Welcome to OS @ UNSW

COMP3231/9201/3891/9283
(Extended) Operating Systems
Dr. Kevin Elphinstone

System Software Structure

- Compiled C Code
- System Libraries
- System Calls
- Operating System
- Hardware

Major OS Topics

- Processes and Threads
- Concurrency and Deadlock
- File Systems
- Memory and Virtual Memory Management
- Multiprocessors
- I/O
- Scheduling

Why Learn Operating Systems?

- Understand the whole software stack
- Develop OS code
- Develop concurrent code
- Application performance
  - Understand operating system behaviour and how best to interface with it.
  - Diagnose system performance issues.

How will we learn about Operating Systems?

- Lectures
  - Introduce OS theory and case studies
- Tutorials
  - Re-enforce theory
  - Provide guidance on the assignments
- Assignments
  - Opportunity to write real OS code
  - OS/161 is a simplified UNIX-clone intended for teaching
  - Consist of the following
    - Warm-up exercise
    - Concurrency and synchronisation
    - OS Structure involving system calls and file system
    - Memory management
intended schedule

* Lectures
  - Weeks 1-5, 7-10
  - Tutorials
  - Weeks 2-5, 7-10
  - Assignments
  - Assignment 1: Week 2
  - Assignment 2: Week 4
  - Assignment 3: Week 7
  - Assignment 4: Week 10

* Subject to change

Overview of Course Outline

Pre-requisites

• COMPXXXX Data structures and algorithms
  • Stacks, queues, hash tables, lists, trees, heaps,....
• COMPXXXX Microprocessor and Interfacing or Computer Systems Fundamentals
  • Assembly programming
  • Mapping of high-level procedural language to assembly language
  • Interrupts

Assumed Knowledge

• Computing Theory and Background
  • Basic computer architecture
  • CPUs, memory, buses, registers, machine instructions, interrupts/exceptions
  • Common CS algorithms and data structures
  • Linked lists, arrays, hashing, trees, sorting, searching...
  • Ability to read assembly language
  • Exposure to programming using low-level systems calls (e.g. reading and writing files)
• Practical computing background
  • Capable UNIX command line users
  • Familiar with the git revision control system
  • Competent C programmers
  • Understand pointers, pointer arithmetic, function pointers, memory allocation (malloc())
  • The dominant language for OS (and embedded systems) implementation.
  • Comfortable navigating around a large-ish existing code base.
  • Able to debug an implementation.

Why does this fail?

```c
void set(int *x)
{
    *x = 1;
}
void thingy()
{
    int *a;
    set(a);
    printf("%d\n",*a);
}
```

Operating System Coding

Why does this fail?

```c
void set(int *x)
{
    *x = 1;
}
void thingy()
{
    int a;
    set(&a);
    printf("%d\n",a);
}
```
Lectures

• Common for all courses (3231/3891/9201/9283)
• The lecture notes and video will be available on the course website
  • http://www.cse.unsw.edu.au/~cs3231
• Video will be made available in sync with the lecture timetable
• Slide numbers available for Q&A

Tutorials

• Start in week 2
• Attendance is strongly recommended
  • but not marked.
• Tutorial questions cover a broad range of examples
  • Answers available online the week after.
• Use the tutorial to focus where needed
  • Review the questions beforehand
  • We’ll experiment with prioritising with online polls or similar

Assignments

• Assignments form a substantial component of your assessment.
• They are challenging!!!!
  • Because operating systems are challenging
• We will be using OS/161,
  • an educational operating system
  • developed by the Systems Group At Harvard
  • With local changes.
• It contains roughly 20,000 lines of code and comments
• Comments are part of the documentation

Assignments Submission Times

16% late
Assignments

- Late penalty
  - 4% of total assignment value per day
    - You get 18, and are 2 days late
    - Final mark = 18 - (0.04 * 2) = 16 (16.4)
  - Assignments are only accepted up to one week late. >5 days = 0

Assignments

- Warmup assignment (ASST0)
  - Done individually
  - Available NOW!!!!
  - ASST2 and ASST3 are in pairs
    - Info on how to pair up available soon
  - Additionally, advanced versions of the assignment 2 & 3
    - Available bonus marks are small compared to amount of effort required.
    - Student should do it for the challenge, not the marks.
    - Attempting the advanced component is not a valid excuse for failure to complete the normal component of the assignment

Assignments

Assignment 0

- Warm-up exercise due in week 2
  - It’s a warm-up to have you familiarize yourself with the environment and easy marks.
  - Practice with git revision control
  - Practice submitting a solution
  - Practice using code browser/editor
  - Do not use it as a gauge for judging the difficulty of the following assignments.

Assignments

Assignment Due

- ASST0 Week 2
- ASST1 Week 4
- ASST2 Week 7
- ASST3 Week 10

Assignments

Assignment 0

- Warm-up exercise due in week 2
  - It’s a warm-up to have you familiarize yourself with the environment and easy marks.
  - Practice with git revision control
  - Practice submitting a solution
  - Practice using code browser/editor
  - Do not use it as a gauge for judging the difficulty of the following assignments.

Assignments

Assignment 0

Group Work Policy

- Groups of two
- Group members do not have to be in the same tutorial
- Group assignments will be marked as a group
  - Including ‘groups’ of one.
- Group members are expected to contribute equally to each assignment.
  - No “I’ll do the 2nd if you do the 3rd assignment”
  - We accept statements of unequal contributions and do adjust marks of the lesser contributor down.
- Submissions are required to have significant contributions attributable to individual group members.
  - E.g. verifiable using the git revision control system
Plagiarism

- We take cheating seriously!!!
- We systematically check for plagiarised code
  - Penalties are generally enough to make it difficult to pass
- We can google as easy as you can
  - Some solutions are wrong
  - Some are greater scope than required at UNSW
    - You do more than required
    - Makes your assignment stick out as a potential plagiarism case
  - We do vary UNSW requirements

Exams

- There is NO mid-session
- The final written exam is 2 hours
- Supplementary exam are available according to UNSW & school policy, not as a second chance.
  - Medical or other special consideration only

Assessment*

- Exam Mark Component
  - Max mark of 100
  - Based solely on the final exam

- Class Mark Component
  - Max mark of 100
  - 100% Assignments

* Course outline is authoritative.

Assessment

- The final assessment is a weighted geometric mean of 60% exam (E) and 40% class (C) component.

\[ M = e^{\frac{60 \ln E + 40 \ln C}{100}} \]

- Additionally, minimum of 40 required in exam (E) and class (C) components to pass.

Textbook

References

- A. Silberschatz and P.B. Galvin, Operating System Concepts, 5th, 6th, or 7th edition, Addison Wesley
- Uresh Vahalla, UNIX Internals: The New Frontiers, Prentice Hall, 1996
- McKusick et al., The Design and Implementation of the 4.4 BSD Operating System, Addison Wesley, 1996

Ed Forums

- Forum for Q/A about assignments and course
  - Ask questions there for the benefit of everybody
  - Share your knowledge for the benefit of your peers
  - Look there before asking

  - https://edstem.org/
    - Longer link on class web page
    - You will have received an invite from them to your UNSW email address.
    - z8888888@unsw.edu.au
    - Please join and contribute.

Consultations/Questions

- Questions should be directed to the forum.
- Admin and Personal queries can be directed to the class account cs3231@cse.unsw.edu.au
- Don’t PM me in Piazza
- We reserve the right to ignore email sent directly to us (including tutors) if it should have been directed to the forum.
- Consultation Times
  - See course web site.
  - Must email (cs3231@cse) at least an hour in advance and show up on time.
  - If we get at least one email, we’ll run the consult.

What next?

https://wiki.cse.unsw.edu.au/cs3231cgi/Checklist