Course Staff

Name	Function	Contact
Kevin Elphinstone	Lecturer in Charge	kevine@cse.unsw.edu.au
Alex Kroh	Course Admin	akroh@cse.unsw.edu.au

Note: Unless the issue is personal in nature, please use Piazza to ask questions instead of emailing course staff directly. This is so answers can benefit others.

Note: Consultation times are available on the the course web page.

Course Information

Units of Credit: 6

Course Delivery

COURSE_CODE are taught in parallel. The lectures are common for all courses.

Course Aims

Operating systems are an essential part of computer systems, a course on operating systems is an essential part of any computer science or computer engineering program. This course exposes students to the essential concepts and issues that underlie operating systems and their design.

In general terms, the course aims to educate students in the basic concepts and components of operating systems, the relevant characteristics of hardware, and the trade-offs between conflicting objectives faced by the operating systems in efficiently supporting a wide range of applications.

This course provides an understanding of the underlying operating systems which students have implicitly relied upon when developing applications in the foundational courses within Computer Science and Engineering. The knowledge gained will continue to be relevant in future careers when developing systems and applications. The course also builds upon COMP2121/9032 *Microprocessors and Interfacing* by illustrating how more complex software systems are supported on microprocessor-based systems.

Specific Aims and Outcomes

Technical

Understand the key concepts and mechanisms of modern operating systems:

- processes and process management, including threads and concurrency management,
- physical and virtual memory management techniques ,
- on-line storage methods (file systems),
- security and protection (optional),

Educational

Understand the reasons why operating systems are built the way they are, and what the implications and lessons are for other software systems. Specific learning objectives are:

- appreciation of design trade-offs and design decisions and their dependence on the target environment;
- appreciation of the distinction between mechanisms and policies, and why this is important;
- exposure to low-level code;
- exposure to current trends in operating systems research and development.

Professional

The tutorial format gives students the opportunity to present and discuss course content to an audience of their peers. Furthermore, the whole course encourages critical examination and analysis of "standard" solutions.

The assignments give students an opportunity to develop the skills required to work as a team on a technical project, and the opportunity to work with a substantial body.

Teaching Approach

Our approach to teaching operating systems is

- introducing operating system theory in the lecture component of the course, including case studies of real systems
- actively engaging students in the practical application of that theory through challenging assignments, including assessment and feedback on their efforts.

- We also seek engagement of students by encouraging dialogue between staff and students both in lectures and tutorials to re-enforce concepts being taught.
- Students also are encouraged to learn collaboratively through group assignments, which also hones their skills as "team players" for future entry into the workforce.

Course Development

We strive to continually improve this course by actively seeking feedback both during session, and via various evaluation surveys (CATEI and my own) at the end of the course. The results of surveys and any feedback are taken seriously and are actively used as input to further improvements in the course. See the survey archive (http://cgi.cse.unsw.edu.au/~cs3231/surveys.html) for what students thought about the course, and how the course is evolving based on their feedback.

Prerequisites

General

It is assumed that the student is familiar with the organisation of a general-purpose computer (in particular, CPU, memory, bus, registers, machine instructions, interrupts/exceptions).

Students are also assumed to be competent in the C programming language. More specifically, students should understand pointers, function pointers, memory allocation (`malloc()`), and be comfortable navigating around an existing code base.

Course Prerequisites

> PHP Notice: Use of undefined constant COURSE_CODE - assumed 'COURSE_CODE' in /tmp_amd/adams/export/adams/1/cs3231/public_html/17s1/intro-content.php on line 129 PHP Notice: Use of undefined constant COURSE_CODE - assumed 'COURSE_CODE' in /tmp_amd/adams/export/adams/1/cs3231/public_html/17s1/intro-content.php on line 147 PHP Notice: Use of undefined constant COURSE_CODE - assumed 'COURSE_CODE' in /tmp_amd/adams/export/adams/1/cs3231/public_html/17s1/intro-content.php on line 154 PHP Notice: Use of undefined constant COURSE_CODE - assumed 'COURSE_CODE' in /tmp_amd/adams/export/adams/1/cs3231/public_html/17s1/intro-content.php on line 162 PHP Notice: Use of undefined constant COURSE_CODE - assumed 'COURSE_CODE' in /tmp_amd/adams/export/adams/1/cs3231/public_html/17s1/intro-content.php on line 229

Course Exclusions

1. COMP9201/9283 Operating Systems (for COMP3231/3891)
2. COMP3231/3891 Operating Systems (for COMP9201/9283)

Constituents

Broadly, the course consists of lectures, tutorials, assignments, and a final exam.

Lectures

The lecture delivery (3 hours per week) and material covered is common for all operating system students, independent of the course code enrolled under.

Lectures will cover roughly the following topics (although not in this order):

- **Introduction and Overview**
  - Introduction to operating systems, computer systems structures, operating system structures
- **Threads**
  - execution abstraction, state, context, preemption, scheduling, synchronisation, deadlocks
- **Process Management**
  - resources, address spaces, multi-threading
- **Storage Management**
  - memory management, virtual memory, file systems

- **Case Studies**

- drawn from UNIX, Linux, Windows and OS/161 will be used throughout the course.

A detailed list of lecture topics and the slides used for the lectures will be posted on the course website as session progresses.

Tutorials

Tutorials start in Week Two. Part of the allocated tutorial time will be devoted towards solving problems and discussing solutions related to lecture topics. The remaining time will be used to discuss topics relevant to assignments.

Students will be required to present tutorial problems under the guidance of the tutor.

Assignments

There will be three assignments (plus a warm up exercise for bonus marks), due approximately in weeks 5,9, and 13. Note that the first assignment is a really trivial familiarisation exercise, and should not be used to judge the difficulty of the other assignments. Except for the first familiarisation assignment, the three main assignments are done in groups of two. Assignments will use the OS/161 educational operating system running on a simulated MIPS R3000 computer called System/161. Both the operating system and the simulator were developed at by the Systems Research at Harvard group. The simulator is relatively platform independent and runs on Unix as well as Windows systems. Assignment solutions will be submitted in C code. Details will be released in due course.

The last 2 assignments have *advanced* components which provide a significantly higher level of challenge than the standard assignments (which most students find fairly challenging already). We want students to do the advanced assignments for the challenge rather than the marks, therefore the opportunity to earn marks is intentionally kept small compared to the amount of extra work required. The marks are limited to 10% of the class mark component of the course, see the Assessment section below for details.

PHP Notice: Use of undefined constant COURSE_CODE - assumed 'COURSE_CODE' in /tmp_amd/adams/export/adams/1/cs3231/public_html/17s1/intro-content.php on line 258

Students have, unfortunately, a tendency to underestimate the time required to do the assignments. Experience from the last few years shows that the assignments in this course are considered challenging, and they consume a fair amount of time. It is important to start early. If you start on them on the weekend before the deadline you'll almost certainly miss the deadline. As a tip, the majority of the assignment is learning how to solve the problem (approx. 75% of the effort). The programming component is actually the minority of the work required (approx 25%).

In order to encourage students to start early, we are also offering 10% bonus marks (10% of the raw mark awarded) for submitting *their final solution one week before the deadline* (i.e., the bonus marks are lost if the assignment is re-submitted less than a week before the deadline). Such bonus marks will allow students to make up for missed marks in the same assignment (i.e., one cannot get above the full marks for the assignment).

Penalty for late submission of assignments will be 4% (of the worth of the assignment) subtracted from the raw mark per day of being late. In other words, earned marks will be lost. For example, assume an assignment worth 25 marks is marked as 20, but had been submitted two days late. *The late penalty will be 2 marks, resulting in a mark of 18 being awarded.* **No assignments will be accepted later than one week after the deadline.** The late penalty is purposely lenient due to the tendency of students to underestimate the work required.

- Assignments will be marked as a group, i.e. groups of one will be marked the same as groups of two or three.
- All group members are expected to contribute equally to EACH assignment.. In submitting an assignment, the group is agreeing that all members have contributed equally unless we receive an agreed-to statement from all group members identifying differing levels of contribution.

Academic Honesty and Plagiarism

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

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

- Student Code Policy
- Plagiarism Policy Statement
- Plagiarism Procedure
- Student Misconduct Procedure

Mid-Session Quiz

There will be **no** mid-session quiz.

Final Exam

A two hour written final examination will be conducted. Note: Extended and normal OS have different exams with approximately 75% common content.

Supplementary assessments

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

Assessment

The final assessment is based on a "class mark" (C) and a final exam (E). The class mark is based on the assignment work and any bonuses accrued during the semester.

- For COMP3231/9201 the class mark is formed as follows: `(asst1 + asst2 + asst3)/9*10 + warm_bonus + min(10, adv_bonus)`
  - Note: the assignments are out of 90 (`asst1 + asst2 + asst3`) and scaled to a mark out of 100.
  -
- For COMP3891/9283 the class mark is formed as follows: `(asst1 + asst2 + asst3 + asst2_adv + asst3_adv) + warm_bonus + min(10,adv_bonus)`
  - `(asst1 + asst2 + asst3 + asst2_adv + asst3_adv)` is out of 100.

As described above, any early bonuses contribute to the applicable individual assignment (which is capped at max marks for that assignment).

(Note that the class mark is capped at 100, irrespective of how many bonus marks you have accumulated.) The exam will be the second component of the final assessment. The weighted harmonic mean of the ``class mark'', $\boldsymbol{C}$, and the final exam mark, $\boldsymbol{E}$, is used to determine the final mark, $\boldsymbol{M}$, according a 40/60 weighted harmonic mean (40% class mark, 60% exam) as shown below.

$$\boldsymbol{M} = \frac{5\,\boldsymbol{E}\,\boldsymbol{C}}{\boldsymbol{2E} + 3\boldsymbol{C}}$$

A final mark of 50% is required in order to pass.

We reserve the right to moderate mark components of the individual courses (COMP3231/9201/3891/9283) separately where appropriate.

More information on harmonic means in general is available here.

TEXT and REFERENCE BOOKS

Textbook

- A. Tannenbaum, *Modern Operating Systems*, 3rd ed., Prentice Hall, 2008

Reference Books

For Operating Systems:

- A. Silberschatz, P.B. Galvin and Greg Gagne, *Operating System Concepts*, 7th ed. (2004)
- William Stallings, *Operating Systems: Internals and Design Principles,* 5th edition, 2005, Prentice Hall.
- A. Tannenbaum, A. Woodhull, *Operating Systems--Design and Implementation*,3rd ed., Prentice Hall, 2006
- John O'Gorman, *Operating Systems*, MacMillan, 2000
- Uresh Vahalla, *UNIX Internals: The New Frontiers*, Prentice Hall, 1996

For the C language:

- B Kernighan and D. Ritchie, *The C Programming Language*, 2nd ed, Prentice Hall.
- S. Harbison and G. Steele, *C: A Reference Manual*, Prentice Hall.

Copies of lecture slides and other information can be found under the course's WWW home page at URL http://www.cse.unsw.edu.au/~cs3231/.