Syllabus

Meetings: Tuesdays and Thursdays, 12:30-1:45pm in Rice Hall 130 ("Olsson Auditorium")

Teacher: David Evans.

Assistant Teachers/Teacher's Assistants: Alex Lamana, Ben Terner, Corey Ames, Josh Lisko, Purnam Jantrania, Rob Michaels, Weilin Xu, Wil Thomason, and Zeming Lin.

Office Hours: To be scheduled.

Course Site: All course materials will be posted at http://rust-class.org/.

IRC: We will use the "cs4414" chat room at *irc.mozilla.org* for real-time chat. (See IRC for more on IRC.)

Textbook: We will not follow a textbook closely, but will have several readings from Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau's Operating Systems: Three Easy Pieces book. This book is freely available on-line, but may also be ordered as a printed book for $29. (If you are not sure if you should prefer reading things on paper, you should read *The Reading Brain in the Digital Age: The Science of Paper versus Screens*, Scientific American, 11 April 2013.)

Overview

According to the official course description, this course *"Analyzes process communication and synchronization; resource management; virtual memory management algorithms; file systems; and networking and distributed systems."*

That description describes some of the topics we will cover in cs4414, but doesn't capture *why* anyone would want to take a course on these topics, or what you should hope to get out of it.

The primary goal of this course is **to improve your ability to build scalable, robust and secure computing systems**. We focus on doing that by **understanding what underlies the core abstractions of modern computer systems**.

We want to do this because:

1. Understanding how these abstractions are designed and implemented will enable you to use them more effectively.
2. Exploring these abstractions will get at some of the most intellectually interesting concepts in compter science, in the context of making practical systems.
3. Being able to modify computer systems at a lower level gives you powers that mere application programmers can only envy.

Most programming today is done at a very high level using libraries and tools that shield programmers from needing to understand much about what is going on underneath. In this class, we will enter

the black boxes that most programs use as abstractions and understand how they work and how to implement them efficiently and robustly.

The other main goal of this course is to provide experiences and knowledge that will help you **develop as professional programmers and computing system designers**. This includes:

- Experience working with larger and more complex programs (than you have encountered in previous classes).
- Experience working in a team, both as a leader and contributor.
- Experience learning to use a new programming language mostly on your own, without much guidance or available documentation.
- Experience using tools commonly used by productive developers (including git and Make).

If you do not feel my goals for the course align well with your personal goals, but you need to take this course anyway to satisfy a degree requirement, you should meet with me to figure out a way to make this course useful for satisfying your personal goals.

Expected Background

Students entering this course are expected to be comfortable reading, designing, and writing complex programs that involve thousands of lines of code distributed over many modules. You should be comfortable learning how to use new programming language features and APIs by reading their documentation (or source code when no documentation is available), and not be surprised when solving programming assignments requires you to seek documentation beyond what was provided in class.

Some specific things I expect of students entering this course:

- You should have some experience programming in C or C++ and some exposure to assembly language programming (at least as much as is covered in cs2150).
- You have written at least one program with over 5000 lines of code.
- You should be able to explain the difference between inheritance and subtyping.
- You should be comfortable writing procedures that take other procedures as inputs.
- You should understand why it is impossible to determine if an *arbitrary* program will fail to run to completion (e.g., exit with a run-time error), but why it is often possible to reason about correctness of a *given* program.
- You should be able to explain why an algorithm whose worst-case asymptotic running time is in $\Theta(N \log N)$ may be preferred to one whose worst-case asymptotic running time is in $\Theta(N)$.
- You should find computing exciting and delighting, and believe you can use computing to make the world a better place.

Honor

As a student at Mr. Jefferson's University, **you are trusted to be honorable**.

We take advantage of this trust to provide a better learning environment for everyone. In particular, students in cs4414 are expected to follow these rules:

- **I will not lie, cheat or steal.** If I am unsure whether something would be considered lying, cheating or stealing, I will ask before doing it.
- **I will carefully read and follow the collaboration policy on each assignment.** I will not abuse resources, including any submissions or solutions for assignments from last semester's version of this course, that would be clearly detrimental to my own learning.
- **I will do what I can to help my fellow classmates learn.** Except when specifically instructed not to, this means when other students ask me for help, I will attempt to provide it. I will look at their answers and discuss what I think is good or bad about their answers. I will help others improve their work, but will not give them my answers directly. I will try to teach them what they need to know to discover solutions themselves.
- **I will ask for help.** I will make a reasonable effort to do things on my own first (or with my partners for group assignment), but will ask my classmates or the course staff for help before getting overly frustrated. There are many ways to ask for help including the course website, IRC, and office hours.
- **I grant the course staff permission to reproduce and distribute excerpts from my submissions for teaching purposes.**
- **I will provide useful feedback.** I realize that this is a new and experimental course, and it is important that I let the course staff know what they need to improve the course. I will not wait until the end of the course to make the course staff aware of any problems. I will provide feedback either anonymously or by contacting the course staff directly. I will fill out all requested surveys honestly and thoroughly.

Assignments and Exams

The main assignments and deadlines are:

- **Problem Set 0** Course registration and Rust tutorial (Due 18 January)
- **Problem Set 1** Zhttpto: simple web server (Due 23 January)
- **Problem Set 2** Gash: shell (Due 9 February) - Processes
- **Problem Set 3** Zhtta: web server (Due 3 March) - Synchronization, Memory Management
- **Problem Set 4** IronKernel (Due 2 April) - Modifying a kernel
- **Final Project** (Due 29 April, with several earlier intermediate deadlines) - Almost anything you want

For most of the problem sets, you will work in teams. Except in unusual circumstances, all members of a team will receive the same grade, although students will be expected to provide honest and confidential feedback on their teamates.

There will be two exams during the semester:

- **Exam 1** (Out 10 February, Due 13 Feburary)
- **Exam 2** (Out 4 March, Due 7 March)

Unless I have reason to doubt the class is following the honor policy well, both examples will be take-home, open-resources exams. Most of the questions on the exams will be taken directly from questions on the provided course notes, with a few additional synthesis questions. We will not use the registrar scheduled in-class final, and for most students the final project will eliminate the need for a final exam. There will be an opportunity for students where this is not the case to schedule an optional oral final exam.

A tenative and (continually) updated schedule is available as Google calendar. Except when noted otherwise, assignments are due at 11:59pm on the due date.

Grading

I prefer to spend my time, as well as the TAs time, focused as much as possible on *teaching*, and as little as possible on *grading*. The assignments in this class are designed to maximize *learning*, rather than primarily for *assessment*.

That said, I understand that students do need to be assigned grades at the end of the semester, and sometimes grades are a powerful and effective motivator. Grades will be determined based on your performance on the problem sets, final project, exams, and class contributions (including on-line contributions). There is no predefined weighting. In general, if there is some combination of the above that demonstrates that you have gotten what I hope out of the class then you'll receive an A grade.

