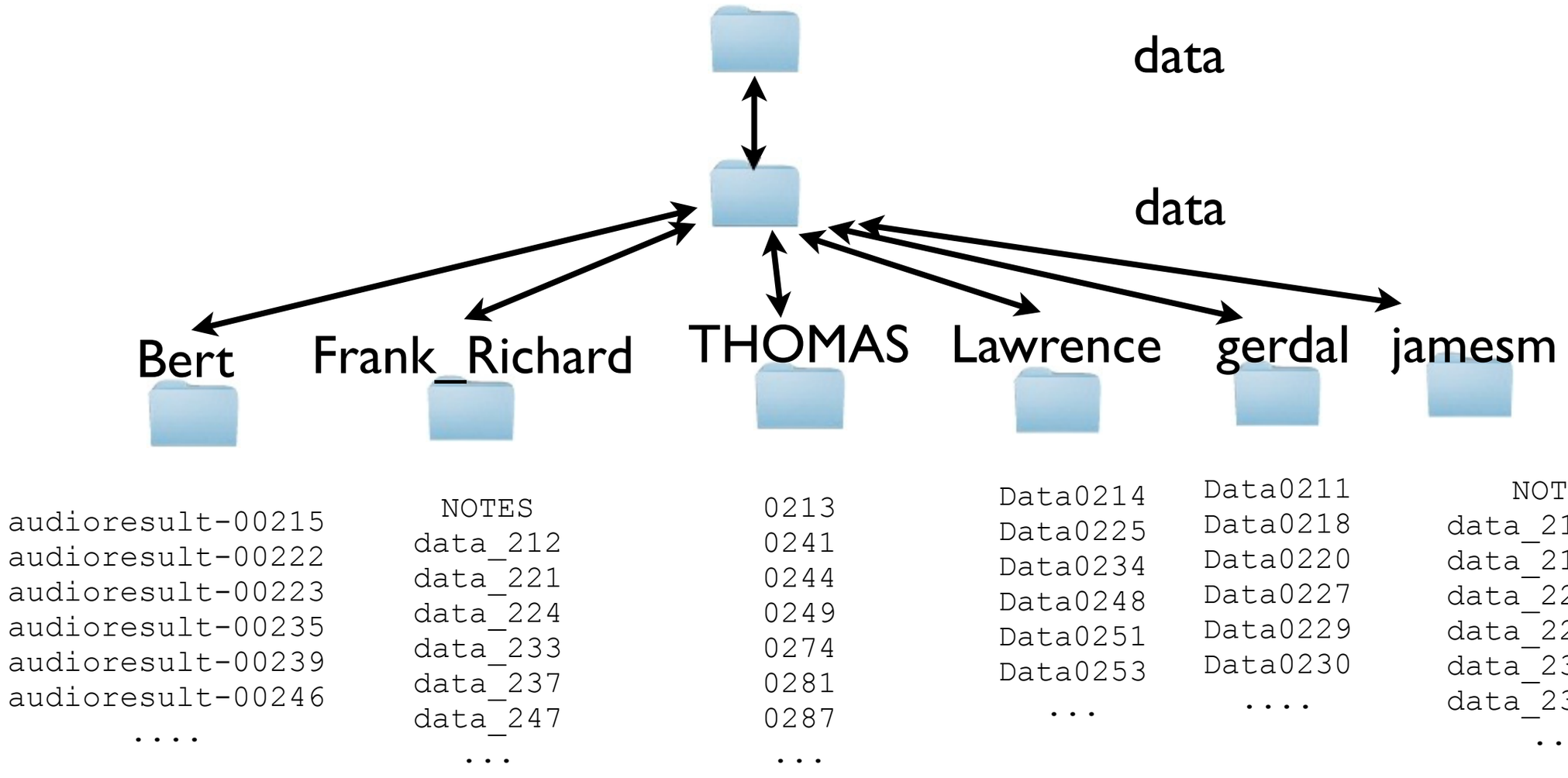
Each participant has a unique subject ID,and a made-up subject name.

Each experiment has a unique experiment ID.

http://en.wikipedia.org/wiki/File:Cochlear_implant.jpg

SciNet

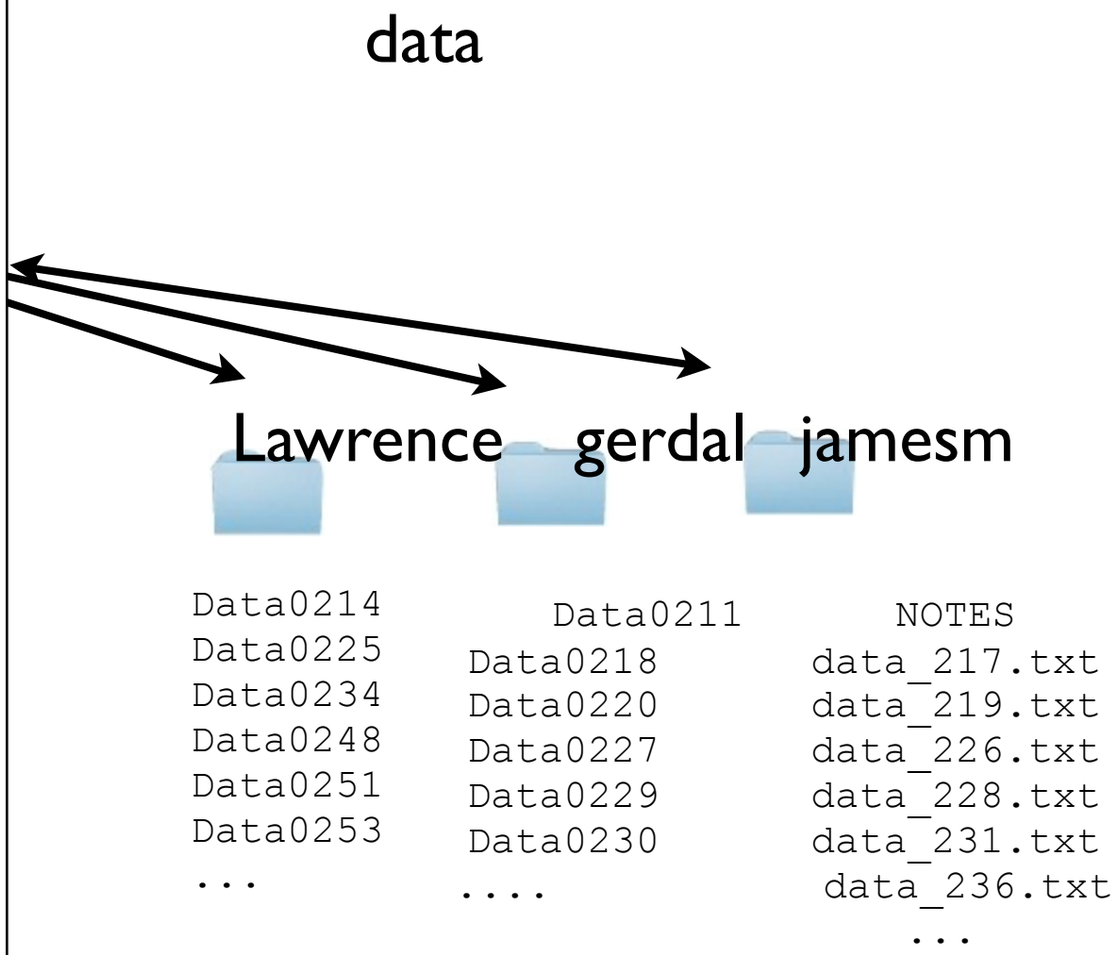
Data is a bit of a mess



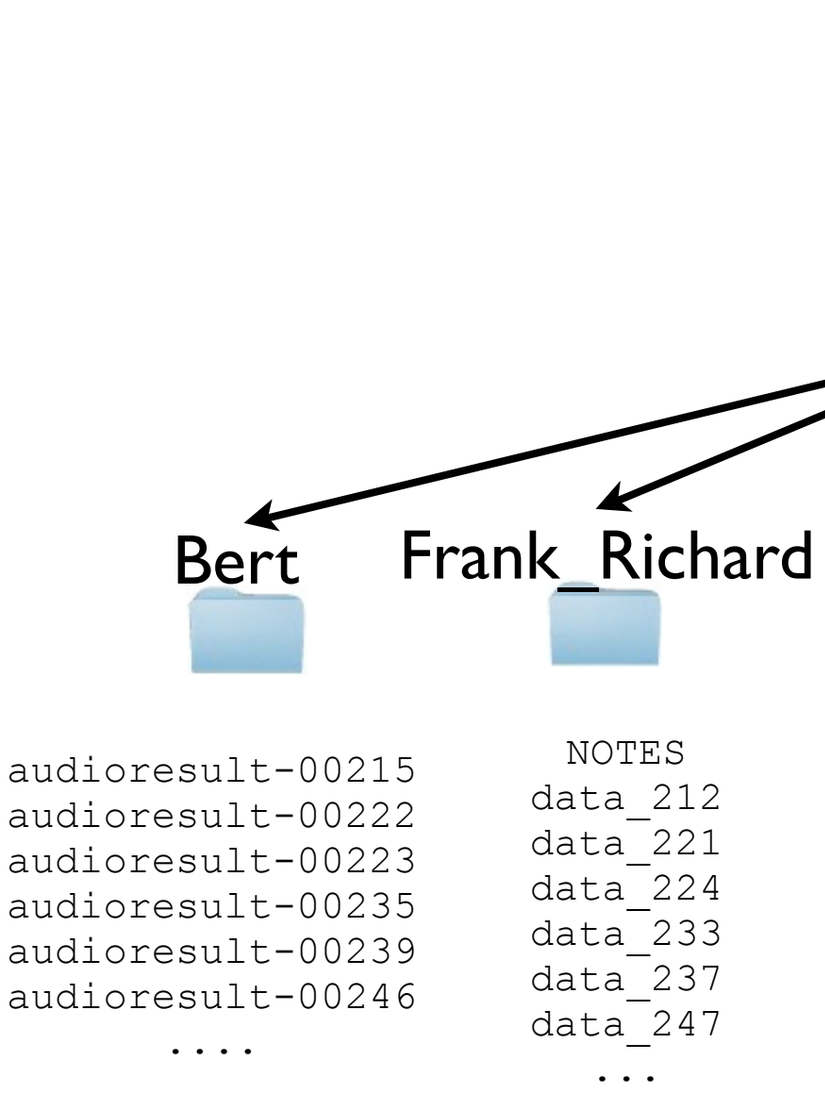
SciNet

Data is a bit of a mess

- Inconsistent file names
- Some directories have extraneous NOTES file
- multiple directories.



Data is a bit of a mess



Our mission:

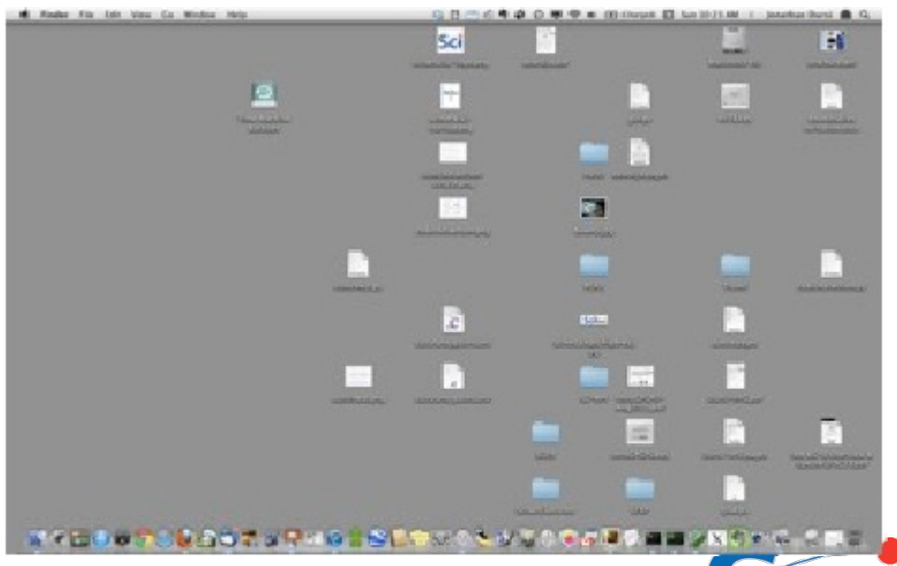
- Make **one** directory (alldata)
  - have all *data* files in there,all with .txt extension
- Get rid of NOTES files.

Shell vs GUI

- Presents a Command Line Interface (CLI,or CUI) vs GUI interface to your computer.

- Why on earth would you use a command line interface?

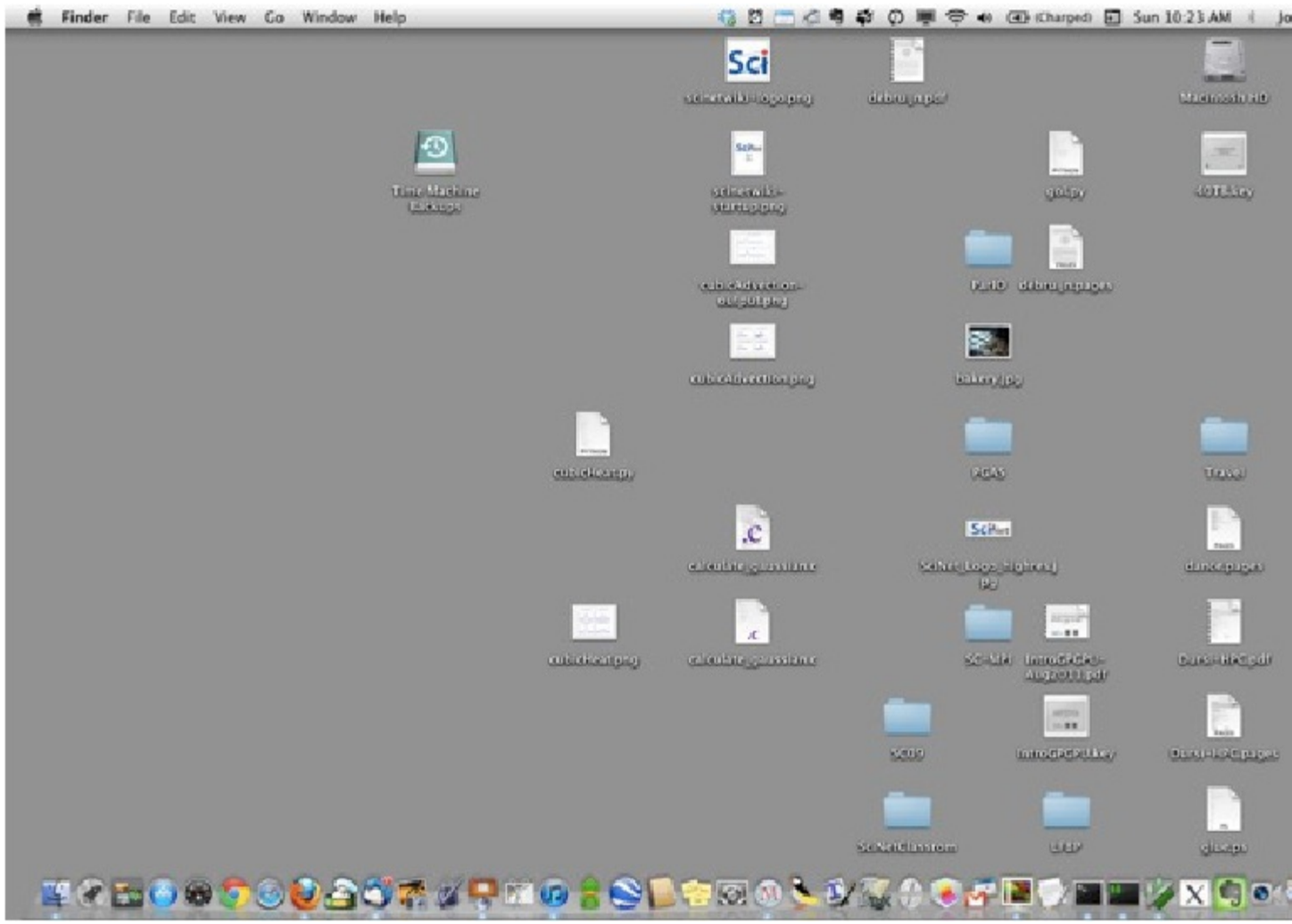
vs.



SciNet

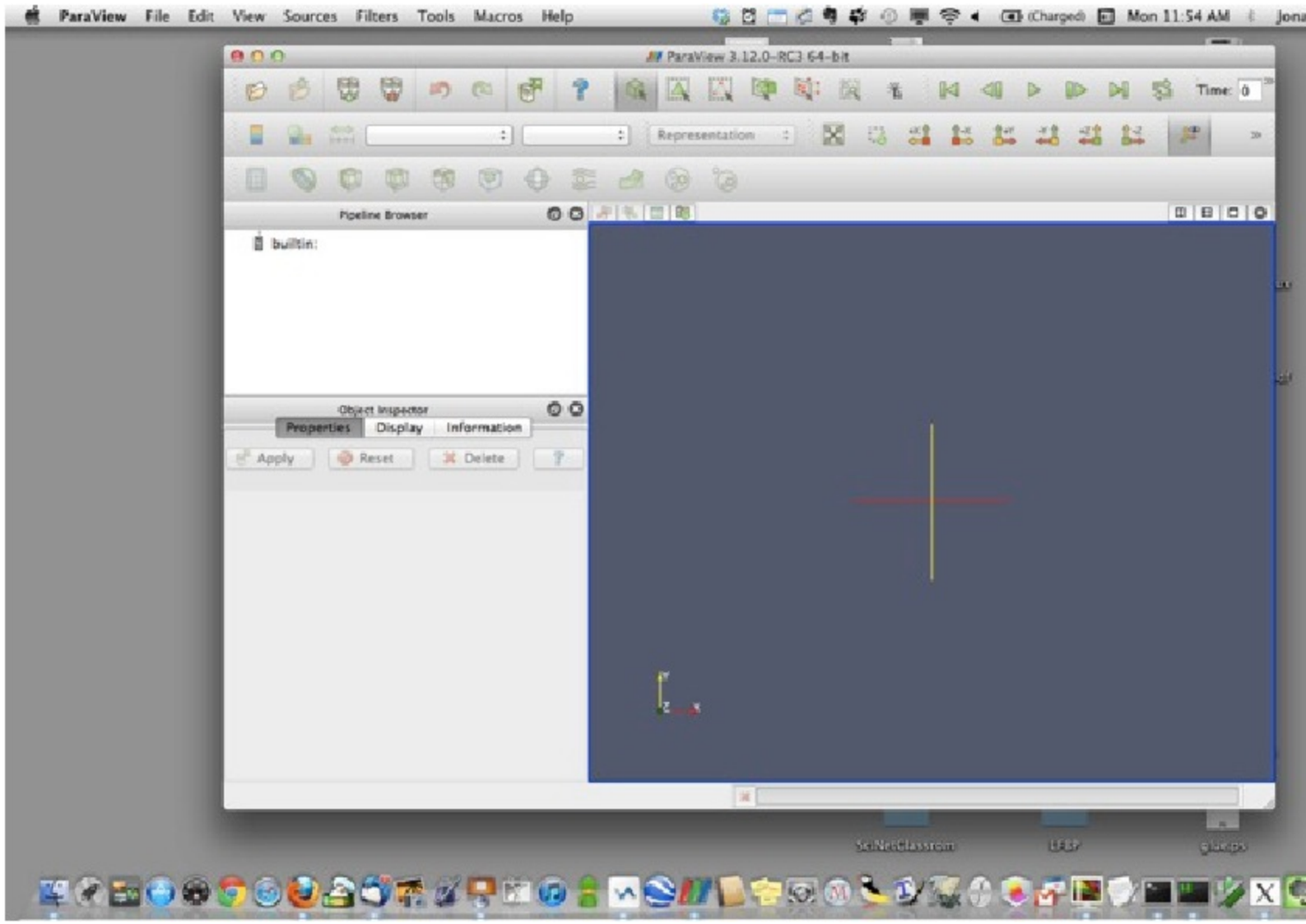
GUI:Operating

- Very good at **operating** an existing system.
- Click on *existing* controls, use *existing* functionality.



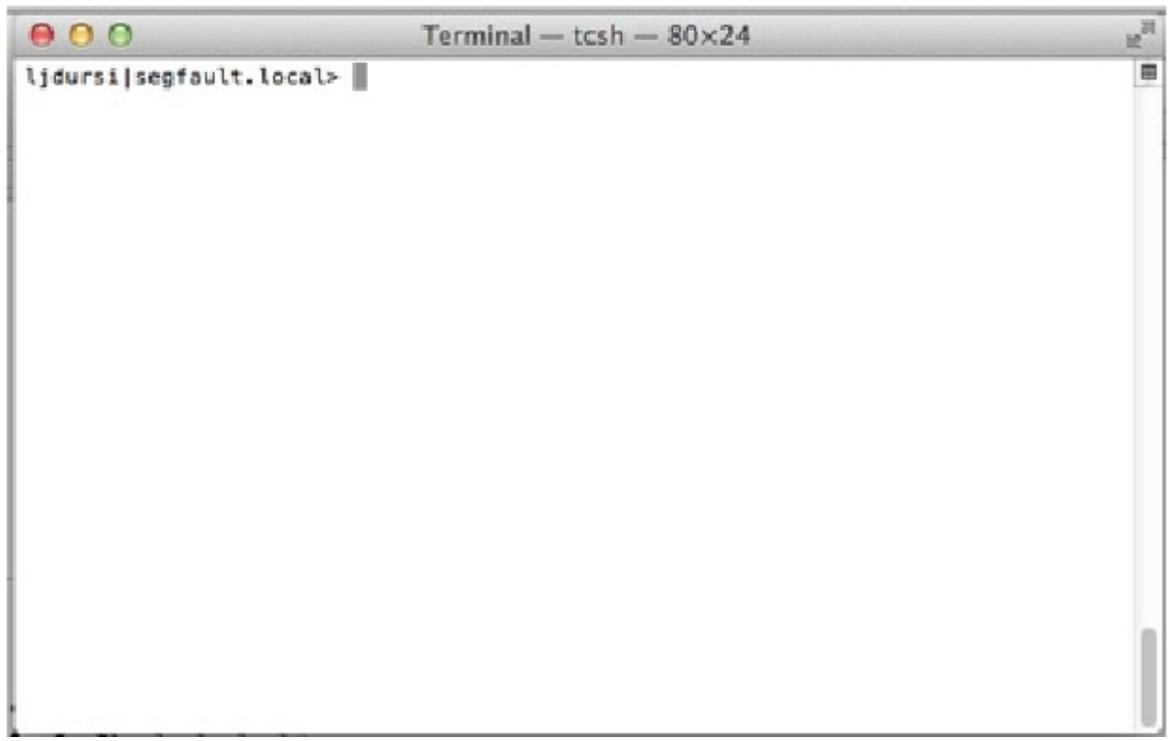
GUI:Operating

- Useful for basic computer operations,
- Operating existing software packages.



CLI - creating

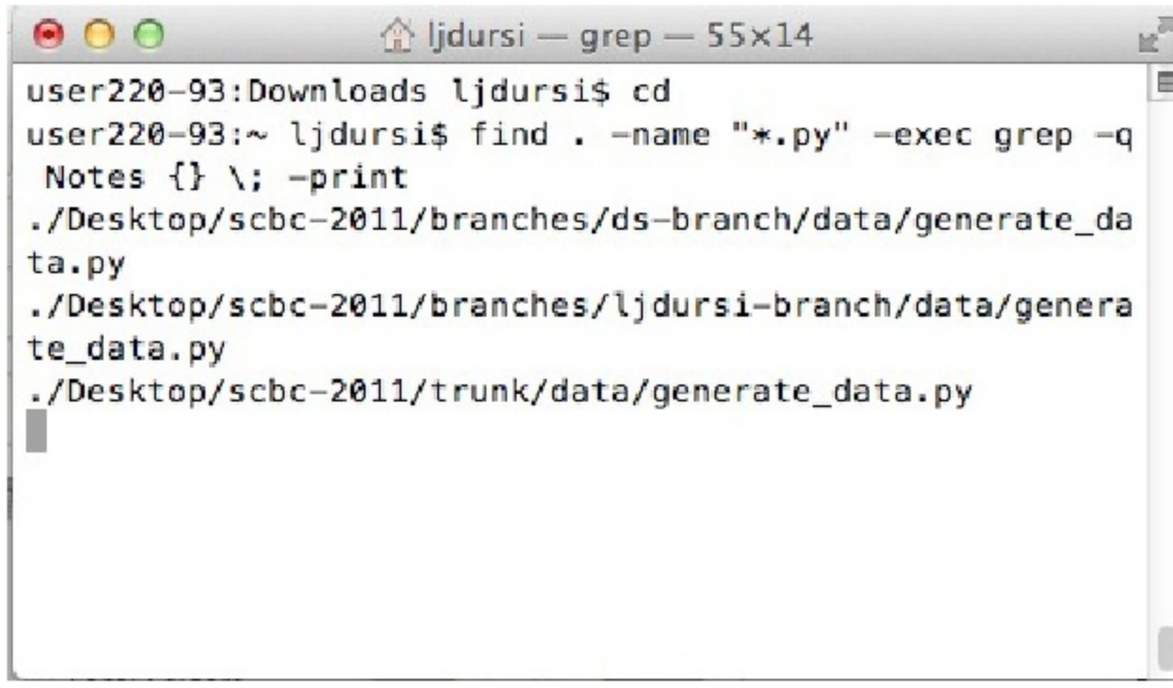
- For better or worse, a blank canvas
- Good for creating/ expressing new things.
- General programming in a GUI hard



SciNet

CLI - replicatable

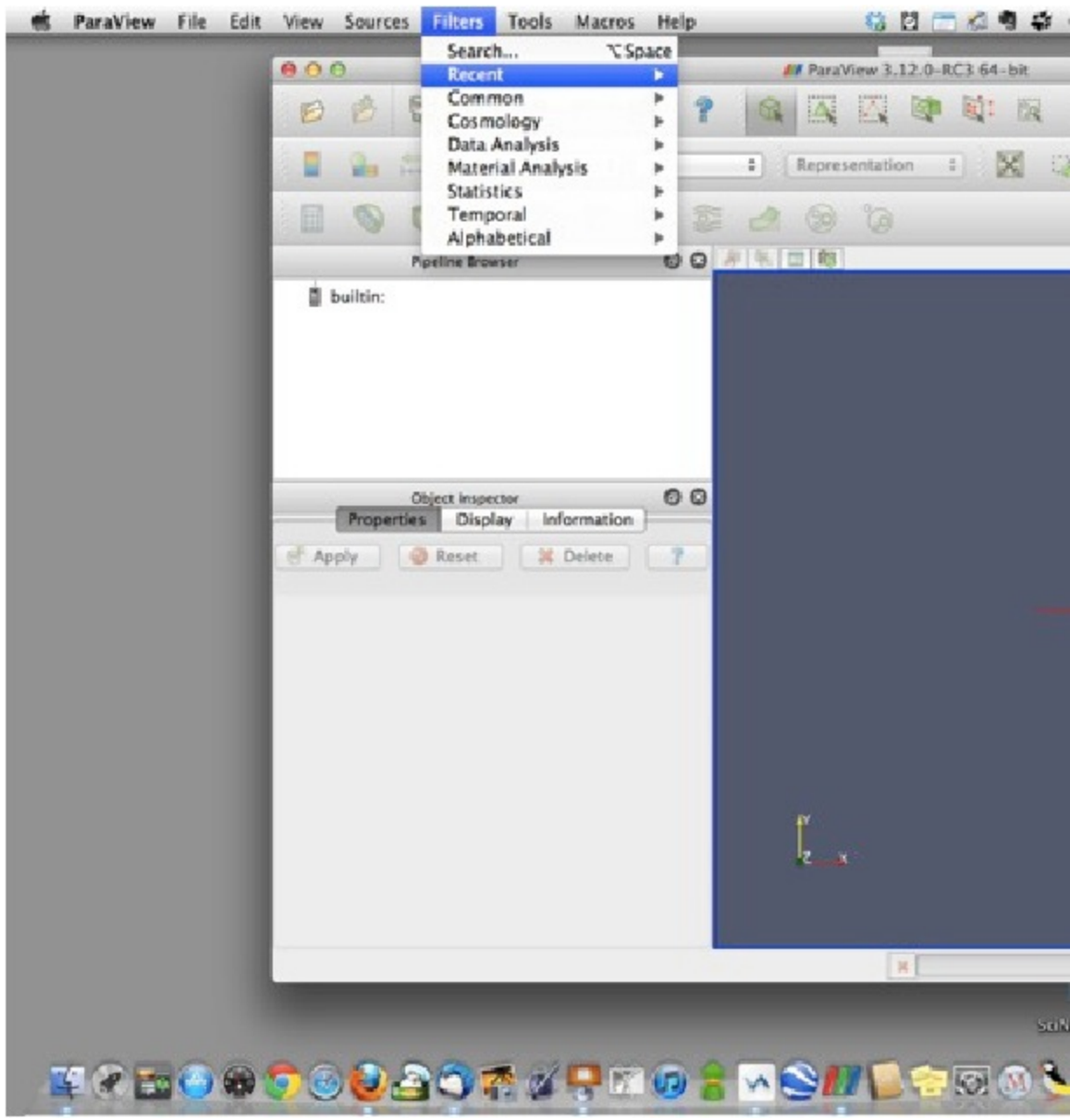
- Command lines can be cryptic to learn,but once you have the command, you can communicate it to others exactly.



SciNet

GUIs - not as replicatable

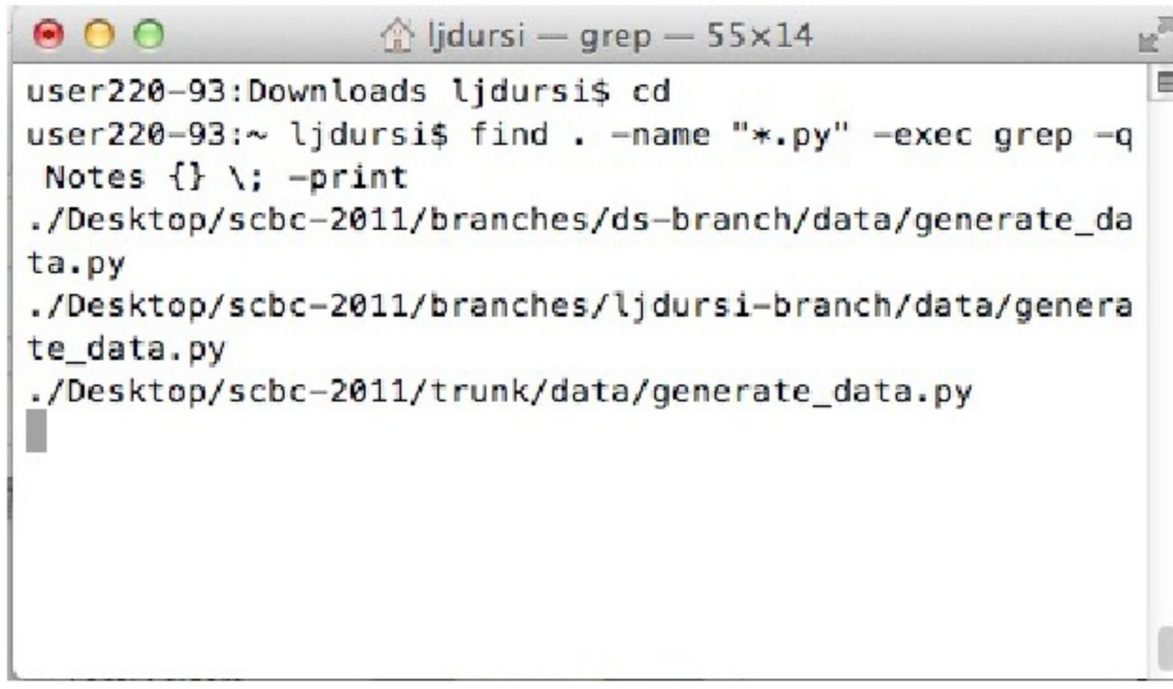
- "Click on Filters,then 'Recent'"
- "Then drag the green arrow down to the big grey box.."
- "...No,the other one.."
- "...Not there!"
- "Ok,let's start again..."



SciNet

CLI makes you more productive

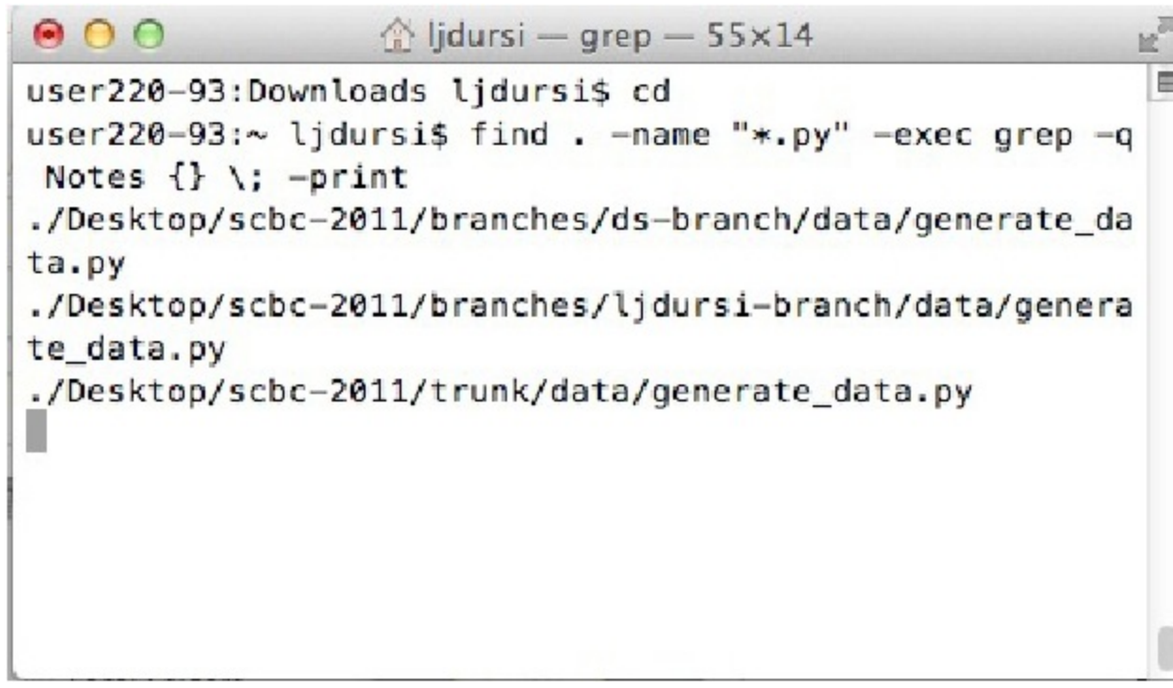
- Replicatable - stop wasting time re-discovering how to do things
- Automatable - can do the same thing hundreds of times easily without wasting time
- More time doing research



SciNet

CLI makes you more productive

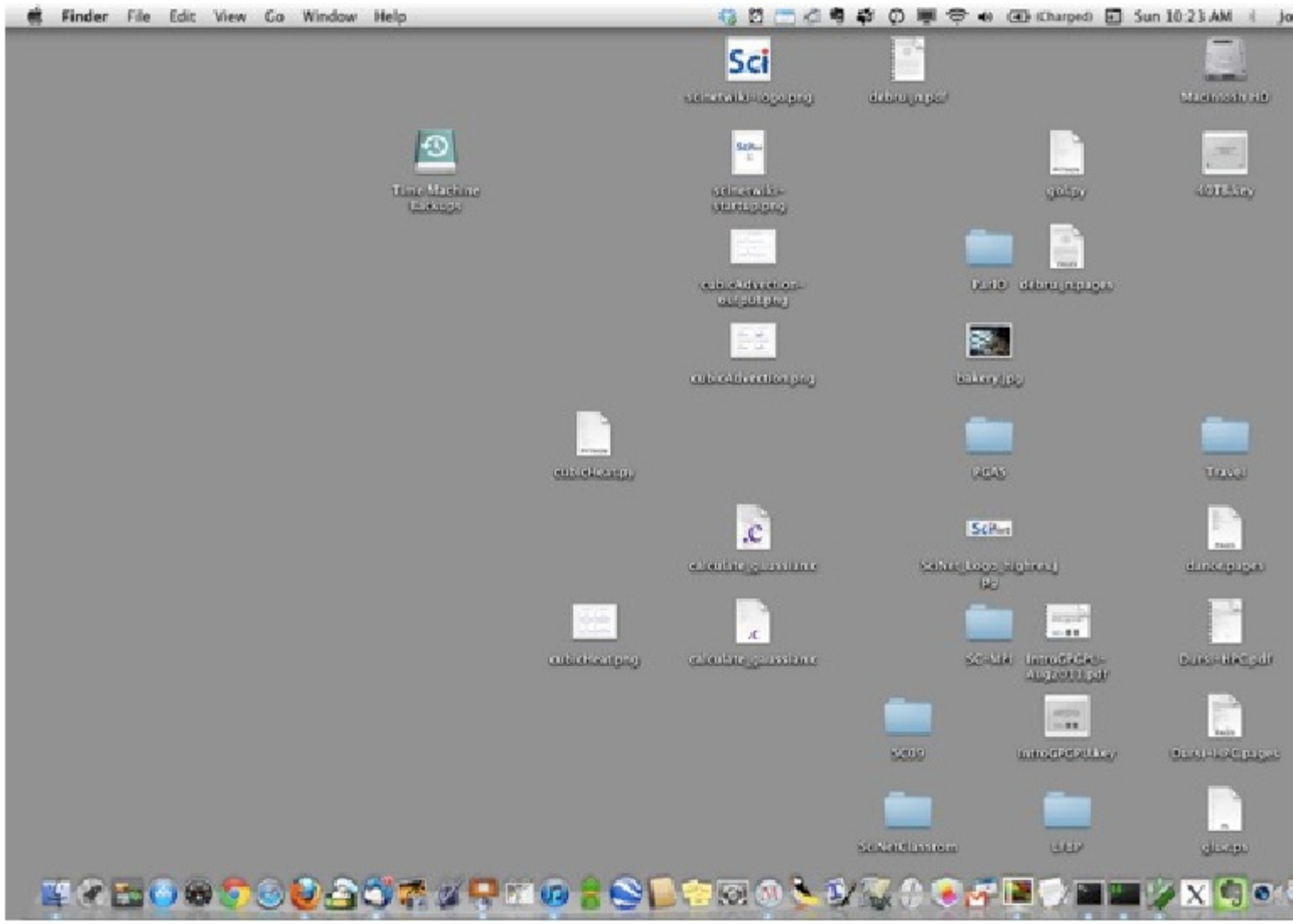
- But there's a learning curve.
- Investment in future productivity.



SciNet

GUI - Easy / Hard

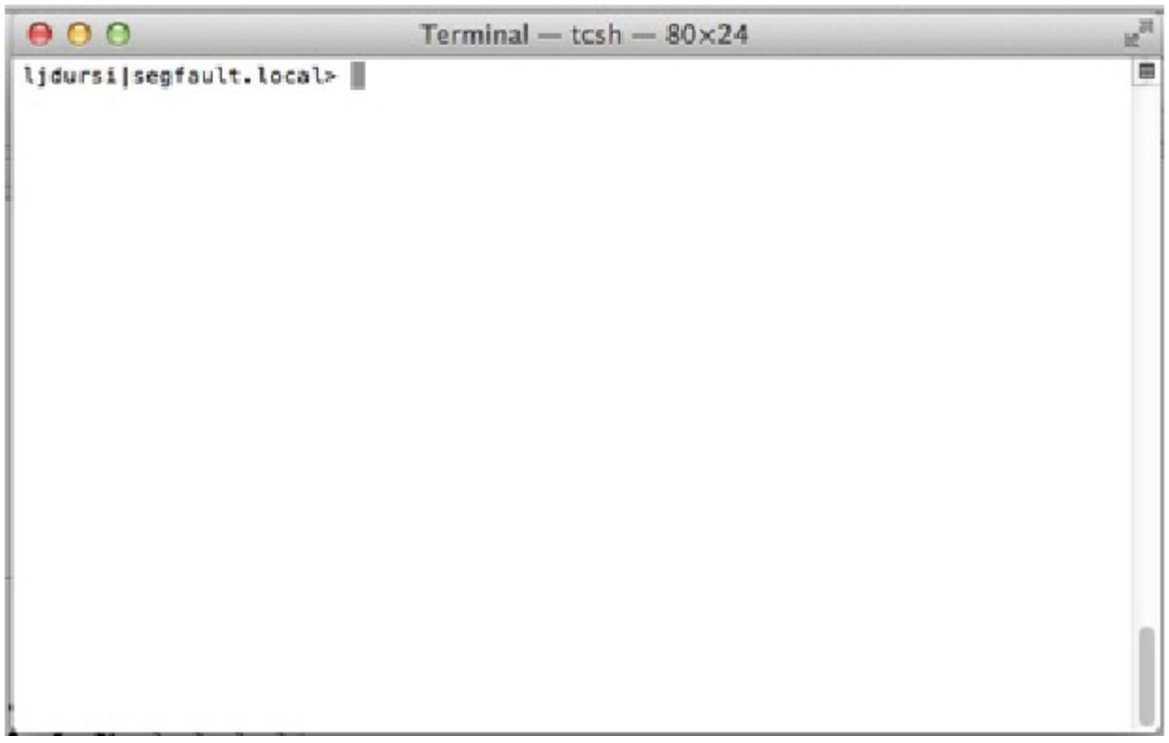
- Easy to *learn/ discover*
- Hard to *use* for big tasks productively.



SciNet

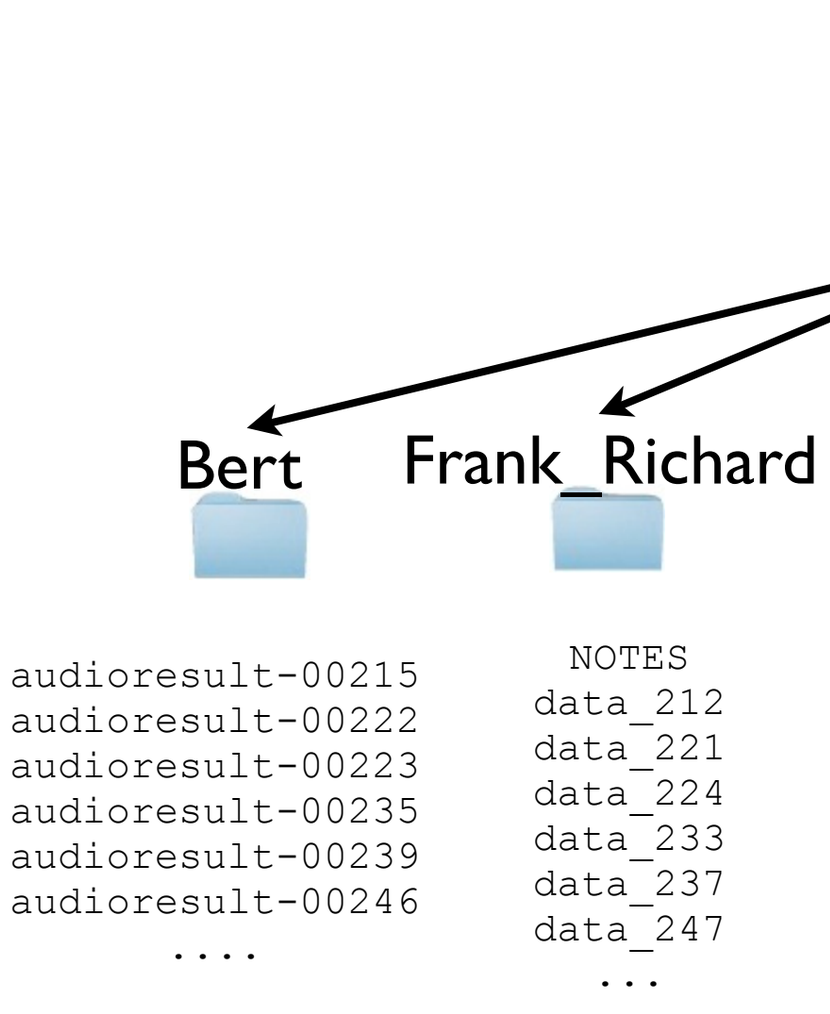
CLI - Hard/Easy

- Hard to *learn/ discover*
- Easy to *use* for big tasks productively.



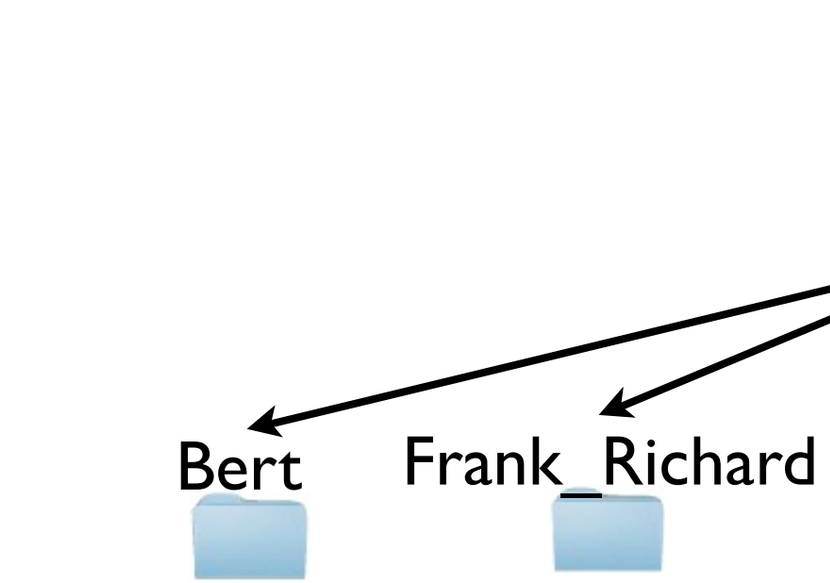
SciNet

GUI vs CLI



- With GUI,we could (painfully) do this one file at a time.
- But 2 months later when there's another 350 files, have to do exactly same thing again.
- No further ahead.

GUI vs CLI



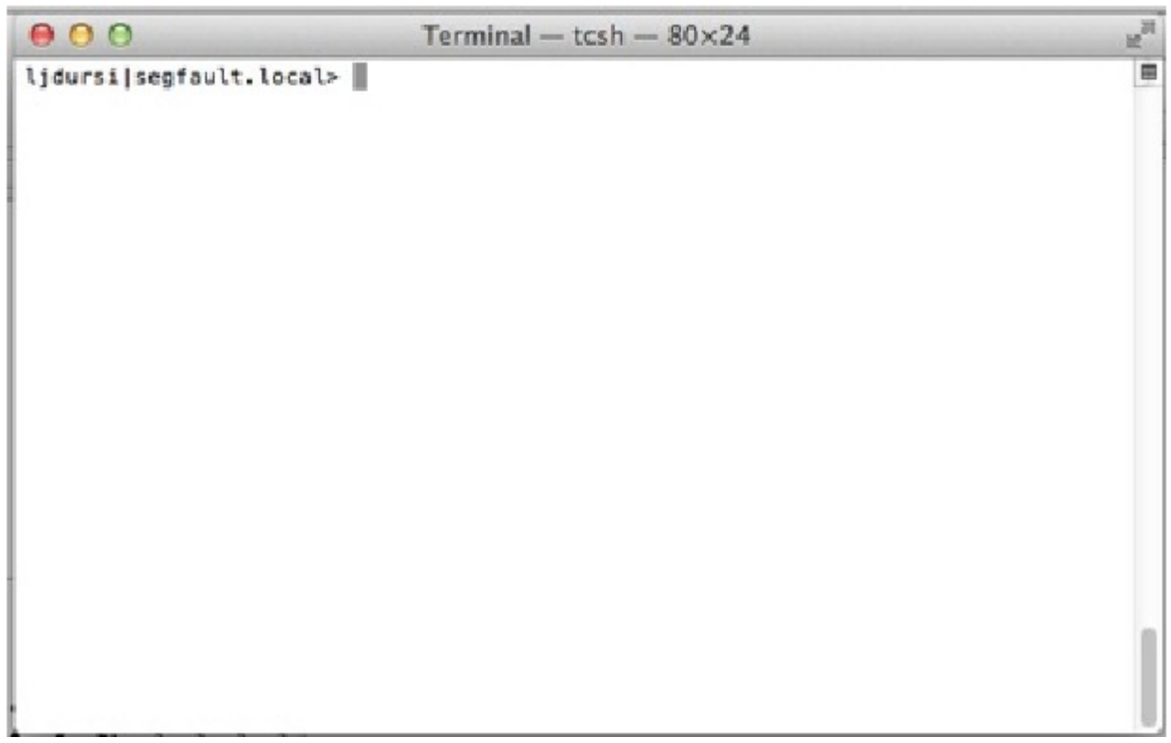

```
audioresult-00215
audioresult-00222
audioresult-00223
audioresult-00235
audioresult-00239
audioresult-00246
....
```

```
NOTES
data_212
data_221
data_224
data_233
data_237
data_247
...
```

- We're going to spend a lot of time today learning to do this in the shell.
- But next time, just re-run script and it's done.
- More Productive.

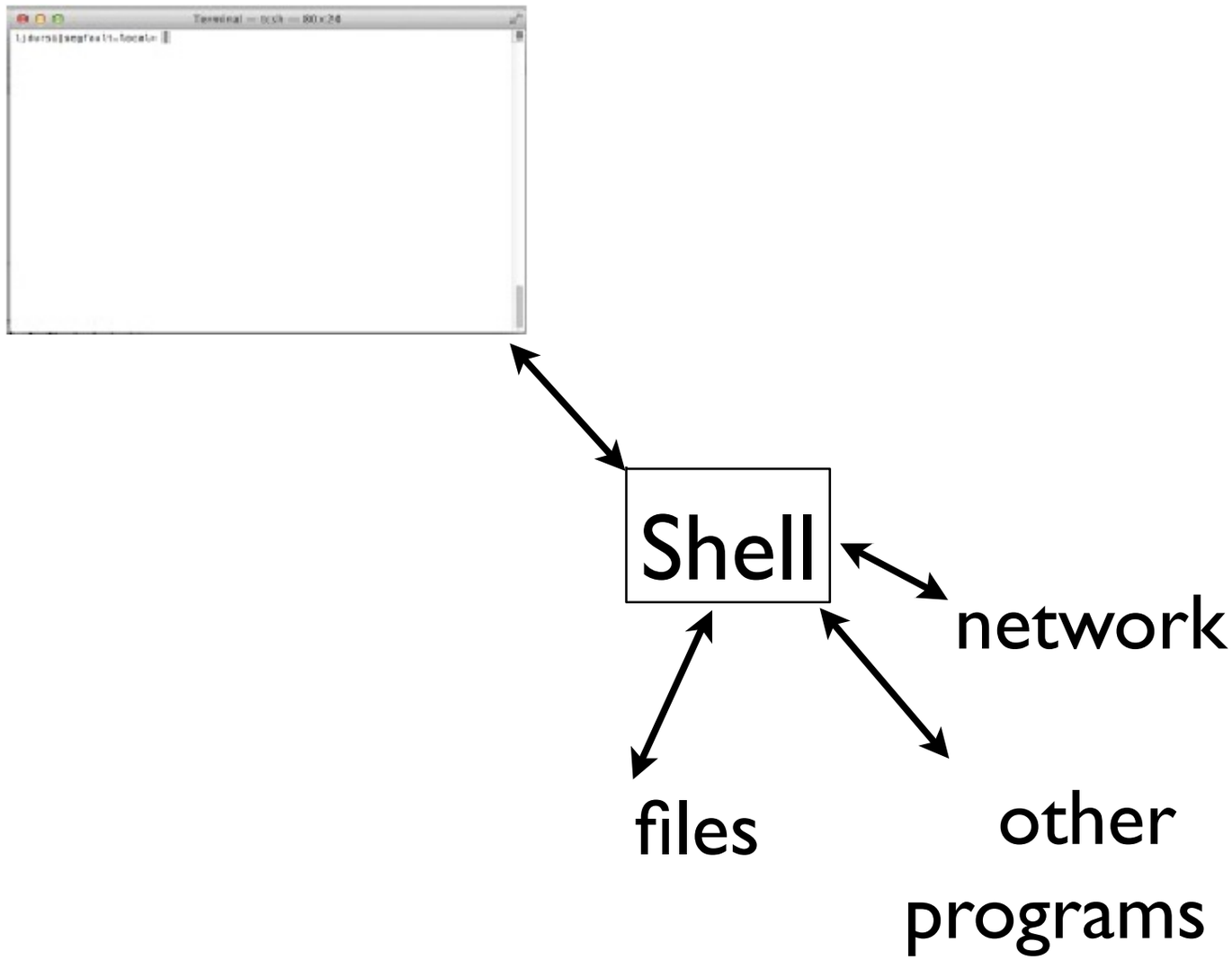
Open a Terminal

- Mac: Applications/ Utilities/Terminal. (May as well drag this to the dock)
- Windows:Start up MobaXTerm
- Linux:Various.



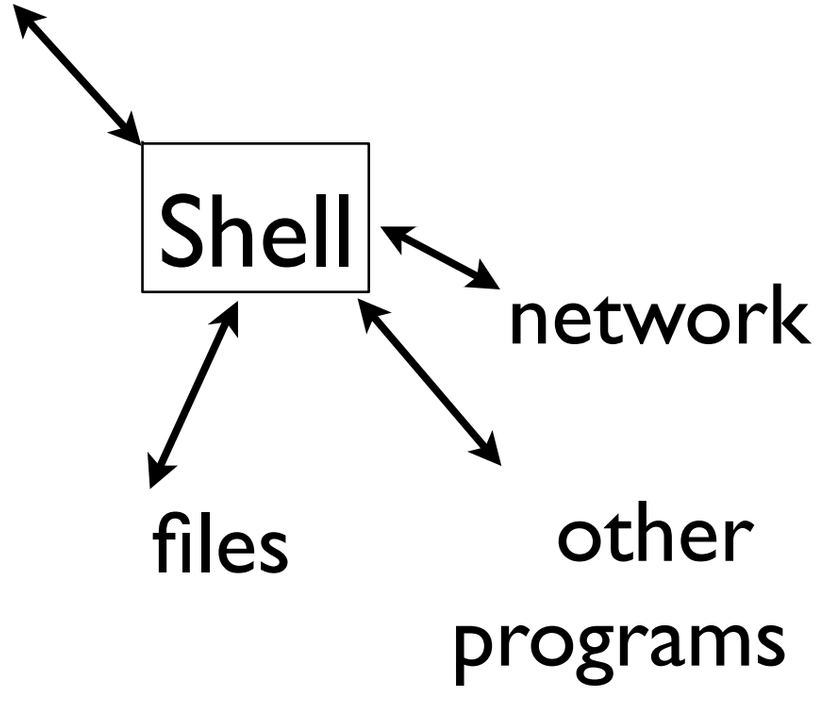
Terminal launches a shell

- When you use a terminal,you're interacting with the shell
- A program provides access to files,network, other programs.



Terminal launches a shell

- You type in commands
- Shell interprets them
- Performs actions on its own,or (more often) launches other programs



SciNet

“The” shell

- The shell most commonly used in linux is bash (Bourne-Again SHell).
- There are others;mostly the same but some syntax is different.
- Windows power shell - many similarities
- Type `hello=”world”` (no spaces).
- If you get an error about no command you’re probably running tcsh. Type“bash” to start a bash shell and try again.

SciNet

Basics - echo

- Let's start by having the shell greet you:

```
segfault:~ ljdursi$ hello="world"

segfault:~ ljdursi$ echo Hello, world
           Hello, world

segfault:~ ljdursi$ echo Hello, $hello
           Hello, world
```

Basics - File system

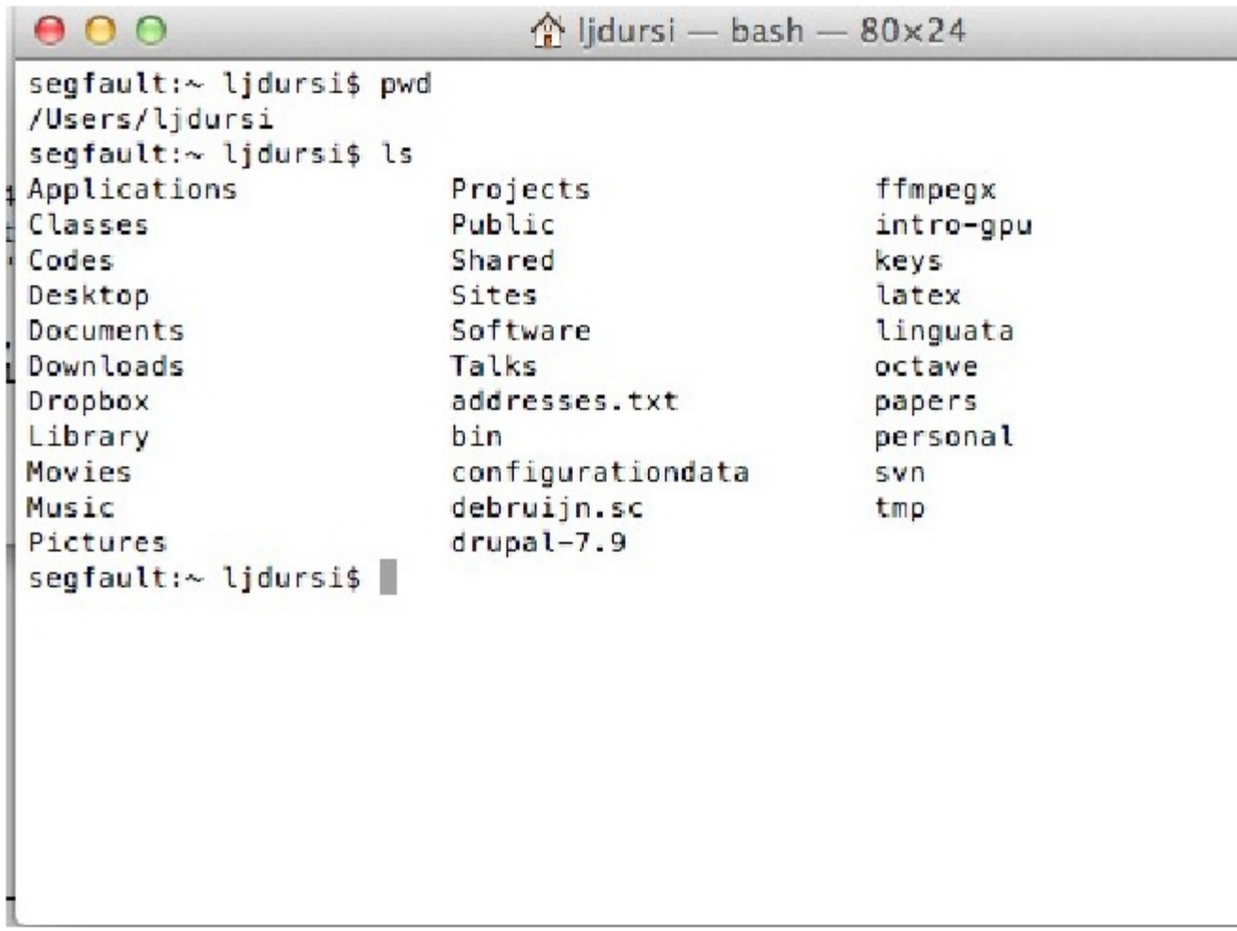
- Now let's learn how to start moving around amongst our files and directories.
- This is easy to do in a GUI (click on folders), harder here,but you get very fast at it in the shell...

Basics - File system

- Let's start poking around

- Type `pwd`. Prints "working"directory - where you are in the file structure

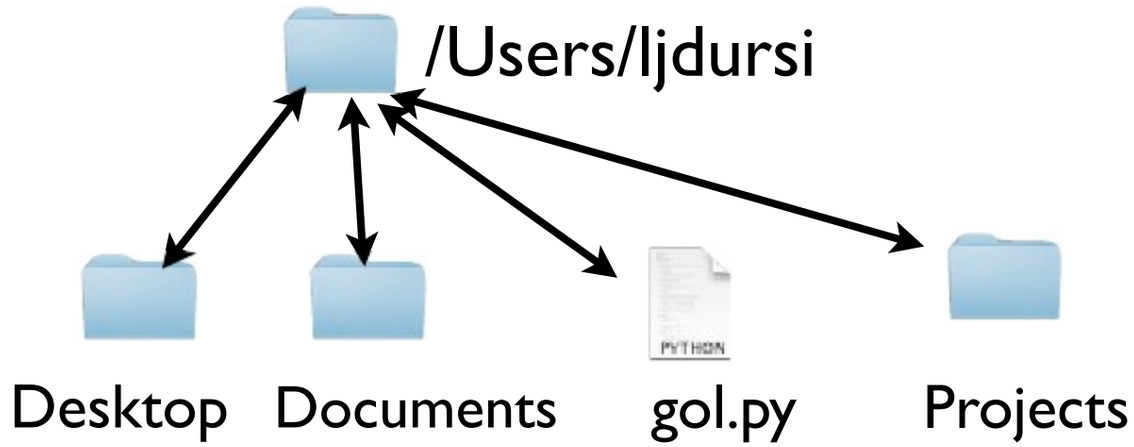
- Type `ls` - that will list the files in that directory



SciNet

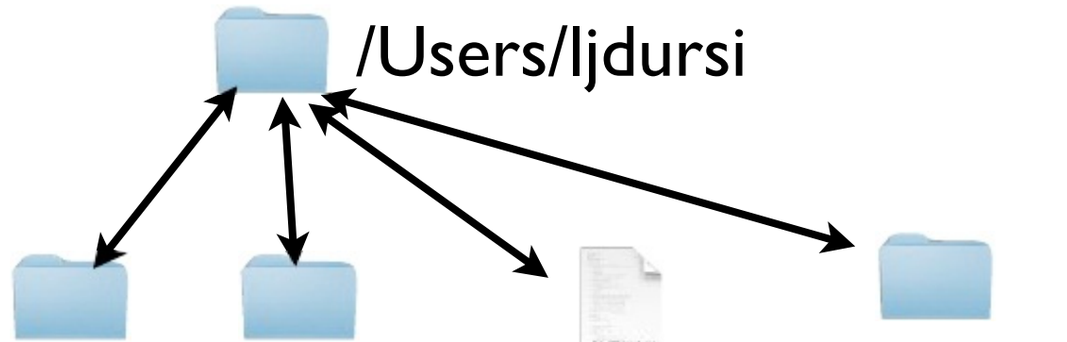
Directories = folders

- Often called folders because of how they're represented in GUIs
- Directories are listings of files - can be data files or other directories



Start at Home

- When you launch a shell,it starts in your home directory
- /Users/[username] or /home/[username] or something
- Top directory of all your stuff



File types

- Would like to know which entries are directories,which are plain files

- ls -F :labels directories with ‘/’,executables with‘*’,etc.

```
segfault:~ ljdursi$ ls -F
Applications/    addresses.txt
Classes/         bin/
Codes/           configurationdata/
Desktop/         debruijn.sc
Documents/       drupal-7.9/
Downloads/       gol.py*
...
```

SciNet

Changing Directories:cd

- Choose one of the directories in your home directory and type `cd [dir]`
- And then `ls -F`
- Listing of contents of new directory
- `cd` without arguments will return to home dir

```
segfault:~ ljdursi$ cd Desktop

segfault:Desktop ljdursi$ ls -F
40TB.key            cubicAdvection.png
Dursi-HPC.pages     cubicAdvection.py
Dursi-HPC.pdf       cubicHeat.png
IntroGPGPU.key      cubicHeat.py
LFBP/               dance.pages
...

segfault:~ ljdursi$ cd

segfault:~ ljdursi$ pwd
/Users/ljdursi
```

SciNet

Commands so far

- A couple things to observe:
- Commands designed to be fast/easy to *use*.
- Pretty cryptic to *learn*.

`echo`	Prints output
`pwd`	Print current directory
`cd [directory]`	Change directory
`cd`	Change directory to home
`ls`	Directory LiSting
`ls -F`	LiSting with Filetypes

SciNet

Options: -something

- ls (and it turns out lots of others) have options
- eg,-F
- or --help
- How do we know what the options are?

`echo`	Prints output
`pwd`	Print current directory
`cd [directory]`	Change directory
`cd`	Change directory to home
`ls`	Directory LiSting
`ls -F`	LiSting with Filetypes

SciNet

Manual:man pages

- Most programs have a you already use;
- Good for finding out more about a command you already use;
- Less good for learning what a command does.

```
segfault:~ ljdursi$ man ls

LS(1)                                   BSD Ge

NAME
     ls -- list directory conten

SYNOPSIS
     ls [-ABCFGHLOPRSTUW@abcdefgh


DESCRIPTION
     For each operand that names
     other than directory, ls di
     well as any requested, asso
     tion.  For each operand tha
     type directory, ls displays
```

SciNet

Manual:man pages

- Many programs have gazillions of options.
- No human being who has ever lived has known all the options to ‘ls’ at same time.
- Over time you find a few that you find useful for your favourite commands.

```
segfault:~ ljdursi$ man ls

LS(1)                                   BSD Ge

NAME
     ls -- list directory conten

SYNOPSIS
     ls [-ABCFGHLOPRSTUW@abcdefgh
        [file ...]

DESCRIPTION
     For each operand that names
     other than directory, ls di
     well as any requested, asso
     tion.  For each operand tha
     type directory, ls displays
```

SciNet

Using ls on other directories

```
segfault:~ ljdursi$ pwd
/Users/ljdursi/Desktop

segfault:Desktop ljdursi$ ls -F /Users/ljdursi
Applications/    addresses.txt
Classes/         bin/
Codes/           configurationdata/
Desktop/         debruijn.sc
Documents/       drupal-7.9/
Downloads/       gol.py*
...
```

- If you give ls an argument,it will do the listing of that directory...

Using ls on other directories

```
segfault:~ ljdursi$ pwd
/Users/ljdursi/Desktop

segfault:Desktop ljdursi$ ls -F /Users/ljdursi/codes
FLASH2.5/                athena3.1/
Gadget-2.0.3-SP.tgz      vine1.01.tar.gz

segfault:Desktop ljdursi$
```

- If you give ls an argument,it will do the listing of that directory...

Using ls on other directories

```
segfault:~ ljdursi$ pwd
/Users/ljdursi/Desktop

segfault:Desktop ljdursi$ ls *.py
cubicAdvection.py   gol.py
cubicHeat.py

segfault:Desktop ljdursi$ ls /Users/ljdursi/*.py
/Users/ljdursi/gol.py
```

- ...or those files.

SciNet

The shell interprets arguments

- The shell takes my line "`ls *.py`"
- It looks for all files that are of the form [anything].py,
- and passes them as arguments to the ls command (`/bin/ls`).

```
segfault:~ ljdursi$ pwd
/Users/ljdursi/Desktop

segfault:Desktop ljdursi$ ls *.py
cubicAdvection.py   gol.py
cubicHeat.py

segfault:Desktop ljdursi$ ls /Users/ljdursi/*.py
/Users/ljdursi/gol.py
```

SciNet

The shell interprets arguments

- `echo *.py` works just as well;
- Shell generates list of .py files, puts them as arguments to `echo`
- `echo` echos them to screen.

```
segfault:~ ljdursi$ pwd
/Users/ljdursi/Desktop

segfault:Desktop ljdursi$ ls *.py
cubicAdvection.py   gol.py
cubicHeat.py

segfault:Desktop ljdursi$ ls /Users/ljdursi/*.py
/Users/ljdursi/gol.py
```

SciNet

The shell interprets arguments

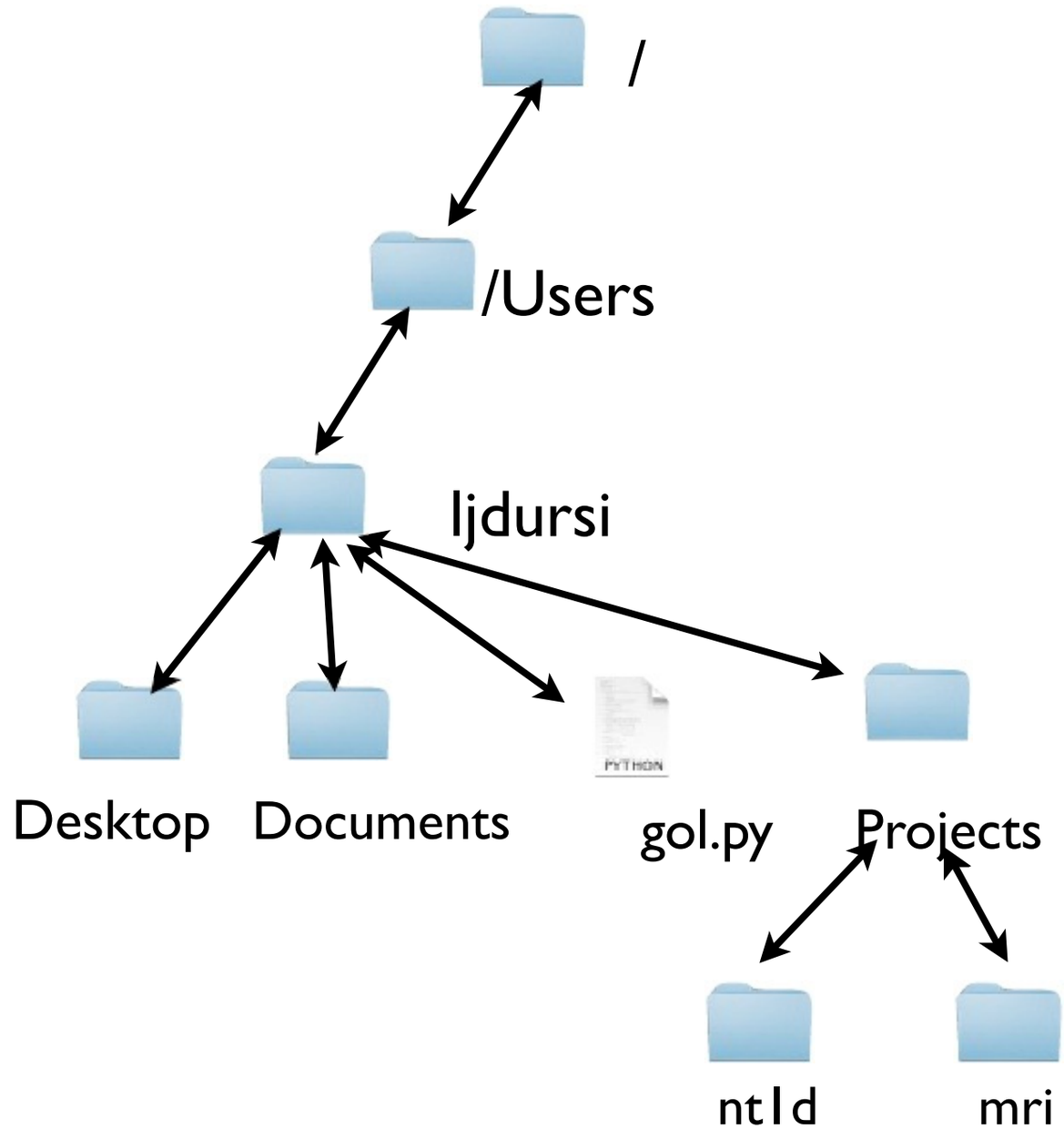
- If the argument is a directory (or a file name),there's no processing to be done
- Passes it to 'ls'

```
segfault:~ ljdursi$ pwd
/Users/ljdursi/Desktop

segfault:Desktop ljdursi$ ls -F /Users/ljdursi/codes
FLASH2.5/              athena3.1/
Gadget-2.0.3-SP.tgz  vine1.01.tar.gz
```

SciNet

Directories in the shell

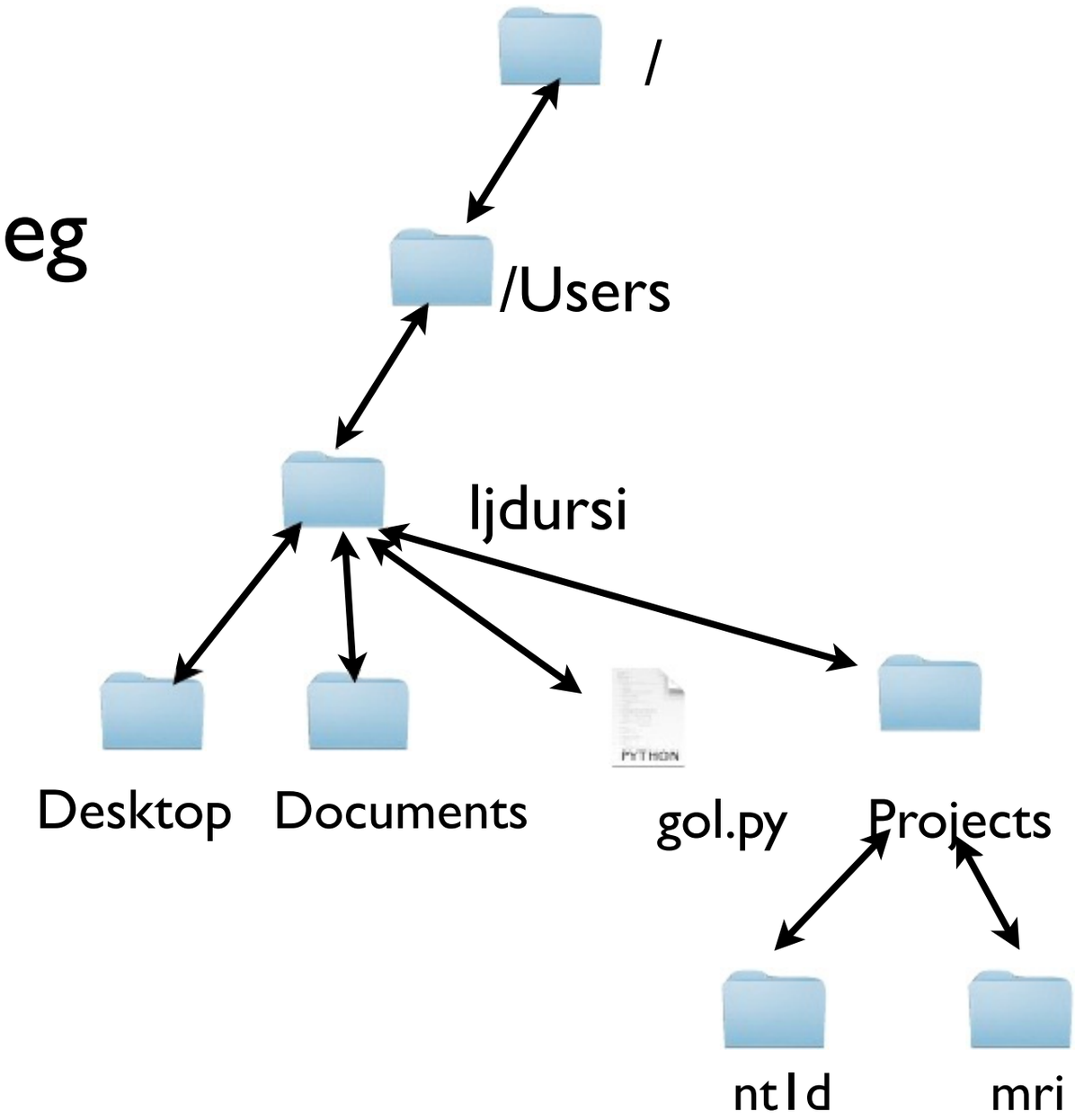
- A couple things to observe:
- Directories in bash separated by "/". (Windows - by "\").
- The top directory is"/";under that, Users,under that, ljdursi,etc.



Directories in the shell

- Can always specify a file by it's full "address",eg
  `/Users/ljdursi/Projects/mri/README.txt`

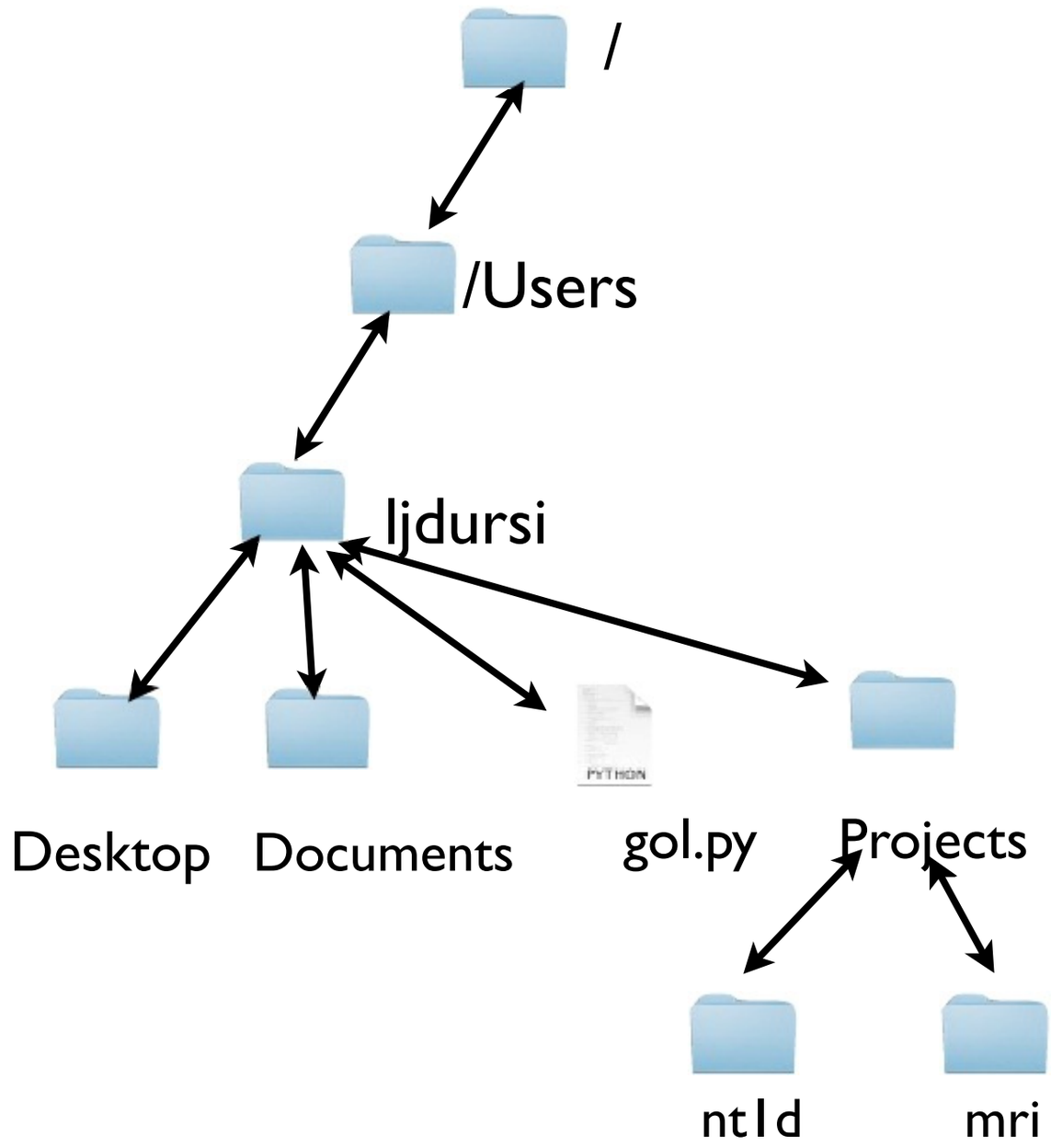
- If you are in that directory,can just say `README.txt`



SciNet

Directories in the shell

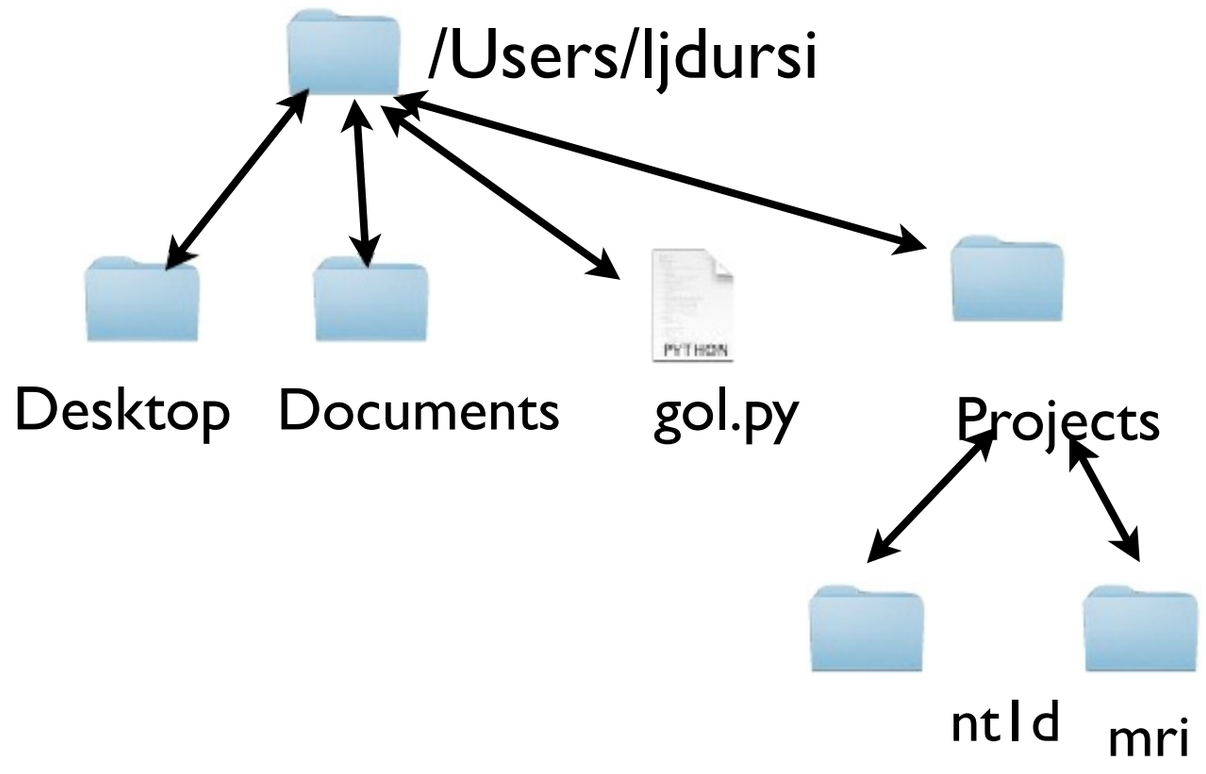
- But can also specify relative paths;if you're in Projects, `mri/README.txt` is enough.



SciNet

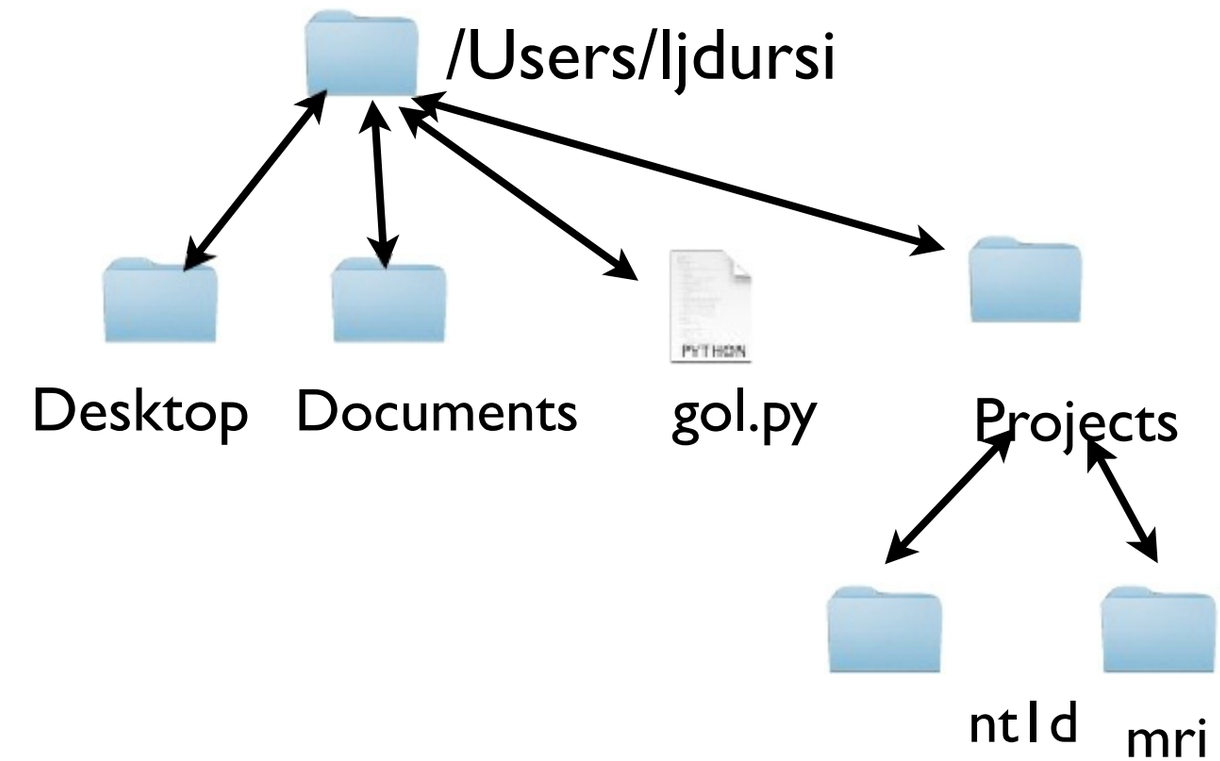
Shortcuts for moving around directories:

- A shortcut for "one directory up" is . .
- If I'm in Desktop, `ls ..` does an ls of home directory;
- and `ls ../Projects` looks in my Projects directory.



Shortcuts for moving around directories:

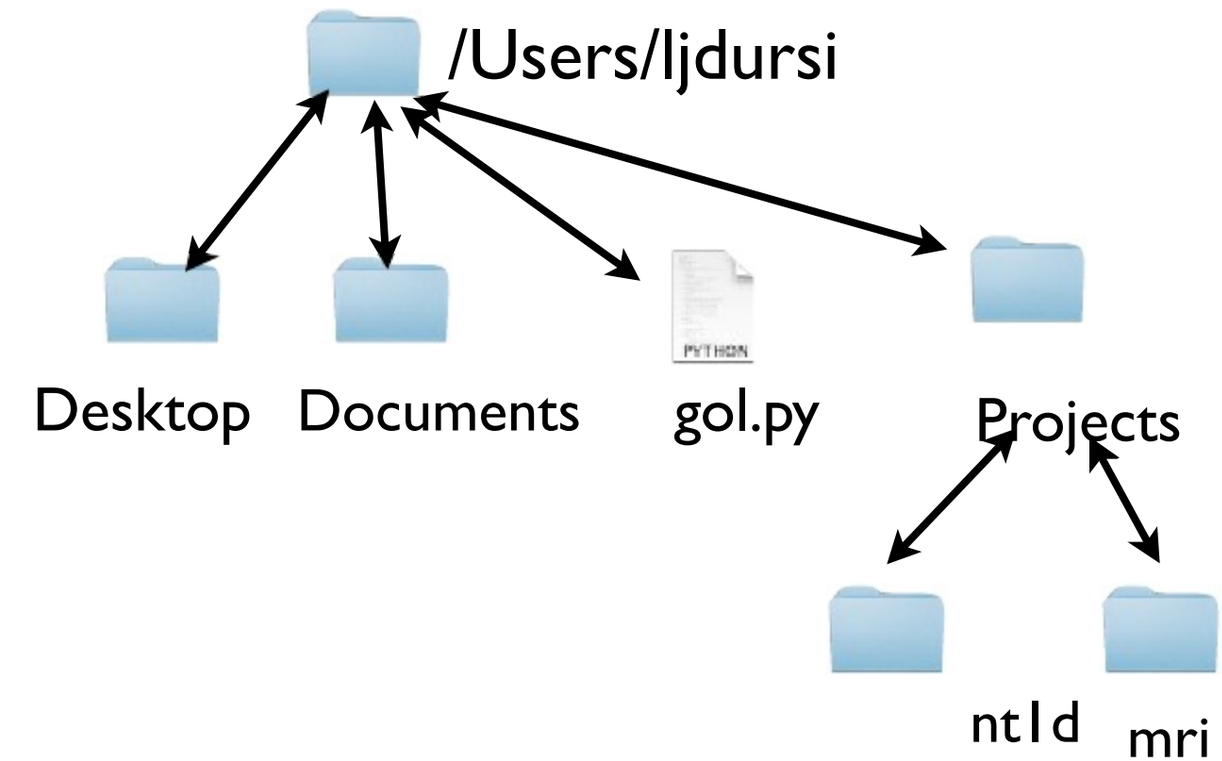
- One dot means the current directory: .
- If I'm in my home directory, `ls ./gol.py` just lists the gol.py there.



SciNet

Shortcuts for moving around directories:

- A shortcut for your home directory is `~`
- Wherever I am, `ls ~` does a listing of `/Users/ljdursi`
- `ls ~/Desktop` does a listing of `/Users/ljdursi/Desktop`.



SciNet

Looking at files

- Let's go into the data directory you downloaded

data

```
segfault:~ ljdursi$ cd ~/wherever/data

segfault:data ljdursi$ ls -F
data/           ex_data.txt     generate_data.py

segfault:data ljdursi$ cd data

segfault:data ljdursi$ ls -F
Bert/                Lawrence/  alexander/ jamesm/
Frank_Richard/    THOMAS/       gerdal/

segfault:data ljdursi$ cd Bert

segfault:Bert ljdursi$ ls
audioresult-00215 audioresult-00332 audioresult-00451
audioresult-00222 audioresult-00350 audioresult-00453
audioresult-00223 audioresult-00353 audioresult-00460
audioresult-00235 audioresult-00355 audioresult-00466
...

segfault:Bert ljdursi$
```

Looking at a file

```
segfault:~ ljdursi$ file audioresult-00215
audioresult-00215: ASCII text
```

echo	**Prints output**
pwd	**Print current directory**
cd [directory]	**Change directory**
cd	**Change directory to home**
ls	**Directory LiSting**
ls -F	**LiSting with Filetypes**
man [cmd]	**MANual page for [cmd]**
file [filename]	**What is in [filename]?**

SciNet

Looking at a file

```
segfault:Bert ljdursi$ file audioresult-00215
audioresult-00215: ASCII text

segfault:Bert ljdursi$ file au<TAB>
segfault:Bert ljdursi$ file audioresult-00
```

Tab completion!

If you hit <TAB> when typing a filename,shell will complete what you're typing (as much as possible)

Looking at a file

```
segfault:Bert ljdursi$ file audioresult-00215
audioresult-00215: ASCII text

segfault:Bert ljdursi$ file au<TAB>
segfault:Bert ljdursi$ file audioresult-00
```

Other handy tip -
Up arrow lets you preview previous commands;can edit and/or press <Return>

SciNet

Looking at a file

```
segfault:Bert ljdursi$ file audioresult-00215
audioresult-00215: ASCII text

segfault:Bert ljdursi$ more audioresult-00215
#
Reported: Sun Jun 26 14:56:54 2011
Subject: beyonceLennon177
Year/month of birth: 1993/09
Sex: N
CI type: 20
Volume: 8
Range: 5
Discrimination: 7

segfault:Bert ljdursi$
```

SciNet

Looking at a file

```
segfault:Bert ljdursi$ file audioresult-00215
audioresult-00215: ASCII text

segfault:Bert ljdursi$ more audioresult-00215
#
Reported: Sun Jun 26 14:56:54 2011
Subject: beyonceLennon177
Year/month of birth: 1993/0
Sex: N
CI type: 20
Volume: 8
Range: 5
Discrimination: 7

segfault:Bert ljdursi$
```

echo	**Prints output**
pwd	**Print current directory**
cd [directory]	**Change directory**
cd	**Change directory to home**
ls	**Directory LiSting**
ls -F	**LiSting with Filetypes**
man [cmd]	**MANual page for [cmd]**
file [filename]	**What is in [filename]?**
more [filename]	**Prints out filename(s) by page**
cat [filename]	**Dumps out filename(s)**

SciNet

More vs Cat:

- **try** `more au*`
- **and then** `cat au*`
- What's the difference?

(less - space for next page,q to finish current file)

SciNet

cat'ing files together

- Dumping all the files together is how 'cat' got its name - short for concatenate.
- Try cat'ing all the files together into a new file:

```
segfault:~ ljdursi$ cat au* > all-results
segfault:~ ljdursi$ more all-results
```

SciNet

Redirection

- `[cmd] > [filename]`
  takes what would have gone to the screen,creates a new file`[filename]`, and redirects output to the file.

- Overwrites previous contents of file if it had existed.

Redirection

- `[cmd] >> [filename]` *appends* to `[filename]` if it exists.

- `[cmd] < [filename]` - program's input comes from file,as if you were typing.

SciNet

cat - echos input

- If cat isn't given filenames,it just dumps its input to the screen.

```
segfault:Bert ljdursi$ cat
hello
hello
there
there
^D
```

cat - echos input

- Redirecting stdin means input from a file just as if you typed it:

```
segfault:Bert ljdursi$ cat < all-results
#
Reported: Tue Jul 26 14:01:03 2011
Subject: princeCarrot434
Year/month of birth: 1992/09
Sex: M
CI type: 16
Volume: 9
Range: 2
Discrimination: 4
```

mv,cp

- We've created our first file from the shell!
- We can make copies,or move the file around:

mv,cp

```
segfault:Bert ljdursi$ cp all-results all-results-2
segfault:Bert ljdursi$ ls all*
???

segfault:Bert ljdursi$ mv all-results all-results-3
segfault:Bert ljdursi$ ls all*
???

segfault:Bert ljdursi$ mv all-results-3 ..
segfault:Bert ljdursi$ ls all*
???
```

SciNet

mv,cp - move,copy

```
segfault:Bert ljdursi$ cp all-results all-results-2
segfault:Bert ljdursi$ ls all*
all-results all-results-2

segfault:Bert ljdursi$ mv all-results-2 all-results-3
segfault:Bert ljdursi$ ls all*
all-results all-results-3

segfault:Bert ljdursi$ mv all-results3 ..
segfault:Bert ljdursi$ ls all*
all-results

segfault:Bert ljdursi$ ls ..
Bert              Lawrence  alexander       gerdal
Frank_Richard     THOMAS    all-results-3   jamesm
```

rm - remove

- Deletes (ReMoves) file.
- Does *not* move it to trash;deletes it.
- No safety net!

```
segfault:Bert ljdursi$ ls ..
Bert            Lawrence  alexander      gerdal
Frank_Richard   THOMAS    all-results-3  jamesm
```

SciNet

rm

```
segfault:Bert ljdursi$ ls -F ..
Bert/                Lawrence/  alexander/     gerdal/
Frank_Richard/    THOMAS/    all-results-3  jamesm/

segfault:Bert ljdursi$ rm ../all-results-3

segfault:Bert ljdursi$ ls -F ..
Bert/                Lawrence/  alexander/     gerdal/
Frank_Richard/    THOMAS/    jamesm/
```

mkdir,rmdir

- To create and delete directories,use mkdir and rmdir.
- Uncharacteristically, rmdir protects you - you can’t delete a directory with files in it
- Have to delete them first

mkdir,rmdir

```
segfault:Bert ljdursi$ mkdir foo

segfault:Bert ljdursi$ ls foo

segfault:Bert ljdursi$ cp all-results foo

segfault:Bert ljdursi$ ls foo
all-results2

segfault:Bert ljdursi$ rmdir foo
 rmdir: foo: Directory not empty

segfault:Bert ljdursi$ rm foo/all-results

segfault:Bert ljdursi$ rmdir foo
```

SciNet

wc - word count of text files

- `wc [filename]` prints the lines,words, and characters (non-spaces) in a text file
- `wc -l`, `wc -w`, and `wc -c` print just the # of lines,words,and characters of the file
- try `wc all-results` (tab completion will work after the‘al’)

SciNet

wc

- We've just `wc`'ed a `cat`'ed file
- Should have same as totals of all files
- Let's try that: `wc  au*`

WC

```
segfault:Bert ljdursi$ wc all-results
     423     1124     6916 all-results

segfault:Bert ljdursi$ wc au*
...
       9       24      147 audioresult-00521
       9       24      146 audioresult-00532
       9       24      147 audioresult-00534
       9       24      151 audioresult-00535
       9       24      148 audioresult-00557
     423     1124     6916 total
```

Dealing with too much output

- `wc au*` printed out results for each file,and total - handy.
- But it provided too much output; couldn't see it all.
- How are we going to fix that (using just what we know so far)?

wc, more

```
segfault:Bert ljdursi$ wc all-results
     423     1124     6916 all-results

segfault:Bert ljdursi$ wc au* > all-wcs

segfault:Bert ljdursi$ more all-wcs
```

head, tail

```
segfault:Bert ljdursi$ head all-wcs
???

segfault:Bert ljdursi$ tail all-wcs
???
```

`head, tail` prints start, end of file

- Useful options to head/tail:
  - `-n [number]` : only first/last n lines. (default = 10)

Pipeline of commands

- This idea of chaining commands together - the output from one becomes the input of another - is part of what makes the shell (and programming generally) so powerful.

Pipeline of commands

- So far we've done

```
segfault:Bert ljdursi$ wc au* > all-wcs

segfault:Bert ljdursi$ more all-wcs
```

- Creates a temporary file we don't really care about; we just want to page through all the wc results.

Pipeline of commands

- Interesting (honest,you'll see) fact - like cat, if more isn't given a filename, it also reads from input:
- So this would also work:

```
segfault:Bert ljdursi$ wc au* > all-wcs

segfault:Bert ljdursi$ more < all-wcs
```

SciNet

Pipeline of commands

```
segfault:Bert ljdursi$ wc au* > all-wcs

segfault:Bert ljdursi$ more < all-wcs
```

- This combination of actions - output of one command goes straight into another - so common/useful that shell has special facilities for this:

```
segfault:Bert ljdursi$ wc au* | more
```

SciNet

Pipeline of commands

- Allows you to chain together small pieces into a very powerful analysis pipeline.
- Let's look at another example:

SciNet

`sort` sorts lines in a file

- Let's create a short file and have `sort` sort it.
- Can write file in editor, but let's use our new cat-and-redirection skills:

```
segfault:Bert ljdursi$ cat > toBeSorted
Ernie
Bert
Oscar
Big Bird
^D
segfault: Bert ljdursi$
```

SciNet

`sort` sorts lines in a file

```
segfault:Bert ljdursi$ cat toBeSorted
Ernie
Bert
Oscar
Big Bird

segfault:Bert ljdursi$ sort toBeSorted
Bert
Big Bird
Ernie
Oscar
```

`sort` sorts lines in a file

- Useful options to sort:
- `-n` :sort in numerical order (not lexicographic; eg,101 < 30 without -n.)
- `-k [number]` :sort by the k'th column.
- `-r` :reverses order (decreasing, not increasing)

SciNet

`sort` the data files by size (in characters)

```
segfault:Bert ljdursi$ sort -n -k 3 all-wcs
...
       9        24       151 audioresult-00535
       9        24       152 audioresult-00286
       9        24       152 audioresult-00353
     423      1124      6916 total

segfault:Bert ljdursi$ sort -n -k 3 -r all-wcs
      ..
       9        24       144 audioresult-00239
       9        23       144 audioresult-00453
       9        24       143 audioresult-00393
       9        24       142 audioresult-00493
```

SciNet

`sort` the data files by size (in characters)

```
segfault:Bert ljdursi$ wc au* | sort -n -k 3
...
       9        24       151 audioresult-00535
       9        24       152 audioresult-00286
       9        24       152 audioresult-00353
     423      1124      6916 total

segfault:Bert ljdursi$ wc au* | sort -n -k 3 | more
??
```

Pop quiz!

Modify this to print only smallest, then only largest,data file.

```
segfault:Bert ljdursi$ wc au* | sort -n -k 3
...
       9        24       151 audioresult-00535
       9        24       152 audioresult-00286
       9        24       152 audioresult-00353
     423      1124      6916 total
```

Our first shell script

- So this is useful enough that we are going to write a script that contains this line.
- Will be a program that prints largest (say) data file in the directory.
- First,clean up:

```
segfault:Bert ljdursi$ rm all-wcs all-results toBeSorted
```

SciNet

Our first shell script

- Create the following file, called“biggest”.
- More complex than toBeSorted: use an editor

```
#!/bin/bash
wc * | sort -n -k 3 | tail -n 2 | head -n 1
```

- Now run it with

```
segfault:Bert ljdursi$ source biggest
```

- what do you get?

Our first shell script

- To make this into a"real" program, we're going to tell the OS that this file is executable.
- **Then the `#!/bin/bash` line will tell the OS to run this program with our shell, bash**

```
segfault:Bert ljdursi$ chmod a+x biggest
segfault:Bert ljdursi$ ./biggest
```

SciNet

Note: History

- The shell keeps track of previous commands (which is why up-arrow works)
- To get all recorded previous commands, use the `history` command
- Or `history 10` for the last 10 commands, etc.
- `history 2 > myscript` creates a text file with the last two commands in it, which could be the first draft of a script, for instance.

Largest range - `grep`

- Largest number of characters in data file - probably not super important for our analysis.
- How about experiment with largest range?
- Data files all have line“Range:[Number]”

```
segfault:Bert ljdursi$ grep Range audioresult-00557
Range: 2
```

- grep outputs lines containing the first input string in all of the files given.

```
segfault:Bert ljdursi$ grep Range *
???
```

`grep -v`:
Excludes pattern

- `grep -v pattern file`

prints every line that *doesn't* contain pattern:

```
segfault:Bert ljdursi$ grep -v Range audioresult-00557
??
```

Pop Quiz

- Modify biggest to print out which experiment has the biggest Range.
- Quick tip - what column needs to be sorted?
- (And do we need the head/tail trick?)

Arguments in bash scripts

- We'd like to use this for each directory,but we don't want one copy in each directory.
- Let's move it up one level in directory,and modify it so it would work on any directory's files

```
segfault:data ljdursi$ more biggestRange
#!/bin/bash
grep Range ${1}/* | sort -n -k 2 | tail -1
```

SciNet

Arguments in bash scripts

- When you run a command in the shell,it's name is put in argument 0 ($0)
- Any other arguments are $1,$2...

```
segfault:data ljdursi$ more biggestRange
#!/bin/bash
grep Range ${1}/* | sort -n -k 2 | tail -1
```

SciNet

Arguments in bash scripts

```
segfault:data ljdursi$ ./biggestRange Bert
Bert/audioresult-00384:Range: 10

segfault:data ljdursi$ ./biggestRange THOMAS
THOMAS/0336:Range: 10
```

For loops in bash

- Bash has for loops much like any programming language does.
- We can use this to run our program on several directories:

For loops in bash

```
segfault:data ljdursi$ for dir in Bert gerdal jamesm
> do
> echo "The biggest range in directory " ${dir} " is:"
>  ./biggestRange ${dir}
> done
The biggest range in directory  Bert  is:
Bert/audioresult-00384:Range: 10
The biggest range in directory  gerdal  is:
gerdal/Data0559:Range: 10
The biggest range in directory  jamesm  is:
jamesm/data_517.txt:Range: 10

segfault:data ljdursi$
```

find

- Wildcards are very powerful:
- From data/data directory, type: `ls */*00*`
- Finds files with‘00’ in name in any subdirectory
- Similarly:`echo */*00*`
- or

```
for i in */*00* ; do echo ${i}; done
```

find

- But can only match if you know the path (how many levels of dirs down)
- And can only match by filename.
- `find` is a tool which lets you find files *anywhere* below a given directory, based on *arbitrary* criteria.

SciNet

`find`: do the following

```
segfault:data ljdursi$ find . -print | more
```

directory to start

What to do to the file

SciNet

`find`:can execute arbitrary commands

```
segfault:data ljdursi$ find . -exec echo {} \; | more
```

directory to start

What to do. {} gets filled in with filename;command ends with \;

`find`:can choose files by type

```
segfault:data ljdursi$ find . -type f -print | more
```

directory to start

Only files (type f) get printed;directories are excluded

SciNet

`find`:can choose files by type,name

```
segfault:data ljdursi$ find . -type f -name "*00*" -print | more
```

Only files with 00 in their names;can chain together conditions

`find`:can choose files by contents

```
find . -type f -exec grep "Volume" {} \; -print | more
```

Only search files

If grep returns true (eg,contains "Volume"),then matches

SciNet

uniq

- The command `uniq` strips out repeated adjacent lines (printing out only locally unique lines) - so `sort | uniq` prints only unique lines.
- `uniq -c` prints the lines **and** a count of how many occured
- So the following prints a histogram of volumes:

SciNet

uniq

- So the following prints a histogram of volumes:

```
find  . -type f -exec grep "Volume" {} \; | sort -n -k 2 | uniq -c
    6 Volume:0
   16 Volume:1
   16 Volume:2
   61 Volume:3
   63 Volume:4
   64 Volume:5
   59 Volume:6
   26 Volume:7
   26 Volume:8
   11 Volume:9
    3 Volume:10
```

Assignment

- Copy all of the data files from data/data/.. to a new directory,'cleaneddata'.
- All data files must end in .txt
- Get rid of the NOTES files.
- It's ok if files end in .txt.txt

Assignment

- Do it manually:that works.
- Try to find a solution which will work next time it needs to be done,too.
- Play with things on the command line..
- Many ways to do this!
- “Bonus points”:put it in a script!

SciNet

`echo`	Prints output
`pwd`	Print current directory
`cd [directory]`	Change directory
`cd`	Change directory to home
`ls`	Directory LiSting
`ls -F`	LiSting with Filetypes
`man [cmd]`	MANual page for [cmd]
`file [filename]`	What is in [filename]?
`more [filename]`	Prints out filename(s) by page
`cat [filename]`	Dumps out filename(s)
`wc [filename]`	Line/word/char count of file
`mv [src] [dest]`	Move file
`cp [src] [dest]`	Copy file
`rm [filename]`	Delete file
`head [filename]`	First lines of file
`tail [filename]`	Last lines of file
`sort [filename]`	Sort lines of file
`mkdir [filename]`	Create directory
`rmdir [filename]`	Remove directory
`grep`	Searches input for text
`for..do..done`	for loops in bash
`find`	Searches for files

SciNet

Using the shell on other computers

- What if the programs,data you want to use are on another system?
- Can use ssh (Secure Shell) to log in,or copy data,from other machines securely (encrypted).
- Easy to use from the shell.

SciNet

Using the shell on other computers

- Widely available:comes with MacOS and Linux
- OnWindows built in to MobaXTerm

Using ssh

- ssh username@remote.host.name
- prompts you for password
- you're now using the shell on that remote machine.

SciNet

Using ssh:X Forwarding

- If you will be using graphical programs on the remote host,can forward X windows over ssh
  - ssh -Y username@remote.host.name or
  - ssh -X username@remote.host.name
  - then Xwindows graphics stuff is tunnelled through the ssh connection - can display graphics,show plots,etc.

SciNet

A note on authentication

- The password prompt is a prompt from the remote host which proves that you are allowed to log in as that account.
- Ssh has another authentication mechanism:"keys". You generate a key on *your* machine;presenting that key proves that you are you on your machine
- You then tell the remote machine to allow logins using that key - no password!
- Good to know,but more than we can talk about now:
- https://wiki.scinethpc.ca/wiki/index.php/Ssh_keys

SciNet

Copying files:scp

- Can copy files over ssh using scp
- Like cp: cp sourcefile destfile
- But includes remote username/host information:
- scp localfile username@remote.host:remotefile
  or
- scp username@remote.host:remotefile localfile
- Be careful with wildcards!
- For copying large numbers of files,look up rsync

SciNet

Example:

- Say I want to copy the all-results data file to my remote machine:

```
segfault$ scp all-results ljdursi@remote.com:Desktop/data
```

- Or copy a script from the remote machine to here:

```
segfault$ scp ljdursi@remote.com:Desktop/data/biggest .
```

SciNet

Pop Quiz

- Let's say I want to copy all my txt files from remote machine to local.
- Why won't this work?

```
segfault$ scp ljdursi@remote.com:data/*.dat .
```

SciNet

Makefiles

- Shell is great for automation
- Write scripts that are a list of commands to execute
- Do this,do that,then do those.

Printing out a histogram

- Look in to ~/Desktop/data/make
- One set of data (from jamesm)
- Task:generate a histogram of Volumes (as before,using sort/uniq -c) in a file then run a program histdata.dat
- Then run a program written in C, texthistogram,to produce anASCII-art histogram,histogram.txt

SciNet

Histogram

```
segfault$ grep -h Volume data*.txt | sort -n -k 2 \
          | uniq -c > histdata.dat
segfault$ gcc -o texthistogram texthistogram.c -Wall -std=c99
segfault$ ./texthistogram < histdata.dat > histogram.txt
segfault$ cat histogram.txt
    0 ##
    1 ##
    2 ###
    3 ###
    4 ######
    5 #########
    6 #######
    7 ####
    8 ##
    9 ##
```

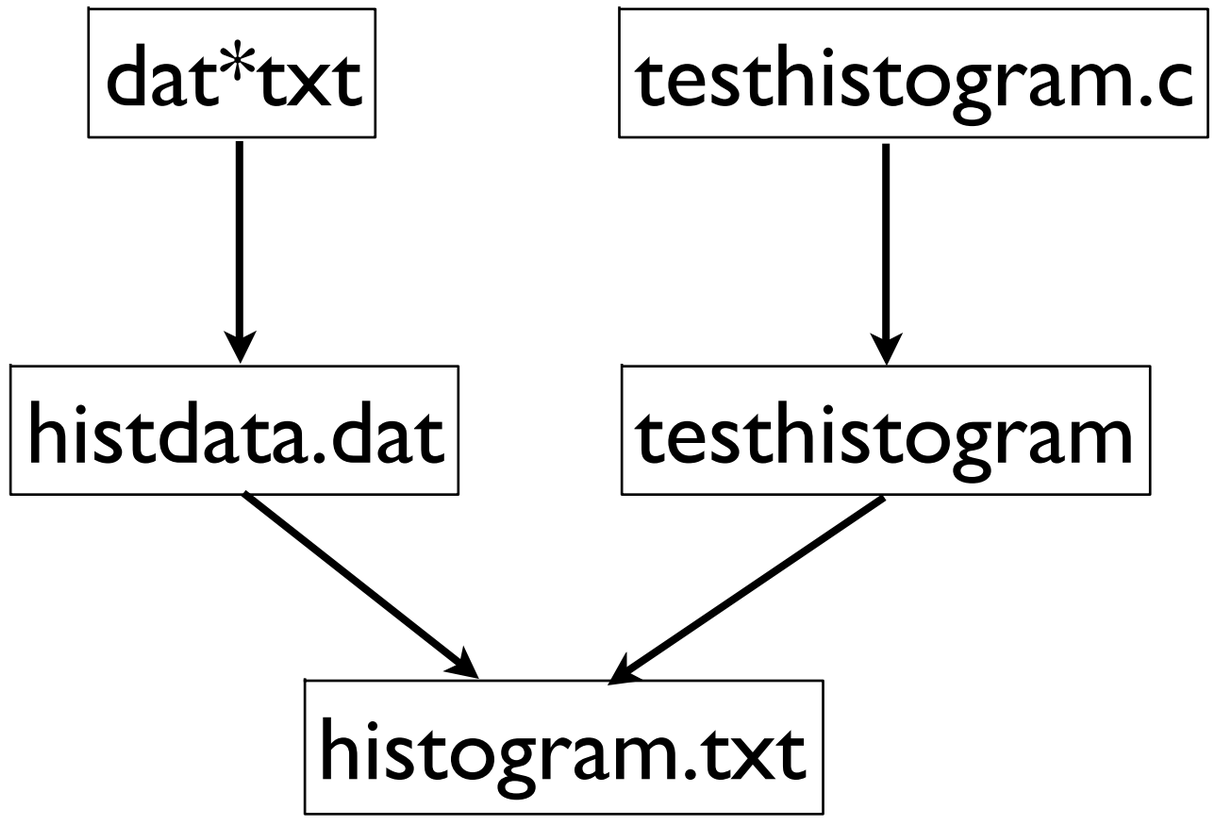
SciNet

Printing out a histogram

- If something changes:
- May have to regenerate histdata.txt from data files
- May have to recompile testhistogram from testhistogram.c
- Probably have to rerun testhistogram.
- How do we know which ones we need to do?

Dependencies

- This comes up all the time;a tree of dependancies to one or more final outputs



Makefile

- A Makefile contains **targets**
- along with a list of the **dependencies** each requires
- followed by a series of shell **commands** necessary to build the target from its dependencies

SciNet

Makefile

```
datafiles = data_*txt

histogram.txt: histdata.dat texthistogram
	./texthistogram < histdata.dat > histogram.txt

histdata.dat: $(datafiles)
	grep -h Volume $^ | sort -n -k 2 | uniq -c > $@

texthistogram: texthistogram.c
	gcc -o texthistogram texthistogram.c -Wall -std=c99

clean:
	rm -f texthistogram histdata.dat histogram.txt
```

SciNet

Makefile

```
datafiles = data_*txt

histogram.txt: histdata.dat texthistogram
	./texthistogram < histdata.dat > histogram.txt

histdata.dat: $(datafiles)
	grep -h Volume $^ | sort -n -k 2 | uniq -c > $@

texthistogram: texthistogram.c
	gcc -o texthistogram texthistogram.c -Wall -std=c99

clean:
	rm -f texthistogram histdata.dat histogram.txt
```

Common pitfall - this has to be a tab.

SciNet

Makefile

```
segfault$ make

grep -h Volume data_217.txt data_219.txt data_226.txt
data_228.txt data_231.txt data_236.txt data_255.txt
data_261.txt data_264.txt data_266.txt data_280.txt
data_282.txt data_290.txt data_293.txt data_312.txt
data_325.txt data_326.txt data_343.txt data_360.txt
data_368.txt data_374.txt data_375.txt data_383.txt
data_394.txt data_395.txt data_401.txt data_418.txt
data_457.txt data_474.txt data_475.txt data_476.txt
data_478.txt data_480.txt data_489.txt data_496.txt
data_509.txt data_510.txt data_517.txt data_524.txt
data_553.txt | sort -n -k 2 | uniq -c > histdata.dat
gcc -o texthistogram texthistogram.c -Wall -std=c99
./texthistogram < histdata.dat > histogram.txt
setfault$
```

Makefile

- `make [target]`
- The make program looks for a Makfile and reads it in
- It then starts to build the target specified (or the first target if none are specified)

Makefile

```
datafiles = data_*txt

histogram.txt: histdata.dat texthistogram
	./texthistogram < histdata.dat > histogram.txt

histdata.dat: $(datafiles)
	grep -h Volume $^ | sort -n -k 2 | uniq -c > $@

texthistogram: texthistogram.c
	gcc -o texthistogram texthistogram.c -Wall -std=c99

clean:
	rm -f texthistogram histdata.dat histogram.txt
```

SciNet

Makefile

```
datafiles = data_*txt

histogram.txt: histdata.dat texthistogram
	./texthistogram < histdata.dat > histogram.txt

histdata.dat: $(datafiles)
	grep -h Volume $^ | sort -n -k 2 | uniq -c > $@

texthistogram: texthistogram.c
	gcc -o texthistogram texthistogram.c -Wall -std=c99

clean:
	rm -f texthistogram histdata.dat histogram.txt
```

Common pitfall:this *has* to be a tab,not spaces.

SciNet

Makefile

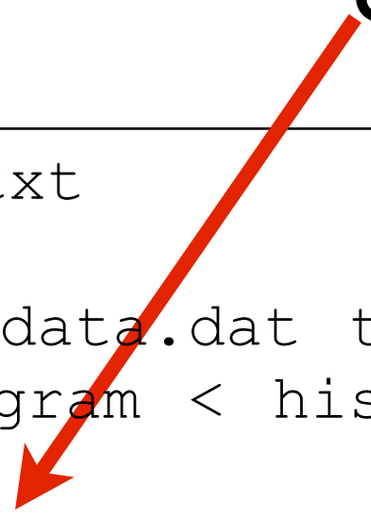
Variable declaration

```
datafiles = data_*txt

histogram.txt: histdata.dat texthistogram
	./texthistogram < histdata.dat > histogram.txt

histdata.dat: $(datafiles)
	grep -h Volume $^ | sort -n -k 2 | uniq -c > $@

texthistogram: texthistogram.c
	gcc -o texthistogram texthistogram.c -Wall -std=c99

clean:
	rm -f texthistogram histdata.dat histogram.txt
```

SciNet

Makefile

Using the variable



```
datafiles = data_*txt

histogram.txt: histdata.dat texthistogram
	./texthistogram < histdata.dat > histogram.txt

histdata.dat: $(datafiles)
	grep -h Volume $^ | sort -n -k 2 | uniq -c > $@

texthistogram: texthistogram.c
	gcc -o texthistogram texthistogram.c -Wall -std=c99

clean:
	rm -f texthistogram histdata.dat histogram.txt
```

SciNet

Makefile

target:dependancies

```
datafiles = data_*txt

histogram.txt: histdata.dat texthistogram
	./texthistogram < histdata.dat > histogram.txt

histdata.dat: $(datafiles)
	grep -h Volume $^ | sort -n -k 2 | uniq -c > $@

texthistogram: texthistogram.c
	gcc -o texthistogram texthistogram.c -Wall -std=c99

clean:
	rm -f texthistogram histdata.dat histogram.txt
```

SciNet

Makefile

list of commands to build target from dependencies

```
datafiles = data_*txt

histogram.txt: histdata.dat texthistogram
	./texthistogram < histdata.dat > histogram.txt

histdata.dat: $(datafiles)
	grep -h Volume $^ | sort -n -k 2 | uniq -c > $@

texthistogram: texthistogram.c
	gcc -o texthistogram texthistogram.c -Wall -std=c99

clean:
	rm -f texthistogram histdata.dat histogram.txt
```

SciNet

Makefile

But what happens if this dependency doesn't exist yet?

```
datafiles = data_*txt

histogram.txt: histdata.dat texthistogram
	./texthistogram < histdata.dat > histogram.txt

histdata.dat: $(datafiles)
	grep -h Volume $^ | sort -n -k 2 | uniq -c > $@

texthistogram: texthistogram.c
	gcc -o texthistogram texthistogram.c -Wall -std=c99

clean:
	rm -f texthistogram histdata.dat histogram.txt
```

SciNet

Makefile

Build it!

```
datafiles = data_*txt

histogram.txt: histdata.dat texthistogram
	./texthistogram < histdata.dat > histogram.txt

histdata.dat: $(datafiles)
	grep -h Volume $^ | sort -n -k 2 | uniq -c > $@

texthistogram: texthistogram.c
	gcc -o texthistogram texthistogram.c -Wall -std=c99

clean:
	rm -f texthistogram histdata.dat histogram.txt
```

SciNet

Makefile

??

```
datafiles = data_*txt

histogram.txt: histdata.dat texthistogram
	./texthistogram < histdata.dat > histogram.txt

histdata.dat: $(datafiles)
	grep -h Volume $^ | sort -n -k 2 | uniq -c > $@

texthistogram: texthistogram.c
	gcc -o texthistogram texthistogram.c -Wall -std=c99

clean:
	rm -f texthistogram histdata.dat histogram.txt
```

SciNet

SpecialVariables

- `$^` - all dependencies
- `$@` - target name
- Lots of others built in: `CC`, `CFLAGS` - the C compiler to use and flags to use when compiling; many others.

SciNet

Implicit rules

- There are actually lots of rules built-in to make for common tasks
- Build an executable from a .c file

SciNet

Resources

- SciNetWiki:
  - http://wiki.SciNetHPC.ca
- Software Carpentry
  - http://software-carpentry.org/4_0/shell/
  - http://software-carpentry.org/4_0/make/