COURSE SYLLABUS

Last updated: March 26, 2016.

Spring Term, 2016

Course Description

This course is about the development of low-level, bare-metal systems—with particular focus on microkernels—and about the role and design of programming languages in this application domain—from low-level assembly, to high-level functional and domain specific languages. The course material is organized in to three parts:

1. **An overview of conventional, low-level programming techniques.** This will include an introduction to some of the programmable hardware components of a typical PC system (such as the memory management unit, interrupt controller, and timer) and their use in the implementation of fundamental operating system abstractions.

2. **Case studies of practical microkernel implementations.** These case studies will be used to develop a more detailed understanding of the requirements for the programming languages that are used in their construction. The two specific microkernels that we plan to study are "pork", which is an implementation of L4 developed at PSU, and "seL4", which is a "security-enhanced" version of L4, developed by a team in Australia, that uses a capability abstraction for access control and resource management. The seL4 system, in particular, has gained some fame in recent years as "the world's first operating system kernel with an end-to-end proof of implementation correctness and security enforcement."

3. **Reflections on the design of programming languages for use in this application domain**. This part of the course will build on and be informed by the examples that we have seen in earlier sections. In particular, we will look at some of the distinguishing features of Habit, a functional programming language with some specific language features and a broad adoption of abstract datatypes, that has been specifically designed to target the development of low-level systems.

Context: The CEMLaBS project:

This course is being offered for the second time in Spring 2016. It was taught for the first time in Spring 2015 in conjunction with "CEMLaBS", which is an NSF-supported research project that began in October 2014. (The "CEMLaBS" name is a long acronym for an even longer formal title: "Using a Capability-Enhanced Microkernel as a Testbed for Language-based Security".) As such, it is important for students to understand that this course will have an experimental and exploratory feel, with a more fluid and flexible syllabus than more established courses. This will not be the kind of class where you can catch up on a missed lecture by reviewing a textbook: much of the material that we will be covering has yet to be documented in that form!

The following text provides some brief context for the CEMLaBS project and a summary of its research goals:

> The work on seL4 is rightly celebrated as a landmark for formal verification, and provides a strong foundation for building trustworthy systems. The costs of its development and maintenance, however, are quite high. Moreover, many of the security properties established in

the seL4 verification (including the absence of code injection attacks, buffer overflows, and null pointer dereferences) could have been guaranteed automatically by writing the microkernel in a language that incorporates language-based security mechanisms such as an enhanced type system. Motivated by such insights, the specific goals of the CEMLaBS project are to evaluate: whether it is actually feasible to write software of this kind in a language with meaningful, language-based security guarantees; what practical impact this might have, particularly in reducing verification costs; and whether it is possible to meet the performance goals that are typically required in such applications.

We hope to offer three versions of the course described here over a period of three years as the CEMLaBS project evolves. For this offering, we will focus mostly on a review of the ideas and material that motivate the work on CEMLaBS, and most of the code that we study and write will be using languages like C and assembly. Our use of the Habit programming language at this time will, for the most part, be limited to a review of its design and goals. For this reason, some students may prefer to wait for a future version of this course where we expect to make more practical use of Habit once the research has developed further and the compiler implementation is more mature.

Prerequisites:

There are no formal prerequisites for this class, but students are expected to have some background in operating systems (as a result of taking a class like CS333 at the undergraduate level or CS533 at the graduate level) and programming languages (as a result of taking classes like CS321/322 at the undergraduate level or CS558 at the graduate level). As a practical consideration, this class will involve substantial programming exercises using C and assembly language in a IA32 Linux environment. Prior experience with the Linux command line and standard toolset will be expected, together with significant programming experience.

Course Objectives:

Upon the successful completion of this course, students will be able to:

1. Write simple programs that can run in a bare-metal environment using low-level programming languages.
2. Discuss common challenges in low-level systems software development, including debugging in a bare-metal environment.
3. Explain how conventional operating system features (multiple address spaces, context switching, protection, etc.) motivate the desire for (and benefit from) hardware support.
4. Develop code to configure and use programmable hardware components such as a memory management unit (MMU), interrupt controller (PIC), and interval timer (PIT).
5. Describe the key steps in a typical boot process, including the role of a bootloader.
6. Describe the motivation, implementation, and application of microkernel abstractions for managing address spaces, threads, and interprocess communication (IPC).
7. Explain the use and implementation of capabilities in access control and resource management.
8. Develop programs using the capability abstraction provided by the seL4 microkernel.
9. Illustrate the use of a range of domain specific languages in the development of systems software.
10. Use practical case studies to evaluate and compare language design proposals.
11. Describe features of modern, high-level programming languages---including abstract datatypes and higher-order functions---and show how they can be leveraged in the construction of low-level software.
12. Explain how the requirements of low-level systems programming motivate the desire for (and benefit from) language-based support.

Masters Tracks:

For students taking the CS 510 version of this course as part of a Masters Degree program, it is intended that this class can be used as a contribution towards the "Languages" track of the masters program. It may also be possible for this class to be counted towards the systems track, or perhaps the security track instead, subject to department, instructor, and advisor approval. Please see the instructor if you wish to pursue this possibility.

Organizational Details

Time and Location:
Lectures: UTS 210, Monday noon-1:50pm
Labs: FAB 88-09, Wednesday noon-1:50pm
Office Hours: FAB 120-20, Monday 3-4pm, or by appointment.

Students are expected to attend all scheduled lectures and labs. Sets of practical exercises will be provided for students to start working on during lab sessions with instructor supervision. Any parts of the exercises that are not completed during the lab session must be finished before the start of the following week's lab (ideally, before the start of the following week's lecture). This is necessary because each set of lab exercises will typically require successful completion of all the previous exercises. The discussion forums on D2L will be used as the primary means for addressing questions from students who are working on the exercises in their own time outside the lab sessions, although interaction in person or via email will also be possible. If necessary, Wednesday lab sessions will also be used for direct instruction, for example, to finish the presentation of material that could not be covered during the preceding Monday lecture.

Instructor: Mark P Jones, Department of Computer Science, Maseeh College of Engineering & Computer Science, Portland State University.

You can contact me by email (mpj@pdx.edu), phone (503 725 3206, but sometimes hit and miss), or in person (FAB 120-20). I usually find that personal meetings work better than email, so please feel free to drop by my office at any time; Please knock, even if it looks like I am already busy; if I am busy, then we can at least arrange another, mutually convenient time to meet. I sometimes work without the main lights on (i.e., with just the glow from the LCD on my computer :-), so don't be discouraged if my room looks dark as you approach; I may still be lurking inside (and I will switch the lights on for visitors)!

Class Web Page:

I will be using D2L (http://d2l.pdx.edu; Odin account required for login) as my primary course website. Announcements, class materials (including lecture slides and assignments), discussion forums, etc.

Text:

There is no required textbook, but there are lots of useful resources that are available on the web; we will provide links to appropriate materials in the "Reference Materials" section of Course Content on D2L as the course progresses.

Supporting software:

The primary working environment for this class will be a Linux virtual machine, which can be hosted on Linux, Windows, and Mac OS X computers using VirtualBox. Use of the Unix command line, as well as assembly language and C programming tools is expected.

Proposed Schedule:

The class will meet over ten weeks, as shown in the following table:

Monday	Wednesday	Topic
Mar 28	30	Introduction: course goals and general overview. Assembly language programming.
Apr 4	6	Bare metal programming and tools. The boot process. Basic PC architecture.
11	13	Hardware support for operating systems abstractions. Interrupts, faults, and exceptions.
18	20	Memory management and protected mode.
25	27	Case study 1: The L4 microkernel — Introduction and fundamentals.
May 2	4	Case study 1, continued: Implementation and use.
9	11	Case study 2: The seL4 microkernel — Capability-based access control and resource management.
16	18	Case study 2, continued: Implementation and use. Domain Specific Languages.
23	25	Language Design I.
30	Jun 1	No lecture on Memorial Day (May 30). Language Design II.
Thursday, June 9		Final, 12:30-14:20pm; (as determined by the University guidelines).

There is quite a lot of flexibility in this plan, and I anticipate a number of changes along the way in response to feedback from the class (and other, external constraints). I am happy to accommodate requests for changes in the syllabus, so long as there is a clear consensus.

Method of assessment:

Students will be assessed on the basis of a portfolio comprising the results of lab work and independent projects, and closely aligned with the course objectives listed above. Further details will be presented during the first lecture and adapted/adjusted as necessary during the term if it turns out that any of the listed objectives cannot be properly covered during lectures.

There will not be any midterm or final exams for this class. In accordance with university regulations, attendance during the final session on June 9 is required; this time will be used either for instruction and/or for demonstrations of student projects.

Academic Integrity

We follow the standard guidelines for academic integrity. This is important because a breach of academic integrity by one student undermines the efforts and achievements of the other students in the class who have

made honest and legitimate attempts to study for and complete assignments. It is permissible to *discuss* assignments with other students, but you must *develop* the solution yourself. *Do not, under any circumstances, copy any part of another person's solution and submit it as your own*. Unless you are given explicit written instructions to the contrary, sharing of code or test cases is not permitted. In particular, posting all or part of your solution on D2L or any other public forum, media, or site may be considered a breach of academic integrity. Writing code for use by another, or using another's code in any form (even with their permission) will be considered cheating. Cheating will result in an automatic failing grade for your portfolio, and will trigger the initiation of disciplinary action at the University level. Please refer to http://www.pdx.edu/dos/codeofconduct for details of the general PSU Student Code of Conduct. Any student with questions about academic integrity issues, either relating to their own behavior, or with concerns about the behavior of other students, should contact the instructor. All such matters will be treated in confidence.

Disabilities and Accommodations

We will do everything possible to provide accomodations, with full confidentiality, to any student who needs them. All accomodations must be approved by the Disability Resource Center, which establishes fair and consistent standards across campus. Students with a documented disability who are registered with the Disability Resource Center are responsible for ensuring that their specific requirements are clearly communicated to the instructor at the earliest possible opportunity, either directly or via the DRC. Students should take steps to notify the instructor as soon as possible if they feel that their needs are not being met.
