

Introduction to 15-412

Dave Eckhardt
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

Outline

- Introductions
 - [If not now, when?]
- Administrative information
- Class goals
- 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 use.
 - Notable local exceptions: AFS, Mach, Coda

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...
 - The world has lots of (quality) low-level software still unwritten.

Course Plan

- Lectures
 - Not entirely
 - 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”
 - Proposing your own project is encouraged
- Samples
 - <http://www.cs.cmu.edu/~412/projects/>
 - So far roughly half Linux, half Plan 9
- Ideas
 - See course Wiki

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 set time, for real
 - 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 the area of endeavor
 - committed to taking pride in your work
- Sounds like a recipe for success!

Grading Mechanisms

- Smaller pieces
 - Proposal (final version), web page
 - Weekly status “blog”
 - Mini-presentations as appropriate
- More important
 - Code accomplishments
 - Code quality (“invisibly improve”)
 - Code review
 - Testing approach

Grading Mechanisms

Weight	Item
10%	Project design
15%	Class presentations
10%	Blogging
15%	Source control
15%	Code quality
35%	Goal completion

Project Categories / Ideas

- What do you think you're interested in?

Project Categories / Ideas

- Kernel contributions
 - hardware related (not “just device driver hacking”)
 - ok, device driver hacking is an option too
 - 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 Friday
 - A Comparison of Two Distributed Systems: Amoeba and Sprite
 - <http://cs.berkeley.edu/projects/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?