School of Computer Science & Engineering
University of New South Wales

Advanced Operating Systems

COMP9242 2019/T2

Course Outline

CONTENTS

- Course aims
- Course objectives
- Learning Outcomes
- Course prerequisites
- Course constituents
- Assessment
- Other administrative info
- Text and reference books
- Staffing

COURSE AIMS

To provide students with a deep understanding of modern operating system technology, implementation techniques and research issues.

Our approach to achieving this goal is to expose students to advanced topics in operating systems via interactive lectures that examine specialist topics, and selected research papers and their results. Further, students undertake a substantial practical project where they apply their skills to advanced operating system construction. Together, both components give students an advanced theoretical foundation in operating systems, that is re-enforced through practical application.

This course builds upon the basic operating systems course (COMP3231/9201/3891/9283), which provides an understanding of the underlying operating systems which students have implicitly relied upon in developing applications in foundational courses within Computer Science and Engineering, and will rely on in their future careers when developing systems and applications. Advanced operating systems enables students to specialise in operating systems, giving them the background to become operating systems or embedded-systems developers or researchers, either themselves or as part of a team.

This course contributes to the the students graduate attributes in the following ways:

- The challenging project enables student to further develop the analytical skills required to manage system complexity, creative problem solving, and engages student in collaborative and in-depth application of their operating system skills.
- Regular project demonstrations improve technical communication skills.
- Interactive examination during lectures of relevant research in the field develops critical thinking, and engages students in examples of scholarly enquiry.
- The project also requires students to develop their skills in locating the relevant information required complete the project, evaluate the information's relevance to the project, and when required, digest and apply the information, and self-evaluate their own understanding of the material.

OBJECTIVES

Technical

Provide in-depth coverage of modern operating system issues, such as:

- microkernels and IPC,
- user-level OS servers,

- design and implementation of microkernel-based systems,
- performance,
- kernel design and implementation,
- device drivers,
- virtualisation and hypervisors,
- scheduling for real-time,
- symmetric multiprocessing and hardware multithreading,
- effects and control of hardware caches,
- protection and security models,
- OS designs and resulting issues,
- hot topics.

Educational

- Exposing students to current operating systems research and modern OS technology.
- Providing insight in the design principles of very fast kernels.
- Providing experience in low-level systems programming in a realistic development environment.
- Providing experience in reading and evaluating research papers.
- Encouraging interest in further study and research in the area.

Professional

- Working in an environment and on problems similar to a professional OS or embedded systems engineer in industry.
- Designing, implementing and debugging a whole system almost from the hardware up.
- Learning to cope with system complexity.
- Understanding lowest-level OS code and its interaction with hardware.

LEARNING OUTCOMES

On successfully completing the course, students should be capable of:

- Developing low-level operating system code.
- Understanding the performance trade-offs in developing high-performance low-level OS code. Specifically:
  - The influence of cache architecture on OS design.
  - The influence of cache coherence models and multicore on OS and sync primitive design and implementation.
  - The influence of execution models on complexity and performance of OS code.
- Understanding and be capable of developing OS code inside a variety of OS environments, including monolithic, microkernels, and virtual machines, including device drivers.
- Developing benchmarks to evaluate the performance of OSs and application stacks.
- Understanding and of evaluating research published in the field of operating system at a level commensurate with their experience.

PREREQUISITES

General

- Students are expected to be highly competent in programming in C. Students not familiar with C will be expected to learn it on their own (very quickly!)
- Students are expected to be familiar with assembly language programming. The project will not require assembly programming, but for debugging and understanding of calling conventions this knowledge is required. Furthermore, the lectures will examine low-level kernel code, some of which is written in assembler.
- Students are expected to be familiar with basic computer architecture concepts and the main characteristics of a modern RISC processor. The project will use an Odroid-C2 computer based on the Amlogic S905 SoC containing a quad-core Cortex-A53 64-bit CPU (ARMv8 ISA).

Course Prerequisites

1. a **distinction** grade in COMP3231/COMP3891/COMP9201/COMP9283 (Extended) Operating Systems (pre-requisite)

CONSTITUENTS

Lectures

A rough outline of the lectures is (subject to change):

Introduction and Overview

Introduction to the seL4 Microkernel

seL4 system calls, library API and usage (to get you started on the project)

Microkernels and User-level Servers

History and motivation for microkernel systems, Hydra, Mach, discussion, experiences; second-generation microkernel systems; design and implementation of microkernel-based systems, including user-level page fault handling and device drivers

Microkernel Construction

A detailed look at the design and internals of a real secure and high-performance microkernel.

A close look at selected OS issues

- Caching, and its implications for OS
- Events vs threads
- OS security
- Virtual machines
- OS research projects at UNSW/Data61

Other topics (as time allows)

Candidates are

- Comparison of OS designs (L4, Linux, Darwin, Windows)
- Real-time issues
- SMP/SMT issues: locking, cache coherence, scheduling
- Reliable device drivers
- File systems
- Hot topics

Laboratories/Project

Lab work forms a major component of the course. This will be carried out using a take-home hardware kit that can be used in any of the School's Linux labs, or on students own machines.

In the first lecture, students will be provided with their Odroid-C2 lab kits.

The Odroids run seL4, a third-generation microkernel developed from scratch by Kevin Elphinstone and his team, part of the Trustworthy Systems group at Data61 (formerly NICTA). The kernel is the latest in a series of L4 microkernels designed at Karlsruhe, UNSW and Data61, and has some revolutionary properties which will be disused in class. Most famously, it is the world's first (and so far only) OS kernel with a formal proof of functional correctness; this was done by the ERTOS team under the lead of Gerwin Klein. The ERTOS engineers and research students are mostly alumni of this course.

The Odroids connect to Linux hosts running an L4 development environment. OS code is developed and compiled there and then downloaded to the Odroid, which present a minimum environment ideally suited for low-level systems programming exercises. Documentation as well as sample code will be provided.

After some ``warm-up'' experiments students will work in groups of two on a project, which constructs various OS components, with the ultimate aim of producing a small (and very efficient) operating system. A series of milestones are defined to aid the implementation.

Milestones and the final project will be demonstrated to School staff and the code submitted for assessment. Complete system documentation will form the final deliverable.

Milestones must be demonstrated at a consultation time in the week following the week that the milestone is due. Both partners must be present and participate in milestone demonstrations as milestone demonstrations are assessed individually. Milestone deadlines missed by no more than one week will cause a loss of 20% of the mark for that particular milestone, if missed by more that one week the penalty is 40%, up to a maximum of two weeks. No submissions/demos will be accepted later than two weeks after the deadline. Furthermore, students will not be allowed to continue with the course unless they have given a satisfactory demonstration of the first two milestones (m0 and m1) within one week of the respective due date!

Alternative projects may be given to some students by special arrangements. The main criteria for this is that the project is at least as challenging as the standard project, and that I am convinced that the student(s) are up to it.

Cheating

Misconduct and plagiarism in any form will be dealt with consistent with CSE and University policies (see http://webapps.cse.unsw.edu.au/cse/student-conduct.html).

Final Exam

There will be a final exam, in the form of a *24h take-home*. Students will be given one day to read and analyse **two** recent research papers relevant to the material covered in the course, and submit a critical report on it. See the previous years' exams for examples.

Supplementary assessments

Supplementary exams will only be awarded in well justified cases, in accordance with School policy, **not** as a second chance for poorly performing students. In particular, it is highly unlikely that a supplementary will be awarded to students who have actually submitted an exam. **Make up your mind whether or not you are sick before submitting!**

Supplementary exams will either have the same format as the normal exam, or, at the discretion of the LiC, will be orals. They will probably be held on the day after the written supplementary exams held for other courses.

Consultations

See the consultation schedule.

ASSESSMENT

The project is the heart of this course, which is all about learning about OS design and implementation “hands on”. It will dominate the workload for the course, and will develop valuable and highly-sought-after systems skills in students. Consequently, the project is the dominating assessment component. The project work counts for 65% of the final mark. Students must contribute *equally* to the project work. In the case of significantly unequal contributions, the marks of the lesser contributor will be reduced.

The exam contributes 35% of the final mark. **A minimum mark of 14 (i.e., 40% of the maximum) is required in the exam to receive a passing grade.** This is to ensure that no-one can pass the course on project work alone.

OTHER ADMINSTRATIVE INFO

Surveys

We conduct a detailed on-line survey at the end of each session, in order to identify any shortcomings of the course. This has in the past helped to maintain and improve the quality of the course.

In order to emphasise that we take student comments very seriously, we publish all surveys uncensored, with the possible exception of removing statements other people could find upsetting (the LiC isn't easily insulted ;-). In some cases they are accompanied by comments by the LiC.

Parallel Teaching

N/A

Credit Value

COMP9242 is weighted at 6 UoC.

Yellow Form

Expectations and responsibilities of students, OH&S and equity and diversity issues are covered in the School's Yellow Form which every student needs to sign.

TEXT and REFERENCE BOOKS

Textbook

There is no textbook for this course, as no published book covers the material in sufficient depth. Lecture slides and plenty of handouts will be provided.

Reference Books

- Robert Love: Linux Kernel Development, 3rd edition
- A. Tanenbaum, A. Woodhull: Operating Systems: Design and Implementation, 2nd ed. 1997, Prentice Hall.
- Curt Schimmel: UNIX Systems for Modern Architectures, 1994, Addison Wesley.
- M. Beck, H. Böhme, M. Dziadzka, U. Kunitz, R. Magnus, and D. Verworner: Linux Kernel Internals, 1997, Addison Wesley.
- Marshall K. McKusick, Keith Bostic, Michael J. Karels, John S. Quarterman: The Design and Implementation of the 4.4BSD Operating System, 1996, Addison Wesley.
- Helen Custer: Inside Windows NT, 1993, Microsoft Press.
  2nd version authored by David A. Solomon, (1998), 3rd version authored by David A. Solomon and Mark Russinovich titled ``Inside Windows-2000" (2000).
- Helen Custer: Inside the Windows NT File System, 1994, Microsoft Press.
- Scott Maxwell: Linux Core Kernel Commentary, 1999, CoriolisOpen Press.
- John Lions: Commentary on UNIX 6th edition with source code, 1996, Peer-to-Peer Communications. (The famous Lions Book, identical to the 1977 UNSW TR.)
- Selected research papers as referred to in class.
- Source code.

Reference manuals for labs

The menu bar on the left provides links to all relevant documentation for hardware and software.

Other material

Lecture slides and other information can be found under the course's WWW home page at URL http://www.cse.unsw.edu.au/~cs9242/.

STAFFING

Lecturer-in-charge is Kevin Elphinstone. He and Gernot Heiser will deliver most of the lectures. Some lectures will be delivered by other local researchers, visitors or research students.

Last modified: 31 Mar 2020
CRICOS Provider Number: 00098G