

# Introduction to 15-412

Dave Eckhardt  
[de0u@andrew.cmu.edu](mailto:de0u@andrew.cmu.edu)

# Synchronization

- Textbook?
  - Previously I suggested The Practice of Programming
  - Students claimed they knew it all already
  - If you're not sure, take a look at [ttop.awl.com](http://ttop.awl.com)

# Outline

- Introductions
  - [If not now, when?]
- Administrative information
- Class goals
- Grading: philosophy, mechanisms
- Project categories
- Reading material

# Information Sources

- Web site <http://www.cs.cmu.edu/~412>
  - See syllabus
- Coming to class
  - Vital, at least initially
  - Later, one class per week may be “project time”

# Academic Conduct

- I firmly expect everybody knows the rules
- A 412-specific issue: licenses
  - We need to pay attention to them and follow them
  - No disassembling Microsoft products!
  - Code transfers between projects must be
    - Credited appropriately
    - In compliance with both licenses
  - Code is probably better as a textbook than as building material

# Course Goals

- Hands-on experience with “OS” code in real world
  - Build environments
  - Portability issues
  - People issues
- Contributing something to the global software community...
  - Something useful – submission-quality

# Course Goals

- Research is a “mild anti-goal”
  - 15-712 is a “standard grad OS class”
  - Core target of grad-school research is scientific
    - Evaluating a hypothesis or proposal
    - Need a prototype good enough to measure
      - Rarely good enough to *deploy*.
        - Notable local exceptions: AFS, Mach, Coda
  - A “standard grad OS class” covers
    - Reading current literature to understand the current frontiers
    - *Practicing* the problem/solution/evaluation cycle
    - *Ideally* an artifact somebody uses after the semester

# Course Goals

- Meanwhile...
  - Employers want somebody who can write a device-driver today.
    - ...As part of a large OS (or network OS) project...
    - ...Based on incomplete documentation...
    - ...Dealing with buggy hardware...
    - ...But which works reliably.
  - The world has lots of (quality) low-level software still unwritten.

# Course Plan

- Lectures
  - Not a key part of the course!
  - Some initial start-up lectures
  - Extended answers to technical questions
    - (so bring some to class)
  - Discussion of interesting papers
  - Status updates, mini-presentations, design sessions

# Course Plan

- Projects
  - I have some suggestions
    - Security, file systems, networking, “pure kernel”, drivers...
  - Proposing your own project is *encouraged*
- Samples
  - <http://www.cs.cmu.edu/~412/projects/>
  - So far roughly half Linux, half Plan 9

# Course Plan

- Project proposal, things like
  - What existing code does
  - What you want to add
  - Who else is working in the area
  - Lines of code (entire project, broken down by area)
  - Lines of code (you expect to write)
  - Relevant licenses
  - Web resources
  - Standard acceptance process for code in this project

# Unit Count

- What is 9 units?
  - Can be a solid accomplishment
  - Can also be “lost in the shuffle”
- Numbers
  - Subtract 3 hours per week in class (probably less)
  - 6 hours/week \* 15 weeks = 90 hours
  - 90 hours/week = 20 hours/week \* 4.5 weeks
    - Half-time seasoned kernel hacker for a month
    - Roughly enough time for two people to bang out first Unix

# Time Recommendation

- *Schedule* joint work sessions
  - Minimum of 3 hours per session
  - Two to three times per week
- “Schedule” means setting aside repeating fixed time slots
  - Will make better use of lab space
  - Will make it easier for me to drop by

# Grading “Philosophy”

- You shouldn't be here unless you are...
  - technically solidly prepared
  - inspired by your chosen area of endeavor
  - committed to taking pride in your work
- Sounds like a recipe for success!

# Grading “Guidelines”

- A - “Should be accepted into distribution”
  - Something useful to a project works robustly
  - Code quality is good and style fits that project
  - “Enough” documentation is written up
- B - “Work in progress, useful to humanity”
  - Identifiable interesting new features work
  - Quality and packaging good enough for follow-on work
    - “make” + README.DESIGN + TODO = launch
- C - “learning happened but not development”

# Mid-Semester Grades

- A mid-semester grade of C or below is a signal
  - “It doesn't look like you're on track to improve the status of humanity”
- Not a sin!
  - But not what the course goals aspire to, either

# Grading Mechanisms

- Smaller items
  - Class presentations (everybody!)
    - “Concept proposal”, “Topic lecture”/“Reading report”
  - Proposal writeup
  - Demo days (3-5, 1<sup>st</sup> before mid-semester)
    - Everybody meets, each group demonstrates what works
- More important
  - Code accomplishments
  - Code quality (“invisibly improve” what's there)

## *Estimated* Breakdown

<b>Weight</b>	<b>Item</b>
10%	Project design
10%	Class presentations
10%	Planning/status
15%	Code quality
55%	Goal completion

# On the Horizon

- Week 1 – talking about course & projects
- Week 2 – groups & projects declared
- Week 3 – flailing around with/in your project
- Week 4 – proposal presentation (or exception)

# Project Categories / Ideas

- What do you think you're interested in?
- Would you like to work with somebody?

# Project Categories / Ideas

- Kernel contributions
  - hardware related (not “just device driver hacking”)
  - ok, device driver hacking is an option too
  - [kernelnewbies.org/projects](http://kernelnewbies.org/projects) (with caveats)
- File systems
- Window system / graphics
- Security infrastructure
- “Platforms”
  - LinuxBIOS, Xen, QEMU/Bochs

# Upcoming

- Wednesday we'll talk more about projects
- Please begin reading for next Monday
  - A Comparison of Two Distributed Systems: Amoeba and Sprite
    - <http://www.eecs.berkeley.edu/Research/Projects/CS/sprite/sprite.papers.html>
  - Bring to class on a scrap of paper
    - 3 things you respect/envy about the paper
    - 3 things you were not convinced by
    - Guess: Why did I ask you to read this paper?