CIS 330:

Lecture 1:
Course Overview & Introduction to Unix
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
My Background

• Joined UO in March 2013
• Previously:
  – Lawrence Livermore, 1999-2009
  – Lawrence Berkeley, 2009-
  – UC Davis 2009-2013
• Education:
  – 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

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

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
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!!!
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”
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: Weds/Fri 2:00-3:20
Lecture Location: 166 Lawrence Hall
Discussion Times: Fri 11:00-11:50, 12:00-12:50, 16:00-16:50
Discussion Location: 26 Klamath Hall

Instructor: Hank Childs
Instructor Office Hours: Thurs 13:00-14:30, Fri 11:00-12:00
Office Hours Location: 301 Deschutes Hall

Teaching Assistants: Pavel Govyadinov, Hang (Matt) Wu
Teaching Assistants' Office Hours: TBD
Office Hours Location: TBD
Grading For This Course

- Final: 30%
- Projects (made up of multiple pieces)
  - Project 1: Learning Unix (8%)
    - 1A (2%), 1B (3%), 1C (3%)
  - Project 2: C/C++ Primer (15%)
  - Project 3: Large System + Object Oriented Programming (37%)
  - Project 4: Debugging/Profiling (10%)
- Extra Credit through:
  - Unused late passes
  - 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

<table>
<thead>
<tr>
<th>Project</th>
<th>Assigned</th>
<th>Due</th>
<th>Task</th>
<th>% Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1A</td>
<td>4/2</td>
<td>4/4</td>
<td>Editing Files</td>
<td>2%</td>
</tr>
<tr>
<td>1B</td>
<td>4/4</td>
<td>4/11</td>
<td>Scripts/Permissions</td>
<td>3%</td>
</tr>
<tr>
<td>1C</td>
<td>4/4</td>
<td>4/11</td>
<td>Build</td>
<td>3%</td>
</tr>
<tr>
<td>2A</td>
<td>4/11</td>
<td>4/15</td>
<td>File I/O</td>
<td>5%</td>
</tr>
<tr>
<td>2B</td>
<td>4/16</td>
<td>4/20</td>
<td>Unions + Function Pointers</td>
<td>5%</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Norms for this class

• Please ask questions
• Please ask me to slow down
• Please give feedback

• Please always bring paper and pencil/pen ... we will do exercises during class
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.

• Let’s all do our best to give the other the benefit of the doubt.
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.
Do You Need To Attend Class?

• The class is not part of your grade

• However:
  – 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)
  – A lot of the material will come via lecture
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.
Course Materials

• Textbook: Thinking in C++ (free online book)

• PowerPoint lectures will be posted online.

• Other materials (e.g., webpages) will be occasionally used

• I will “live code” sometimes.
Office Hours

• OH: Thurs 10:30-12:00, Fri 11-12
  – Location: 301 Deschutes (new!)
• TA OH: will be determined soon
Working Together

• All projects are individual projects.
• Copying code from other students is cheating.
• However: I highly encourage you to discuss your roadblocks with each other and lean on each other to figure out solutions to your problems.
Working Together, part 2

• I have set up a forum on Piazza.
  – I will monitor (and respond) to the forum and encourage you all to do the same.
  – I may award extra credit to students who are particularly helpful on Piazza.
  – The amount of credit will vary based on involvement, with a maximum of 5%.
Late Passes

- You have 2 "late passes."
- Late passes allow you to turn in your project after the due date for full credit.
  - Submitting a project with a Weds deadline on Friday (i.e., two days later) costs one late pass.
  - Submitting a project with a Friday deadline on the following Weds (i.e., five days later) costs two late passes.
  - Projects with a Friday deadline can possibly be submitted on a Monday for one late pass, but my availability on Mondays is not guaranteed and you may be forced to wait until Weds (and pay two late passes).

- If you run out of late passes, then you may continue to earn half credit on any project.
- Every unused late pass is worth 1% extra credit.
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
Norms for Getting In Touch With Me

• OH

• I’m pretty good about email
  – If I don’t respond within 24 hours, resend

• It is almost never a good idea to drop by my office without an appointment
  – If you want to meet personally, set up an appointment
    • http://cdux.cs.uoregon.edu/childs_calendar.html
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
  - Virtual machines

Who has home access to a Unix environment?

Who has Windows only and wants to pursue Cygwin/VM & needs help?
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)
Shells

• Shells are interpreters
  – Like Python

• You type a command, it carries out the command
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

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

- "~" (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

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/ tribunal
        » (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 program’s behavior.
  - Convention of using hyphens to signify special status
- “ls” is also useful with “wild cards”, which we will also discuss later
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

• 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.
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
• 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”)
vi / vim graphical cheat sheet

Esc

normal mode

~ toggle case 1 2 3 4 5 6 7 8 9 0 "hard" bol
\=. mark

Q ex mode
W next WORD
E end WORD
R replace mode
T back 'till
Y yank line
U undo
I insert at bol
O open above
P paste before
\{ begin parag.
\} end parag.

A append at eol
S subst line
D delete to eol
F "back" find ch
G eol/\g
go to
H screen top
J join lines
K help
L screen bottom
\", reg. spec
\[ bol/\g
go to col
\] not used!

Z: quit
\(\) extracmds
\[\) delete char
\] change to eol
V visual mode
B prev WORD
N prev (find)
M search mid
< unindent
> indent
? find (rev.)
*/ find

motion

moves the cursor, or defines the range for an operator
direct action command, if red, it enters insert mode
operator

requires a motion afterwards, operates between cursor & destination
special functions, requires extra input
command

Q: quit 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
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
Project 1A

• Write >=500 words using editor (vi, emacs, other)

• Topic: what you know about C programming language

• Can’t write 500 words?
  – Bonus topic: what you want from this course

• How will you know if it is 500 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, 2014
Due April 4th, 2014
(which means submitted by 6am on April 5th, 2014)
Worth 2% of your grade

Assignment:
   1) On a Unix platform (including Mac), use an editor (vi, emacs, other) to write a 500 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.

Turn this in using the “turnin” link on the class website.
Note on Homeworks

• Project 1A due on Friday
• Projects 1B and 1C will be assigned on Friday, due following Friday
My advice

• Take some time between now and the discussion section to play with an editor
• Ask questions at discussion section, submit Friday night.
How to submit

• We are using CIS turnin program
• Link on class webpage
• If you run into trouble:
  – Email me your solution
• If you do not have a CIS account:
  – Email me to resolve this problem
What should you do if you run into trouble?

1) Start with Piazza

2) OH this week:
   - Thurs: 10:30-12
   - Fri: 11-12

3) Email me: hank@cs.uoregon.edu
Don’t forget

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

• All project prompts are available online