UNIVERSITY OF OREGON

CIS 330:

```
   __  __      _                         __   ______    ________    
  / / / /___  (_)  __   ____ _____  ____/ /  / ____/  _/_/ ____/__    __ 
 / / / / __ \/ / |/_/  / __ `/ __ \/ __  /  / /     _/_// /  __/ /___/ /_
/ /_/ / / / / />  <   / /_/ / / / / /_/ /  / /___ _/_/ / /__/_  __/_  __/
\____/_/ /_/_/_/|_|   \__,_/_/ /_/\__,_/   \____//_/   \____//_/   /_/   
```

Lecture 1:
Course Overview & Introduction to Unix

April 2, 2018

Hank Childs, University of Oregon

Enrollment

- Who’s in the class?
- Who’s not?

Outline

- Class Overview
  - My Background
  - Goals
  - Syllabus
- Getting Started With Unix
  - Unix History
  - Shell Prompts
  - Files
  - File Editors
- Project 1

Outline

- Class Overview
  - My Background
  - Goals
  - Syllabus
- Getting Started With Unix
  - Unix History
  - Shell Prompts
  - Files
  - File Editors
- Project 1

Hi, I’m Hank…

- Associate Professor of CIS at UO
- Arrived at UO March 2013
- Research interests include visualization, high-performance computing, computer graphics
- Director of CDUX Research Group (9 PhD, 3 MS, 0 BS)

My Background

- Previously:
  - Lawrence Livermore, 1999-2009
  - Lawrence Berkeley, 2009-2016
  - UC Davis 2009-2013
- Education:
  - B.S. (CS/Math), 1994-1999
  - Ph.D., 2000-2006

I have spent 15 years programming,
almost exclusively using C, C++, and Unix

VisIt: Application for Visualizing and Analyzing of Very Large Data

- Open source tool
- 1.5M lines of C++ code
  - also Python & Java bindings
- Downloaded >200K times, run on many different UNIX environments

Because of these experiences, I am pretty conservative in what features I make use of.

Simulation of Rayleigh-Taylor Instability, run on LLNL BG/L supercomputer with MIRANDA code, visualized with VisIt

I also served as the CSWA of multiple teams, leading SW integration efforts

Outline

- Class Overview
  - My Background
  - Goals
  - Syllabus
- Getting Started With Unix
  - Unix History
  - Shell Prompts
  - Files
  - File Editors
- Project 1

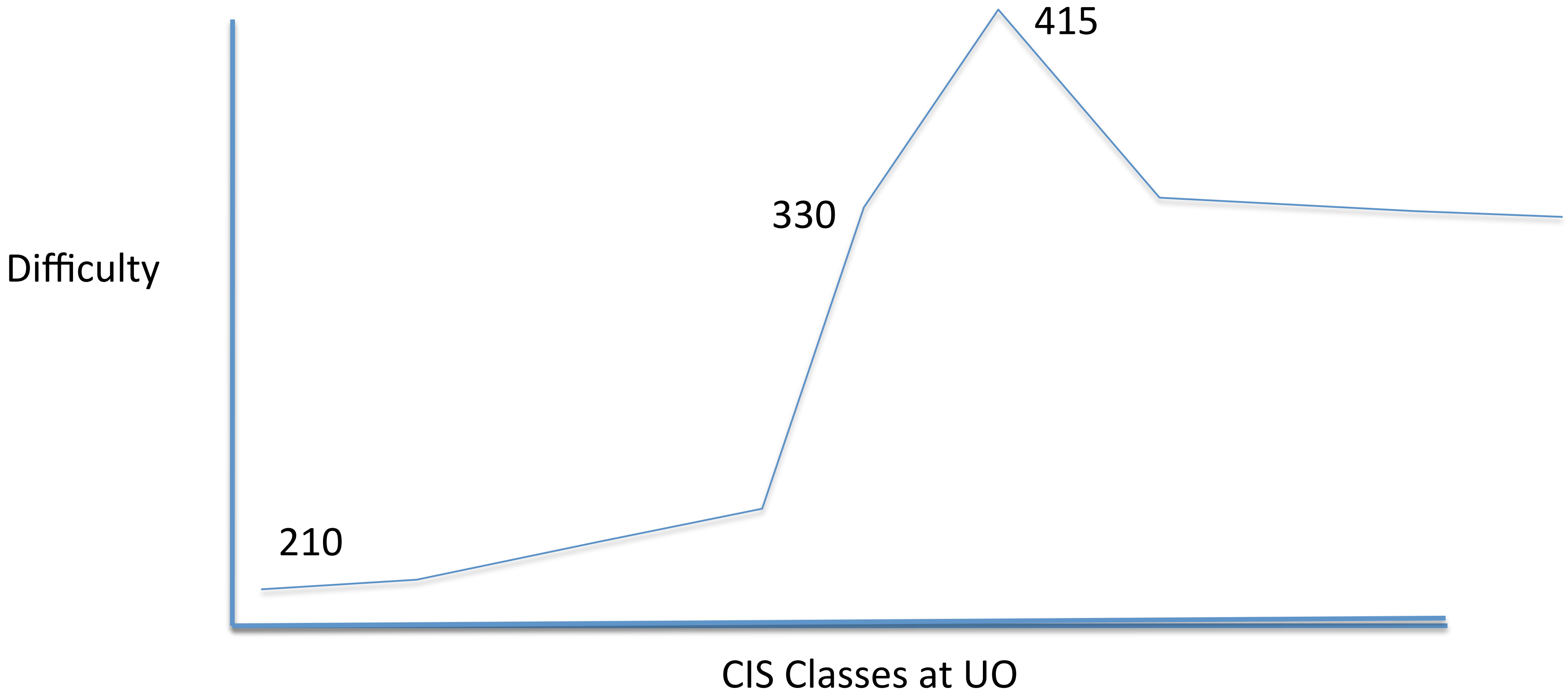
CIS 330 Goals

- Goals: excellence in C, C++, and Unix
- Why?
  - Many of our 400-level classes require strong knowledge in C, C++, and Unix
  - Critical for success with many development jobs
    - Development jobs are good jobs!!!
- Programming Languages Beacon: http://www.lextrait.com/vincent/implementations.html

How Will We Develop Excellence in C, C++, and Unix?

- Answer: many, many projects
- Very hard to learn this material from lecture
  - Really need to get your hands dirty and “do it”

Goal: Getting Ready for 415

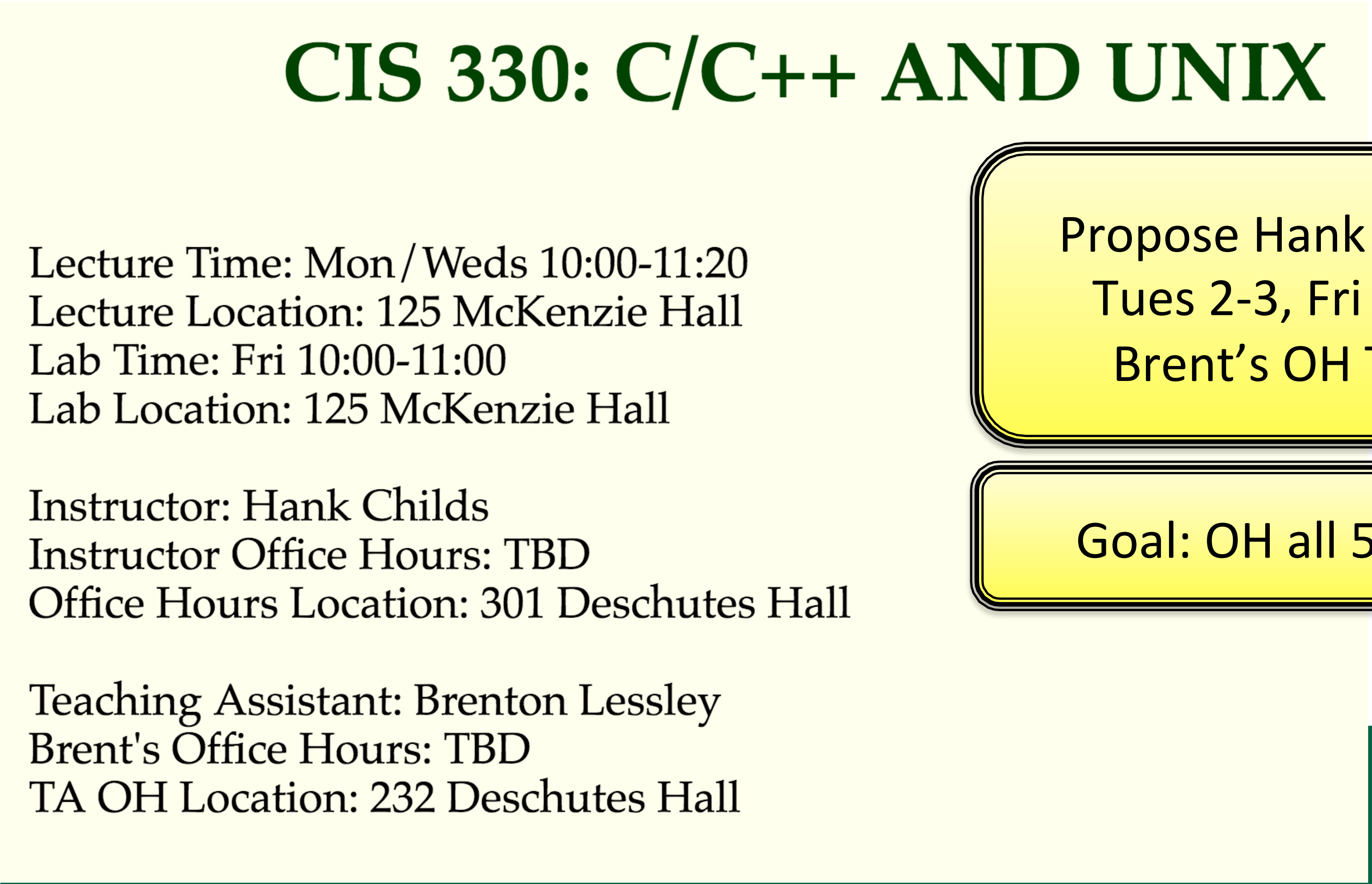


Outline

- Class Overview
  - My Background
  - Goals
  - Syllabus
- Getting Started With Unix
  - Unix History
  - Shell Prompts
  - Files
  - File Editors
- Project 1

Syllabus is Online

- http://ix.cs.uoregon.edu/~hank/330



CIS 330: C/C++ AND UNIX

Lecture Time: Mon/Weds 10:00-11:20
Lecture Location: 125 McKenzie Hall
Lab Time: Fri 10:00-11:00
Lab Location: 125 McKenzie Hall

Instructor: Hank Childs
Instructor Office Hours: TBD
Office Hours Location: 301 Deschutes Hall

Teaching Assistant: Brenton Lessley
Brent's Office Hours: TBD
TA OH Location: 232 Deschutes Hall

Propose Hank OH as
Tues 2-3, Fri 12-1
Brent’s OH TBD

Goal: OH all 5 days

Grading For This Course

- Final: 30%
- Projects: 70% (made up of multiple pieces)
  - Project 1: Learning Unix (5%)
    - 1A (1%), 1B (2%), 1C (2%)
  - Project 2: C/C++ Primer (15%)
  - Project 3: Large System + Object Oriented Programming
  - Project 4: Debugging/Profiling (1
- Extra Credit through:
  - Community participation

These percentages *might* be adapted as the quarter goes on. I will notify you all via email and via class lecture if this happens.

Prerequisites For This Course

- CIS 314 is the only prerequisite

Introductory Projects

Project	Assigned	Due	Task	% Grade
1A	4/2	4/4	Editing Files	1%
1B	4/6	4/11	Scripts/ Permissions	2%
1C	4/9	4/13	Build	2%
2A	4/4	4/9	Memory	3%
2B	4/13	4/18	Unions + Function Pointers	5%
…	…	…	…	…

(This is notional …
here to give you an idea)

This is scary…

- From my end:
  - Huge amount of material to cover…
- From your end:
  - Being asked to learn so much…

- I really want to put you in a winning position
  - Let me know if you think I'm not
- Take this seriously
  - You've probably never had a class like this before

Expectations

- I ask you put a lot of time into this course, and I believe the payoff will be significant for each of you.
- The grading is designed to make sure you are keeping up with the assignments.
  - Staying on top of the projects will be critical to succeeding in this class.

Expectations

- The projects in this class will be hard work.
- It is difficult to quote exactly how much time, since there is variation in background and programming skill.
  - I expect those who have less developed programming skills will find this class to be a considerable effort, but also that they will have significant improvement by the end of the course

This Class In a Nutshell…

- Learn the basics of Unix so that we can be functional
  - More Unix introduced as needed throughout course
- Learn more C, and gently ease into C++
- Do a large C++-based project that embraces object-oriented programming
- Learn basics of development: debugging and profiling

Most of the learning will happen with projects. The lectures are designed to help you do the projects.

Who Is This Class For?

- I am assuming that you have never done Unix or C++ and only got minimal C in 314
  - The class is designed for this profile.
- If you do know a lot of Unix/C/C++, then please try to avoid asking super advanced questions during lecture.
  - Rather, try asking and accept that I may say “too advanced” in response.

Norms for this class

- Please ask questions
- Please ask me to slow down
- Please give feedback
- Quiet classroom greatly valued

Do You Need To Attend Class?

- Short answer: it would be dumb to miss class
  - Unless you are really, really good. Then: your call.
- It is a really bad situation if you miss class and then want me or the TA's to "catch you up"
  - we won't do it
    - unless you had a good reason for missing (in which case, of course you will be accommodated)
- 99% of the material will come via lecture

20% of S16 class failed (~10% in S13, S14, S15)

- How to fail
  - Get behind on projects
  - Stop attending class
    - Every class is basically on how to do the HW
- How to succeed
  - Come to class
  - Get every project completed on time
  - Have a network

25% of S17 class failed

- How did it get worse?
  - (stay tuned)

My mental model about getting behind in this class

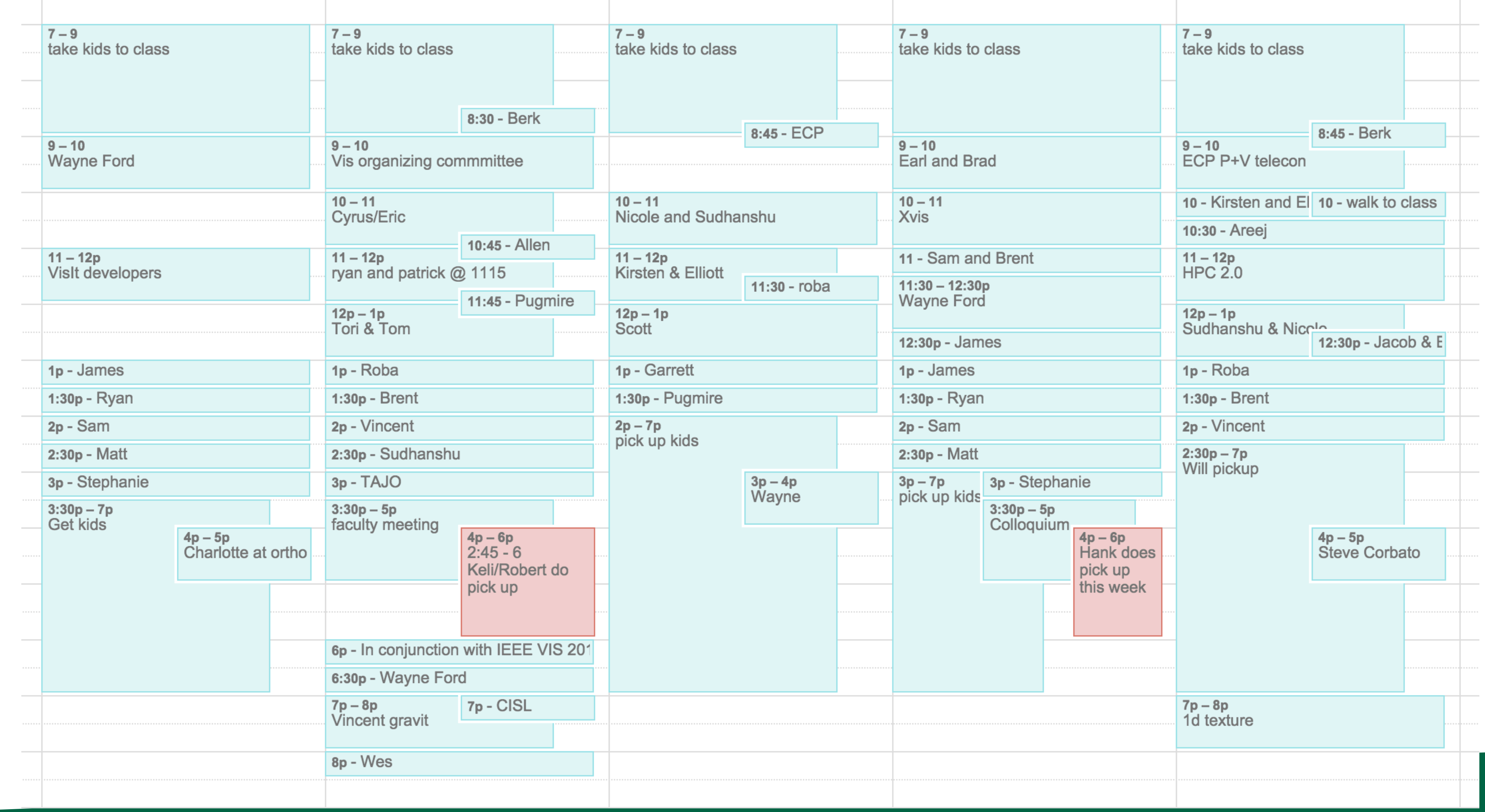
WASHINGTON
UNITED STATES OF AMERICA
Whitman Mission
Oregon City
The Dalles
IDAHO
WYOMING
OREGON
Fort Boise
Fort Hall
Soda Springs
Independence Rock
NEBRASKA
Courthouse Rock
Fort Laramie
Fort Bridger
Chimney Rock
MO
Independence
KANSAS
🟥 = Route of the Oregon Trail

If you get too far behind, then it will not be possible to catch up.

When does this class get hard?

- If you have never done Unix, it might feel hard right away
- If you have done Unix/C, it might feel a bit easy at the beginning, but don’t be fooled.
  - Project 3 is challenging for just about everyone

Norms for interacting with me



7 – 9
take kids to class

9 – 10
Wayne Ford

11 – 12p
VisIt developers

1p - James

1:30p - Ryan

2p - Sam

2:30p - Matt

3p - Stephanie

3:30p – 7p
Get kids

4p – 5p
Charlotte at ortho

7 – 9
take kids to class

8:30 - Berk

9 – 10
Vis organizing commmittee

10 – 11
Cyrus/Eric

10:45 - Allen

11 – 12p
ryan and patrick @ 1115

11:45 - Pugmire

12p – 1p
Tori & Tom

1p - Roba

1:30p - Brent

2p - Vincent

2:30p - Sudhanshu

3p - TAJO

3:30p – 5p
faculty meeting

4p – 6p
2:45 - 6
Keli/Robert do pick up

6p - In conjunction with IEEE VIS 201

6:30p - Wayne Ford

7p – 8p
Vincent gravit

7p - CISL

8p - Wes

7 – 9
take kids to class

8:45 - ECP

10 – 11
Nicole and Sudhanshu

11 – 12p
Kirsten & Elliott

11:30 - roba

12p – 1p
Scott

1p - Garrett

1:30p - Pugmire

2p – 7p
pick up kids

3p – 4p
Wayne

7 – 9
take kids to class

9 – 10
Earl and Brad

10 – 11
Xvis

11 - Sam and Brent

11:30 – 12:30p
Wayne Ford

12:30p - James

1p - James

1:30p - Ryan

2p - Sam

2:30p - Matt

3p – 7p
pick up kids

3p - Stephanie

3:30p – 5p
Colloquium

4p – 6p
Hank does pick up this week

7 – 9
take kids to class

8:45 - Berk

9 – 10
ECP P+V telecon

10 - Kirsten and El

10 - walk to class

10:30 - Areej

11 – 12p
HPC 2.0

12p – 1p
Sudhanshu & Nic

12:30p - Jacob & E

1p - Roba

1:30p - Brent

2p - Vincent

2:30p – 7p
Will pickup

4p – 5p
Steve Corbato

7p – 8p
1d texture

Course Materials

- PowerPoint lectures will be posted online.
- I will “live code” frequently.
- Textbook:
  - – Past terms: none
  - – This term: incorporating “C and Data Structures” by Sventek
    - On Canvas (legal statement next slide)

C and Data Structures - a well-structured approach

Joseph S. Sventek

Academic Misconduct (1 of 2)

- The programming projects are individual efforts
  - You may discuss the projects with your classmates.
  - Do not let someone look at your code on your screen. (BUT: helper can look at helpee’s code)
  - Absolutely, positively do not email code.
  - Do not search the internet for previous implementations (includes github)

Academic Misconduct (2 of 2)

- If I detect collusion, all individuals involved will receive an F in the course immediately
  - I choose to not enumerate cases that involve collusion. Whiteboard conversations are fine. If appropriate, the helper can look at the helpee's code. If you feel you are in a gray area, then you should email me.
  - Please note that if you are the one providing too much help, then you will also get an F

Late Passes

- You have 2 "late passes."
- Late passes allow you to turn in your project two days after the due date for full credit.
- If you run out of late passes, then you may continue to earn half credit on any project.
- Every unused late pass is worth 0% extra credit.
- Don’t need to tell me if you want to use it
  - Late pass assignment will be a question on the final
  - All late projects will be scored at 50% and then credit from late passes will be awarded at the end of the term

Class Summary

- This class will teach you about Unix, C, and C++
- This class will improve your programming skills
- This class will likely help you succeed at a job
- This class will require a lot of work
- The projects are the heart of the class
  - The lectures are designed to help you do the projects

IDEs

- IDEs are great
  - … but in this class, we will learn how to get by without them
- Many, many Unix-based projects don’t use IDEs
  - The skills you are using will be useful going forward in your careers

Accessing a Unix environment

- Rm 100, Deschutes
- Remote logins (ssh, scp)
- Windows options
  - Cygwin / MSYS
  - Virtual machines

Who has home access to a Unix environment?

Who has Windows?

Outline

- Class Overview
  - My Background
  - Goals
  - Syllabus
- Getting Started With Unix
  - Unix History
  - Shells
  - Files
  - File Editors
- Project 1

Reading

- C and Data Structures:
  - Chapter 2.1, 2.2, 2.3, 2.4., 2.5, and 2.6.1.

Outline

- Class Overview
  - My Background
  - Goals
  - Syllabus
- Getting Started With Unix
  - Unix History
  - Shells
  - Files
  - File Editors
- Project 1

What is Unix?

- Operating system
  - Multi-tasking
  - Multi-user
- Started at AT&T Bell Labs in late ’60s, early ‘70s
- First release in 1973

What is Unix?

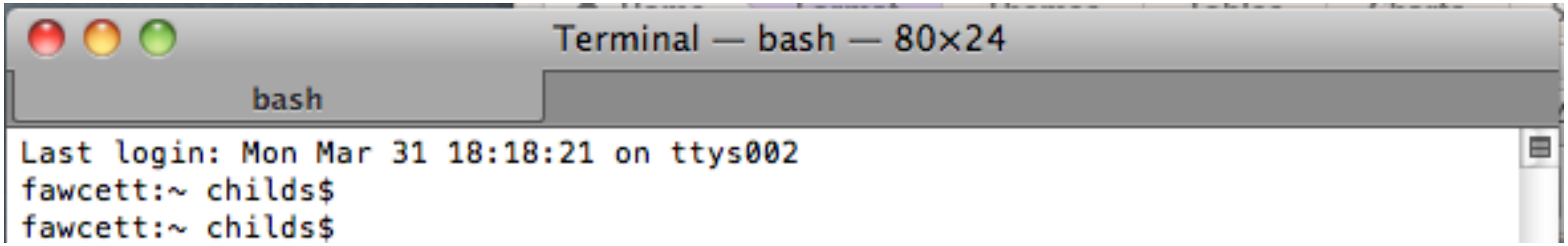
- 80s & 90s: many competing versions, all conforming to same standard
  - AIX (IBM), Solaris (Sun), HP-UX (Hewlett-Packard)
- 1990s: Linux takes off
  - Open source
- 2000s: commercial Unixes abandoned, companies use Linux, back Linux
  - Several variants of Linux
- OS X: used on Macs since 2002
  - Meets Unix standard

Outline

- Class Overview
  - My Background
  - Goals
  - Syllabus
- Getting Started With Unix
  - Unix History
  - Shells
  - Files
  - File Editors
- Project 1

Shells

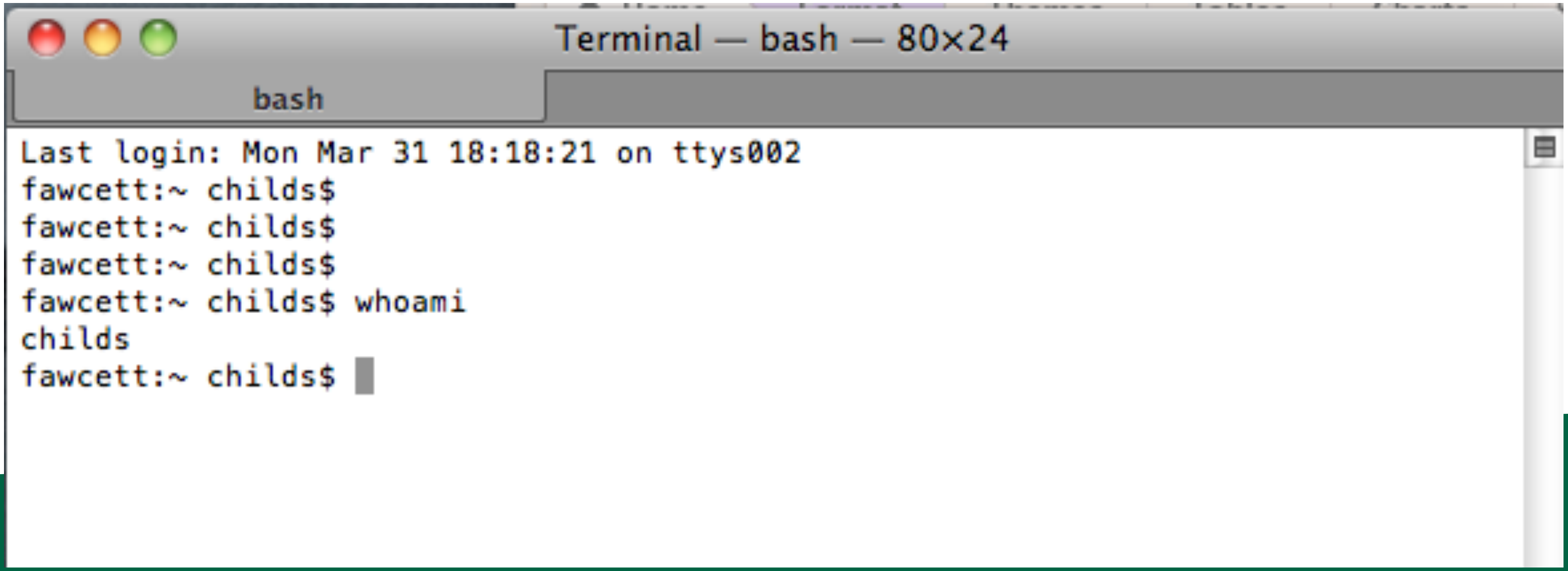
- Shells are accessed through a terminal program
  - – Typically exposed on all Linux
  - – Mac: Applications->Utilities->Terminal
    - (I always post this on the dock immediately upon getting a new Mac)



```
Last login: Mon Mar 31 18:18:21 on ttys002
fawcett:~ childs$
fawcett:~ childs$
```

Shells

- Shells are interpreters
  - Like Python
- You type a command, it carries out the command



```
Last login: Mon Mar 31 18:18:21 on ttys002
fawcett:~ childs$
fawcett:~ childs$
fawcett:~ childs$
fawcett:~ childs$ whoami
childs
fawcett:~ childs$ 
```

Shells

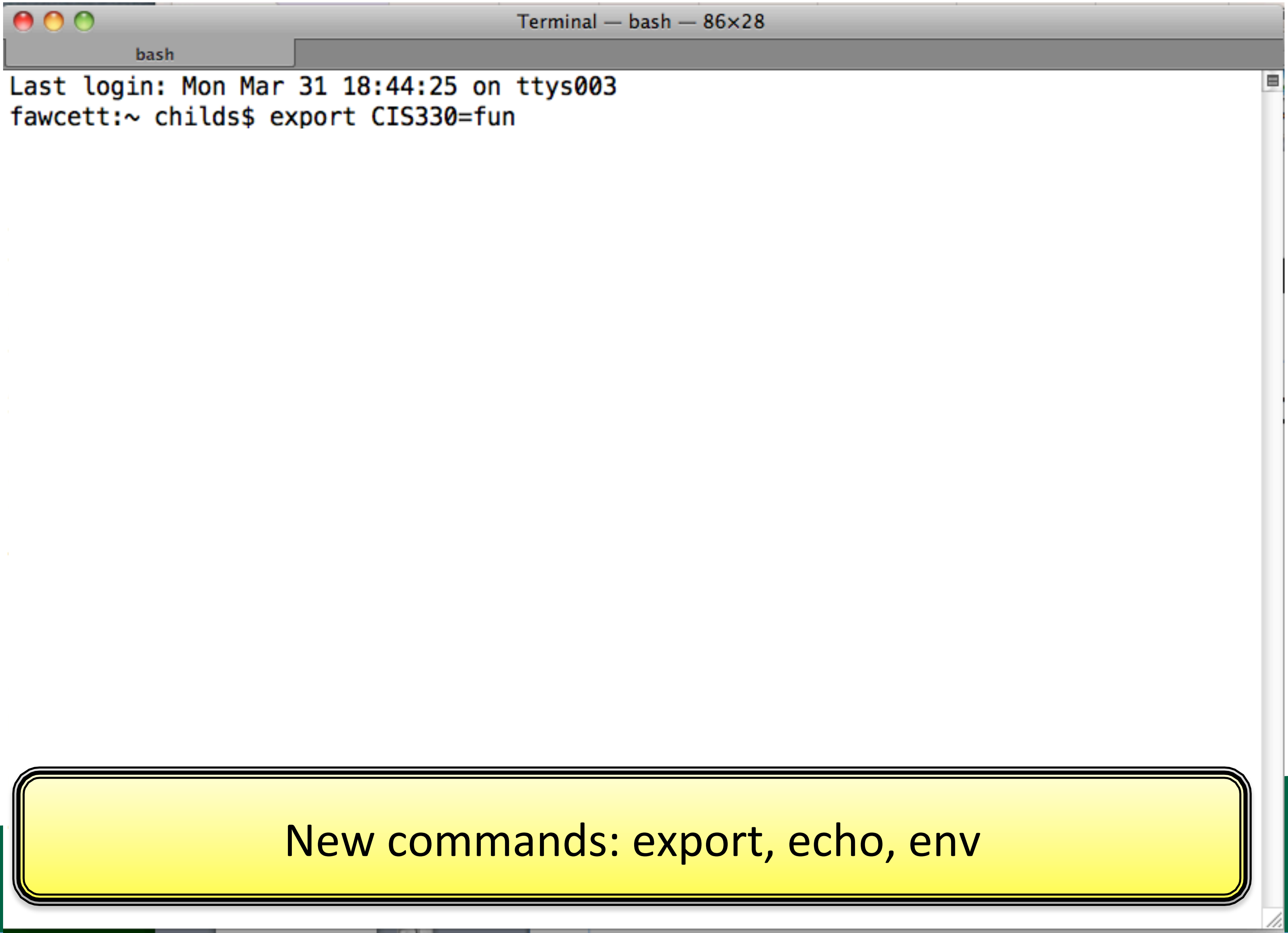
- There are many types of shells
- Two most popular:
  - sh (= bash & ksh)
  - csh (= tcsh)
- They differ in syntax, particularly for
  - Environment variables
  - Iteration / loops / conditionals

The examples in this course will use syntax for sh

Environment Variables

- Environment variables: variables stored by shell interpreter
- Some environment variables create side effects in the shell
- Other environment variables can be just for your own private purposes

Environment Variables



```
Last login: Mon Mar 31 18:44:25 on ttys003
fawcett:~ childs$ export CIS330=fun
```

New commands: export, echo, env

Shells

- There is lots more to shells … we will learn about them as we go through the quarter

Outline

- Class Overview
  - My Background
  - Goals
  - Syllabus
- Getting Started With Unix
  - Unix History
  - Shells
  - Files
  - File Editors
- Project 1

Files

- Unix maintains a file system
  - File system controls how data is stored and retrieved
- Primary abstractions:
  - Directories
  - Files
- Files are contained within directories

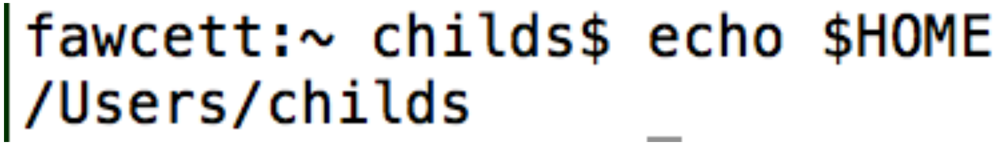
Directories are hierarchical

- Directories can be placed within other directories
- "/" -- The root directory
  - Note "/", where Windows uses "\"
- "/dir1/dir2/file1"
  - What does this mean?

File file1 is contained in directory dir2,
which is contained in directory dir1,
which is in the root directory

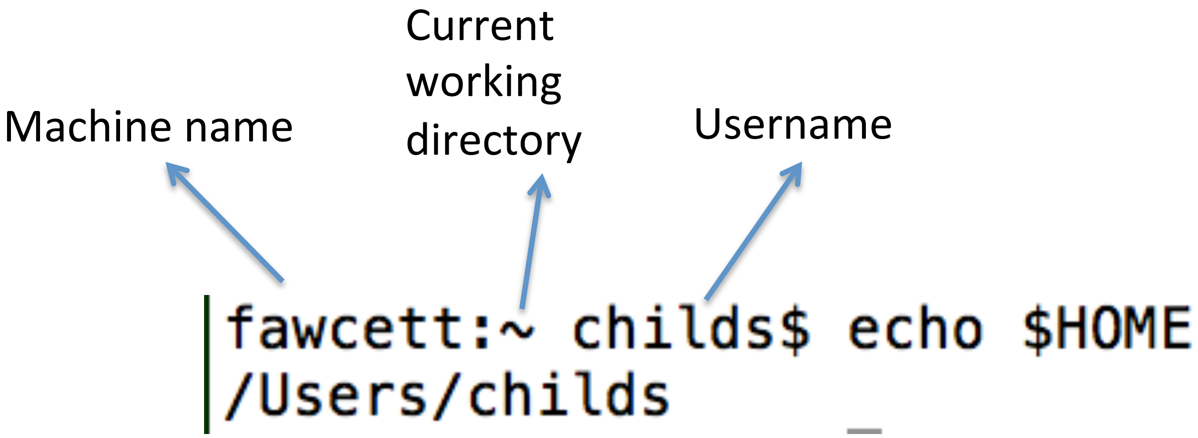
Home directory

- Unix supports multiple users
- Each user has their own directory that they control
- Location varies over Unix implementation, but typically something like "/home/username"
- Stored in environment variables



```
fawcett:~ childs$ echo $HOME
/Users/childs
```

Anatomy of shell formatting



Machine name

Current working directory

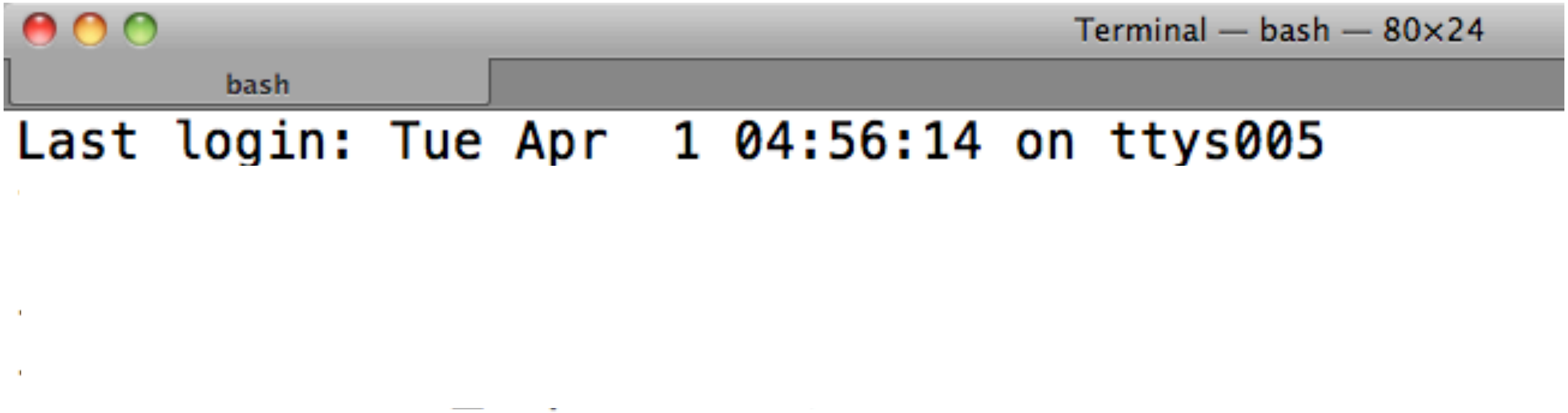
Username

```
fawcett:~ childs$ echo $HOME
/Users/childs
```

- “~” (tilde) is shorthand for your home directory
  - You can use it when invoking commands

The shell formatting varies over Unix implementation and can be customized with environment variables. (PS1, PS2, etc)

File manipulation



```
Last login: Tue Apr  1 04:56:14 on ttys005
```

New commands: mkdir, cd, touch, ls, rmdir, rm

cd: change directory

- The shell always has a “present working directory”
  - directory that commands are relative to
- “cd” changes the present working directory
- When you start a shell, the shell is in your “home” directory

Unix commands: mkdir

- mkdir: makes a directory
  - Two flavors
    - Relative to current directory
      - mkdir dirNew
    - Relative to absolute path
      - mkdir /dir1/dir2/dirNew
        - (dir1 and dir2 already exist)

Unix commands: rmdir

- rmdir: removes a directory
  - Two flavors
    - Relative to current directory
      - rmdir badDir
    - Relative to absolute path
      - rmdir /dir1/dir2/badDir
        - » Removes badDir, leaves dir1, dir2 in place
- Only works on empty directories!
  - "Empty" directories are directories with no files

Most Unix commands can distinguish between absolute and relative path, via the "/" at beginning of filename.
(I'm not going to point this feature out for subsequent commands.)

Unix commands: touch

- touch: “touch” a file
- Behavior:
  - If the file doesn’t exist
    - → create it
  - If the file does exist
    - → update time stamp

Time stamps record the last modification to a file or directory

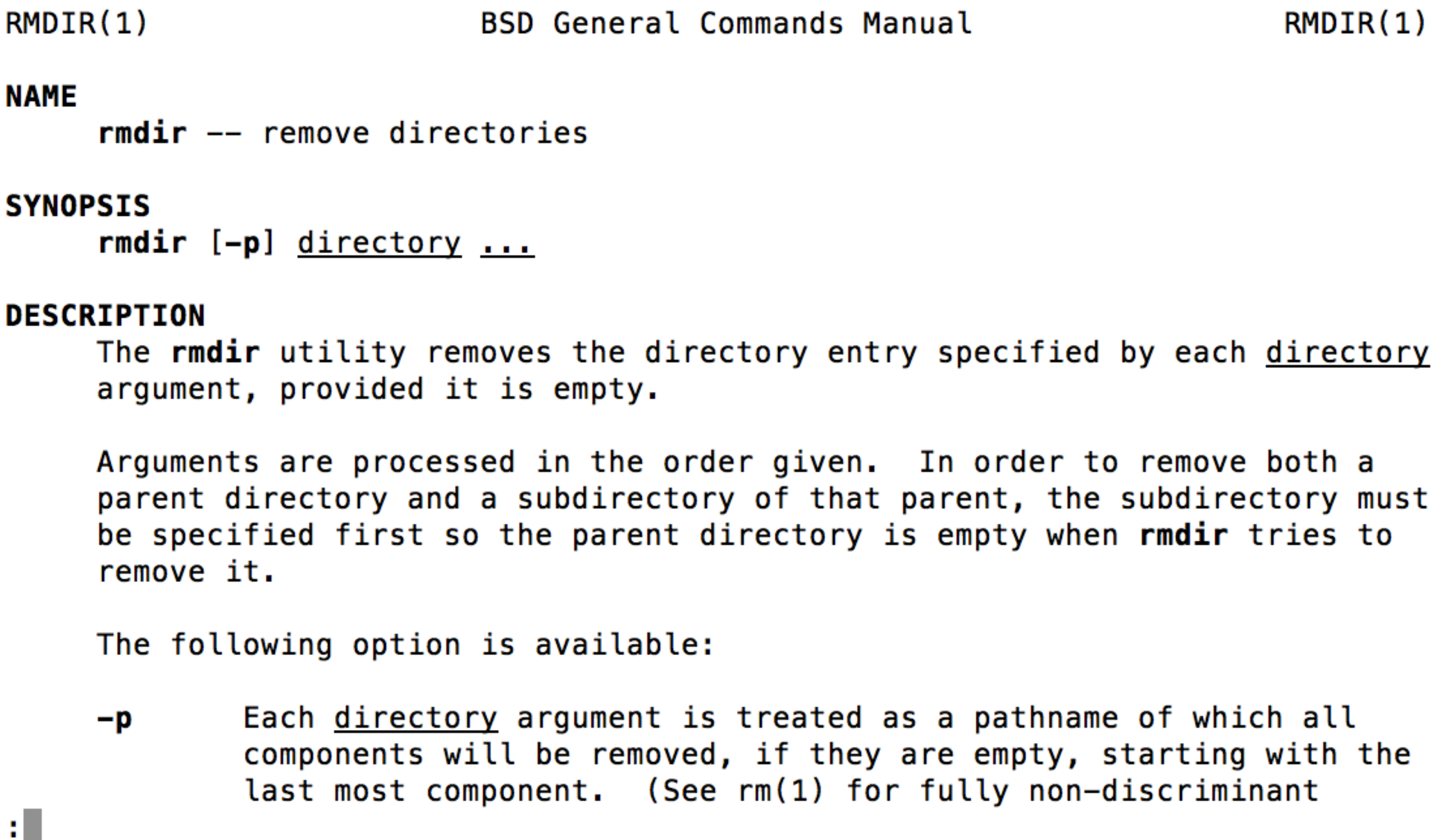
Why could time stamps be useful?

Unix commands: ls

- ls: list the contents of a directory
  - Note this is “LS”, not “is” with a capital ‘i’
- Many flags, which we will discuss later
  - A flag is a mechanism for modifying a Unix programs behavior.
  - Convention of using hyphens to signify special status
- “ls” is also useful with “wild cards”, which we will also discuss later

Important: “man”

- Get a man page:
- → “man rmdir” gives:



```
RMDIR(1)                  BSD General Commands Manual                 RMDIR(1)

NAME
     rmdir -- remove directories

SYNOPSIS
     rmdir [-p] directory ...

DESCRIPTION
     The rmdir utility removes the directory entry specified by each directory
     argument, provided it is empty.

     Arguments are processed in the order given.  In order to remove both a
     parent directory and a subdirectory of that parent, the subdirectory must
     be specified first so the parent directory is empty when rmdir tries to
     remove it.

     The following option is available:

     -p      Each directory argument is treated as a pathname of which all
             components will be removed, if they are empty, starting with the
             last most component.  (See rm(1) for fully non-discriminant
:
```

Outline

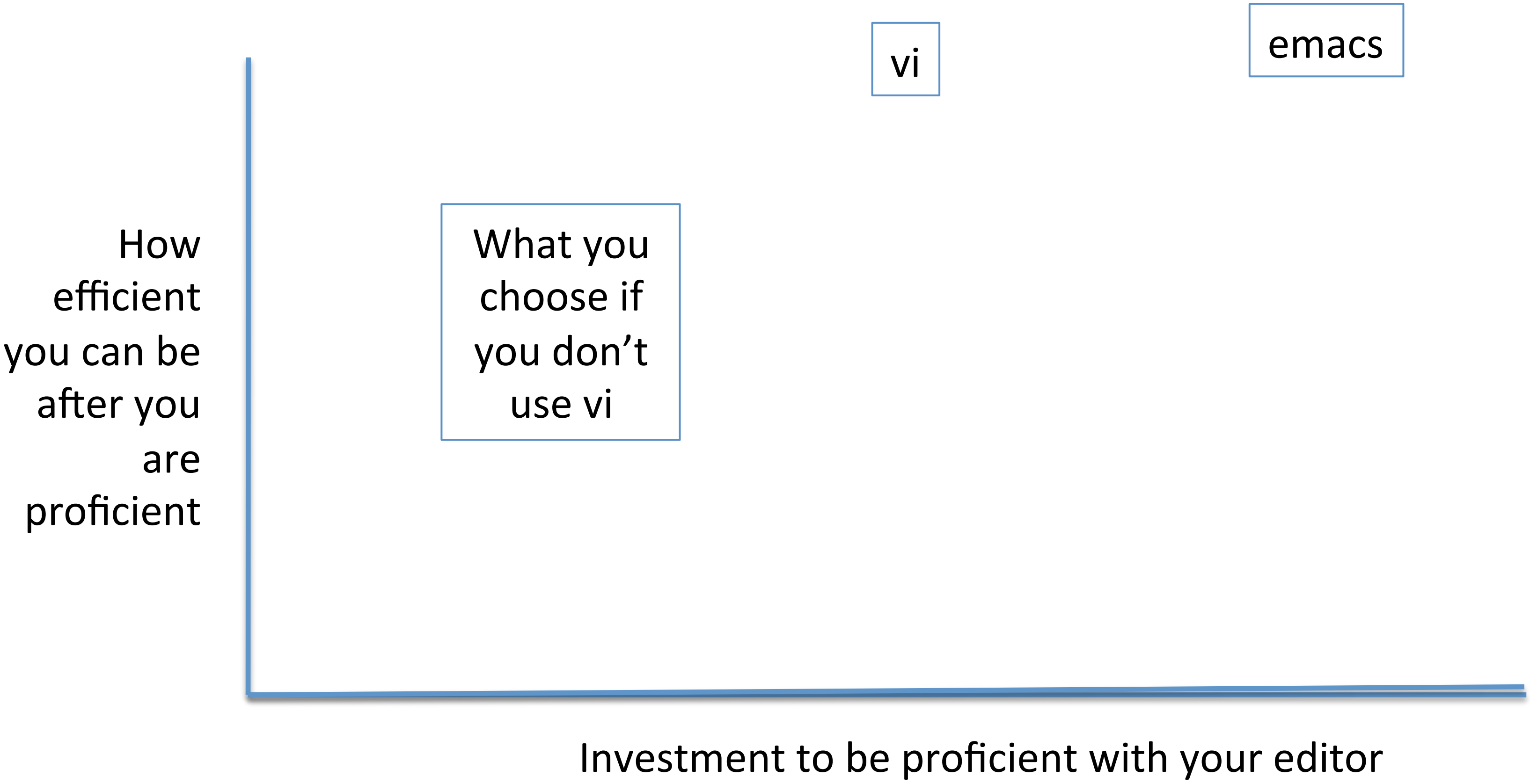
- Class Overview
  - My Background
  - Goals
  - Syllabus
- Getting Started With Unix
  - Unix History
  - Shells
  - Files
  - File Editors
- Project 1

File Editors

- Existing file editors:
  - Vi
  - Emacs
  - Two or three hot new editors that everyone loves (and ultimately fade away and die)
- This has been the state of things for 25 years

I will teach "vi" in this course.
You are welcome to use whatever editor you want.

My Mental Model for File Editors



vi

emacs

How efficient you can be after you are proficient

What you choose if you don’t use vi

Investment to be proficient with your editor

Vi has two modes

- Command mode
  - When you type keystrokes, they are telling vi a command you want to perform, and the keystrokes don’t appear in the file
- Edit mode
  - When you type keystrokes, they appear in the file.

Transitioning between modes

- Command mode to edit mode
  - i: enter into edit mode at the current cursor position
  - a: enter into edit mode at the cursor position immediately to the right of the current position
  - I: enter into edit mode at the beginning of the current line
  - A: enter into edit mode at the end of the current line

There are other ways to enter edit mode as well

Transitioning between modes

- Edit mode to command mode
  - Press Escape

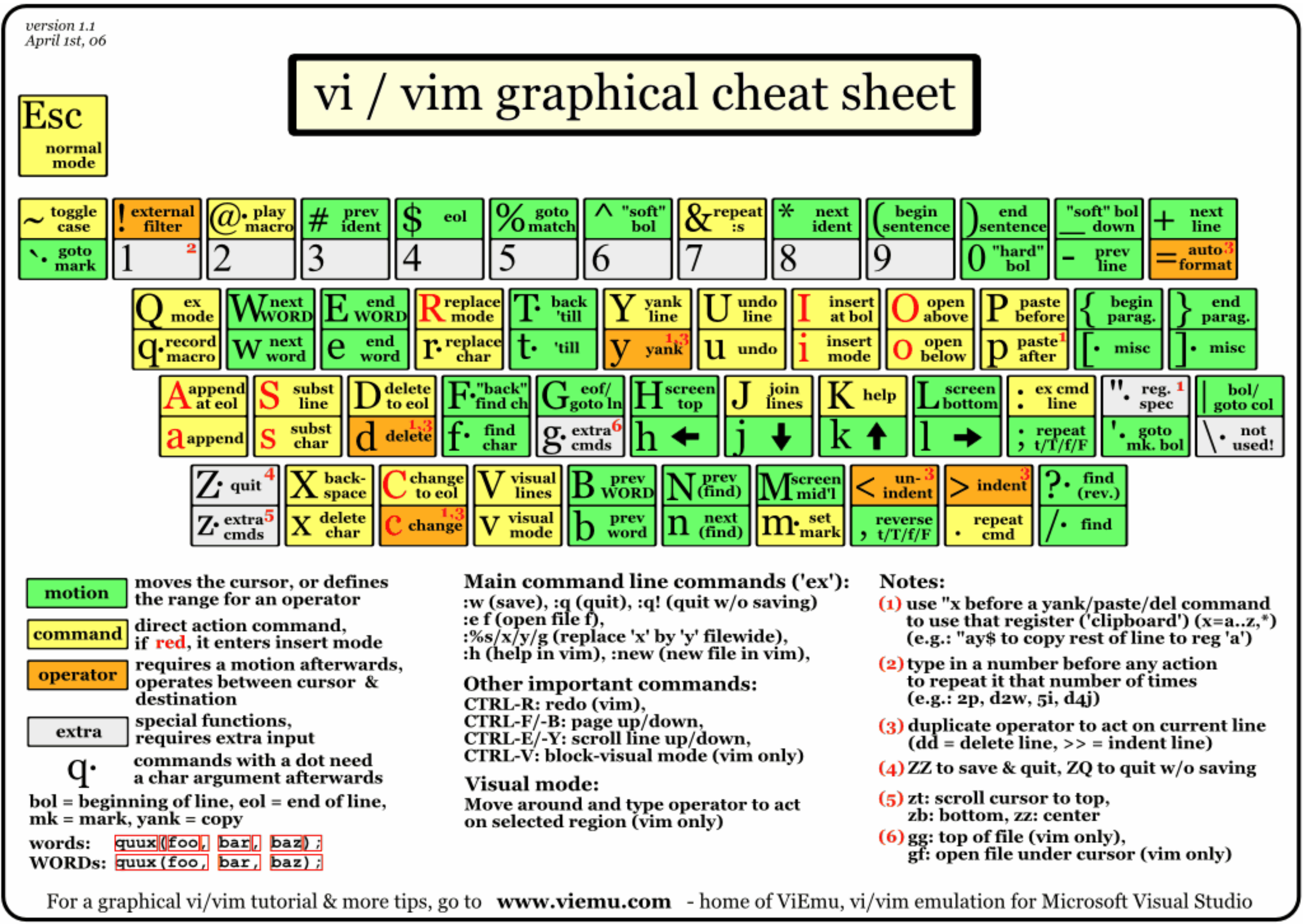
Useful commands

- yy: yank the current line and put it in a buffer
  - 2yy: yank the current line and the line below it
- p: paste the contents of the buffer
  - 2pp: past the contents of the buffer two times
- x: delete the character at the current cursor
- ":100" go to line 100 in the file
- Arrows can be used to navigate the cursor position (while in command mode)
  - So do h, j, k, and l

We will discuss more tips for "vi" throughout the quarter. They will mostly be student-driven (Q&A time each class)

My first vi sequence

- At a shell, type: “vi cis330file”
- Press ‘i’ (to enter edit mode)
- Type “I am using vi and it is fun” (text appears on the screen)
- Press “Escape” (to enter command mode)
- Press “:wq” (command mode sequence for “write and quit”)



version 1.1
April 1st, 06

vi / vim graphical cheat sheet

Esc normal mode

Key	Action	Key	Action
~	toggle case	`	goto mark
!	external filter	1	[2]
@	play macro	2	
#	prev ident	3	
$	eol	4	
%	goto match	5	
^	"soft" bol	6	
&	repeat :s	7	
*	next ident	8	
(	begin sentence	9	
)	end sentence	0	"hard" bol
_	"soft" bol down	-	prev line
+	next line	=	auto format [3]

Key	Action	Key	Action
Q	ex mode	q·	record macro
W	next WORD	w	next word
E	end WORD	e	end word
R	replace mode	r·	replace char
T·	back 'till	t·	'till
Y	yank line	y	yank [1,3]
U	undo line	u	undo
I	insert at bol	i	insert mode
O	open above	o	open below
P	paste before	p	paste after [1]
{	begin parag.	[·	misc
}	end parag.	]·	misc

Key	Action	Key	Action
A	append at eol	a	append
S	subst line	s	subst char
D	delete to eol	d	delete [1,3]
F·	"back" find ch	f·	find char
G	eof/ goto ln	g·	extra cmds [6]
H	screen top	h	←
J	join lines	j	↓
K	help	k	↑
L	screen bottom	l	→
:	ex cmd line	;	repeat t/T/f/F
"·	reg. spec [1]	'·	goto mk. bol
|	bol/ goto col	\\·	not used!

Key	Action	Key	Action
Z·	quit [4]	z·	extra cmds [5]
X	back-space	x	delete char
C	change to eol	c	change [1,3]
V	visual lines	v	visual mode
B	prev WORD	b	prev word
N	prev (find)	n	next (find)
M	screen mid'l	m·	set mark
<	un-indent [3]	,	reverse t/T/f/F
>	indent [3]	.	repeat cmd
?·	find (rev.)	/·	find

motion moves the cursor, or defines the range for an operator

command direct action command, if red, it enters insert mode

operator requires a motion afterwards, operates between cursor & destination

extra special functions, requires extra input

q· commands with a dot need a char argument afterwards

bol = beginning of line, eol = end of line,
mk = mark, yank = copy

words: quux(foo, bar, baz);
WORDs: quux(foo, bar, baz);

Main command line commands ('ex'):

:w (save), :q (quit), :q! (quit w/o saving)
:e f (open file f),
:%s/x/y/g (replace 'x' by 'y' filewide),
:h (help in vim), :new (new file in vim),

Other important commands:

CTRL-R: redo (vim),
CTRL-F/-B: page up/down,
CTRL-E/-Y: scroll line up/down,
CTRL-V: block-visual mode (vim only)

Visual mode:

Move around and type operator to act on selected region (vim only)

Notes:

(1) use "x before a yank/paste/del command to use that register ('clipboard') (x=a..z,*) (e.g.: "ay$ to copy rest of line to reg 'a')

(2) type in a number before any action to repeat it that number of times (e.g.: 2p, d2w, 5i, d4j)

(3) duplicate operator to act on current line (dd = delete line, >> = indent line)

(4) ZZ to save & quit, ZQ to quit w/o saving

(5) zt: scroll cursor to top, zb: bottom, zz: center

(6) gg: top of file (vim only), gf: open file under cursor (vim only)

For a graphical vi/vim tutorial & more tips, go to **www.viemu.com** - home of ViEmu, vi/vim emulation for Microsoft Visual Studio

http://www.viemu.com/vi-vim-cheat-sheet.gif

vimtutor

- Past students have liked vimtutor

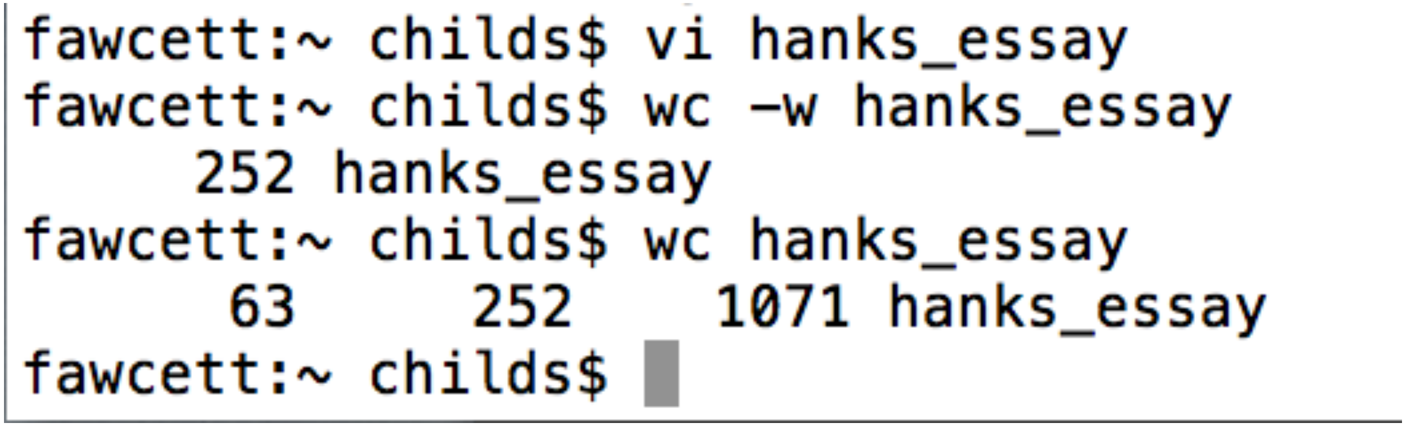
Project 1A

- Practice using an editor
- Must be written using editor on Unix platform
  - I realize this is unenforceable.
  - If you want to do it with another mechanism, I can’t stop you
    - But realize this project is simply to prepare you for later projects
- Due Wednesday (→ 6am Thursday)

Project 1A

- Write >=300 words using editor (vi, emacs, other)
- Topic: what you know about C programming language
- Can’t write 300 words?
  - Bonus topic: what you want from this course
- How will you know if it is 300 words?
  - Unix command: “wc” (word count)

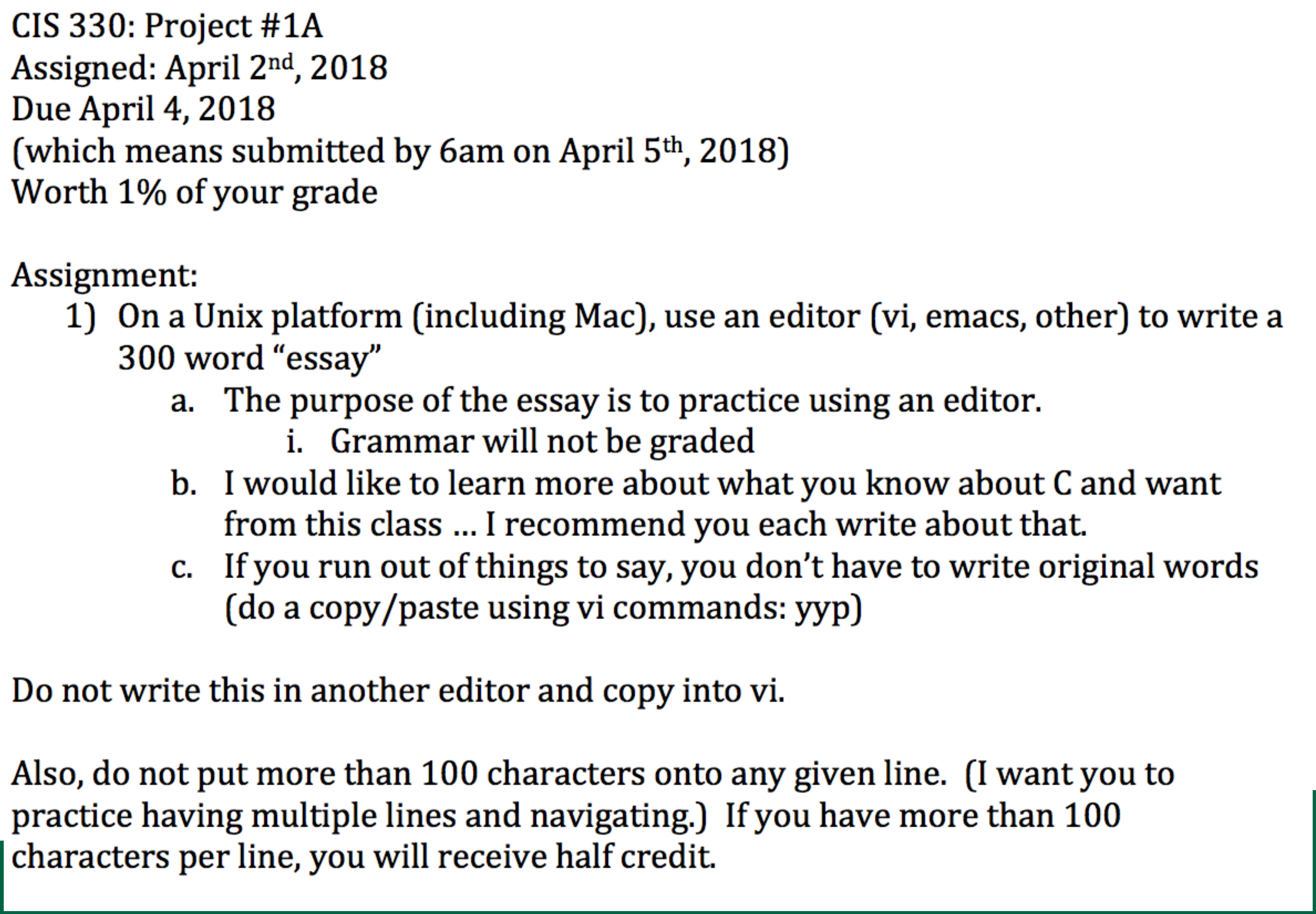
Unix command: wc (word count)



```
fawcett:~ childs$ vi hanks_essay
fawcett:~ childs$ wc -w hanks_essay
     252 hanks_essay
fawcett:~ childs$ wc hanks_essay
      63     252    1071 hanks_essay
fawcett:~ childs$ 
```

(63 = lines, 252 = words, 1071 = character)

Project 1A



CIS 330: Project #1A
Assigned: April 2nd, 2018
Due April 4, 2018
(which means submitted by 6am on April 5th, 2018)
Worth 1% of your grade

Assignment:

1) On a Unix platform (including Mac), use an editor (vi, emacs, other) to write a 300 word "essay"
    a. The purpose of the essay is to practice using an editor.
        i. Grammar will not be graded
    b. I would like to learn more about what you know about C and want from this class ... I recommend you each write about that.
    c. If you run out of things to say, you don't have to write original words (do a copy/paste using vi commands: yyp)

Do not write this in another editor and copy into vi.

Also, do not put more than 100 characters onto any given line. (I want you to practice having multiple lines and navigating.) If you have more than 100 characters per line, you will receive half credit.

Accessing remote machines

- Windows->Unix
  - ??? (Hummingbird Exceed was the answer last time I used Windows)
- Unix->Unix
  - ssh: secure shell   ssh –l hank ix.cs.uoregon.edu
  - scp: secure copy   scp hank@ix.cs.uoregon.edu:~/file1 .
    - Also, ftp: file transfer protocol

Note on Homeworks

- Project 1A due on Wednesday
- Project 2A will be assigned on Wednesday
  - Due in class on Monday
- Project 1B will be assigned on Friday
  - Due on Weds, April 11
- I will lecture on Friday (instead of lab)

How to submit

- Canvas
- If you run into trouble:
  - Email me your solution

Don’t forget

- This lecture is available online
  - http://ix.cs.uoregon.edu/~hank/330

- All project prompts are available online