Course Outline

Contents

- Course Details
- Course Summary
- Assumed Knowledge
- Student Learning Outcomes
- Teaching Strategies
- Teaching Rationale
- Student Conduct
- Assessment
- Course Schedule
- Resources for Students
- Course Evaluation and Development

Course Details

Course Code	COMP9242
Course Title	Advanced Operating Systems
Convenor	Kevin Elphinstone (/users/z8990549) , Gernot Heiser (https://webcms3.cse.unsw.edu.au/users/z9100407)
Admin	Kevin Elphinstone (/users/z8990549)
Classes	**Lectures** : Timetable for all classes (/COMP9242/20T2/timetable)
Consultations	Lab Consultations (https://www.cse.unsw.edu.au/~cs9242/20/index.shtml#consult-labs)
Units of Credit	6
Course Website	http://www.cse.unsw.edu.au/~cs9242/current/ (http://cse.unsw.edu.au/~cs9242/20T2/)
Handbook Entry	http://www.handbook.unsw.edu.au/postgraduate/courses/current/COMP9242.html (http://www.handbook.unsw.edu.au/postgraduate/courses/current/COMP9242.html)

Course Summary

The course provides students with a deep understanding of modern operating system technology, implementation techniques and research issues.

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.

Assumed Knowledge

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).
- Students are expected to have a good understanding and be capable of practically applying basic operating systems concepts including processes, threads, concurrency, file systems, memory management, and virtual memory.

Course Prerequisites

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

Student 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 evaluating research published in the field of operating system at a level commensurate with their experience.

This course contributes to the development of the following graduate capabilities:

Graduate Capability	Acquired in
Scholars capable of independent and collaborative enquiry, rigorous in their analysis, critique and reflection, and able to innovate by applying their knowledge and skills to the solution of novel as well as routine problems	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. Interactive examination during lectures of relevant research in the field develops critical thinking, and engages students in examples of scholarly enquiry.

Entrepreneurial leaders capable of initiating and embracing innovation and change, as well as engaging and enabling others to contribute to change	The course, with its focus on self-directed learning and helping students to excel above expectations contributes to developing leadership skills. Many of its alumni fund companies or end up in leading positions in the innovation ecosyste,
Professionals capable of ethical, self-directed practice and independent lifelong learning	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.
Global citizens who are culturally adept and capable of respecting diversity and acting in a socially just and responsible way	The course is built on world-leading open-source technology and encourages and enables students to become players in the world-wide open-source community. It also enables many students with ordinary WAMs to excel, teaching them that what you do is more important than where you come from.

Teaching Strategies

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 access to their Odroid-C2 (https://www.cse.unsw.edu.au/~cs9242/20/project/odroidc2.shtml) lab kits.

The Odroids run seL4 , a third-generation microkernel developed from scratch by Kevin Elphinstone (http://www.cse.unsw.edu.au/~kevine) and his team, part of the Trustworthy Systems (http://trustworthy.systems/) group at Data61 (http://data61.csiro.au/) (formerly NICTA). The kernel is the latest in a series of L4 microkernels designed at Karlsruhe (http://l4ka.org/) , 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 (http://ssrg.nicta.com.au/projects/TS/l4.verified/) ; this was done by the ERTOS team under the lead of Gerwin Klein (http://www.cse.unsw.edu.au/~kleing/) . 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.

Teaching Rationale

Our approach to achieving the learning outcomes 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.

Student Conduct

The **Student Code of Conduct** (Information (https://student.unsw.edu.au/conduct) , Policy (https://www.gs.unsw.edu.au/policy/documents/studentcodepolicy.pdf)) sets out what the University expects from students as members of the UNSW community. As well as the learning, teaching and research environment, the University aims to provide an environment that enables students to achieve their full potential and to provide an experience consistent with the University's values and guiding principles. A condition of enrolment is that students *inform themselves* of the University's rules and policies affecting them, and conduct themselves accordingly.

In particular, students have the responsibility to observe standards of equity and respect in dealing with every member of the University community. This applies to all activities on UNSW premises and all external activities related to study and research. This includes behaviour in person as well as behaviour on social media, for example Facebook groups set up for the purpose of discussing UNSW courses or course work. Behaviour that

is considered in breach of the Student Code Policy as discriminatory, sexually inappropriate, bullying, harassing, invading another's privacy or causing any person to fear for their personal safety is serious misconduct and can lead to severe penalties, including suspension or exclusion from UNSW.

If you have any concerns, you may raise them with your lecturer, or approach the School Ethics Officer (mailto:ethics-officer@cse.unsw.edu.au) , Grievance Officer (mailto:grievance-officer@cse.unsw.edu.au) , or one of the student representatives.

Plagiarism is defined as (https://student.unsw.edu.au/plagiarism) using the words or ideas of others and presenting them as your own. UNSW and CSE treat plagiarism as academic misconduct, which means that it carries penalties as severe as being excluded from further study at UNSW. There are several on-line sources to help you understand what plagiarism is and how it is dealt with at UNSW:

- Plagiarism and Academic Integrity (https://student.unsw.edu.au/plagiarism)
- UNSW Plagiarism Procedure (https://www.gs.unsw.edu.au/policy/documents/plagiarismprocedure.pdf)

Make sure that you read and understand these. Ignorance is not accepted as an excuse for plagiarism. In particular, you are also responsible that your assignment files are not accessible by anyone but you by setting the correct permissions in your CSE directory and code repository, if using. Note also that plagiarism includes paying or asking another person to do a piece of work for you and then submitting it as your own work.

UNSW has an ongoing commitment to fostering a culture of learning informed by academic integrity. All UNSW staff and students have a responsibility to adhere to this principle of academic integrity. Plagiarism undermines academic integrity and is not tolerated at UNSW. Plagiarism at UNSW is defined as using the words or ideas of others and passing them off as your own.

If you haven't done so yet, please take the time to read the full text of

- UNSW's policy regarding academic honesty and plagiarism (https://student.unsw.edu.au/plagiarism)

The pages below describe the policies and procedures in more detail:

- Student Code Policy (https://www.gs.unsw.edu.au/policy/documents/studentcodepolicy.pdf)
- Student Misconduct Procedure (https://www.gs.unsw.edu.au/policy/documents/studentmisconductprocedures.pdf)
- Plagiarism Policy Statement (https://www.gs.unsw.edu.au/policy/documents/plagiarismpolicy.pdf)
- Plagiarism Procedure (https://www.gs.unsw.edu.au/policy/documents/plagiarismprocedure.pdf)

You should also read the following page which describes your rights and responsibilities in the CSE context:

- Essential Advice for CSE Students (https://www.engineering.unsw.edu.au/computer-science-engineering/about-us/organisational-structure/student-services/policies/essential-advice-for-cse-students)

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.

Item	Due	Marks
Project start (individuals)	Week 1	5%

Project (groups)	Weekly milestones	60%
Final Exam	Exam period	35%

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 (https://www.cse.unsw.edu.au/~cs9242/11/exam/) for examples.

Course Schedule

Lectures

The topics covered are indicated in *Teaching Strategies* above. The precise order varies due to the varied availability of guest experts that cover specific material. See the lectures web page (https://www.cse.unsw.edu.au/~cs9242/current/lectures.shtml) for the current schedule.

Project

There is a weekly milestone due as described on the project web page (https://www.cse.unsw.edu.au/~cs9242/current/lectures.shtml) , except for week 6 (flexibility week).

Resources for Students

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 (http://stid.googlecode.com/files/Linux.Kernel.Development.3rd.Edition.pdf) , 3rd edition
- A. Tanenbaum, A. Woodhull: Operating Systems: Design and Implementation, 2 nd 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.
  2 nd version authored by David A. Solomon, (1998), 3 rd 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 6 th 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.

Course Evaluation and Development

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 (https://www.cse.unsw.edu.au/~cs9242/surveys/) , 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.

Resource created about a year ago (Thursday 21 May 2020, 02:06:07 PM), last modified 15 days ago (Tuesday 01 June 2021, 03:16:23 PM).

Comments

(/COMP9242/20T2/forums/search?forum_choice=resource/46957)

(/COMP9242/20T2/forums/resource/46957)

There are no comments yet.