

# Welcome to OS @ UNSW

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



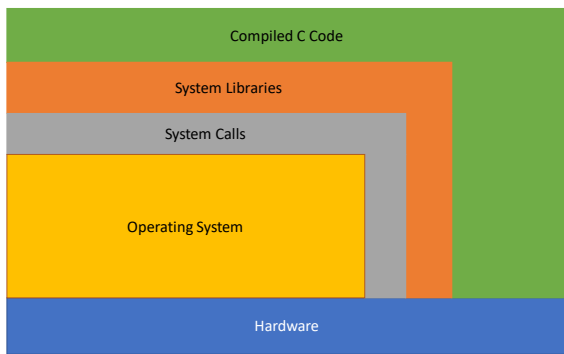
1

## Questions

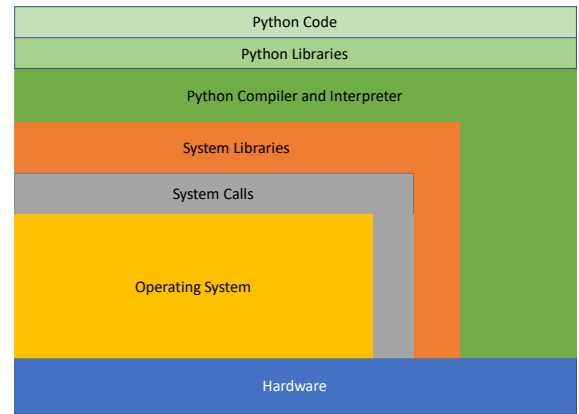
- Ask any questions you have in the course forum before the first lecture.
- I'll answer either on the forum or in the first lecture.

2

## System Software Structure

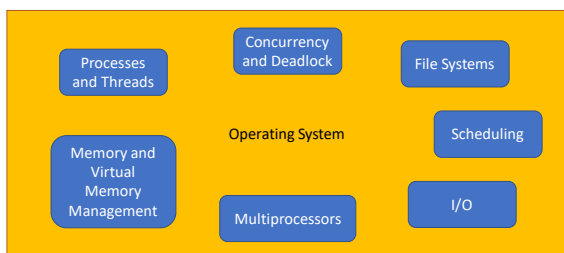


3



4

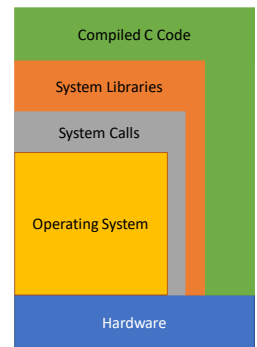
## Major OS Topics



5

## 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.



6


## 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




7

## Intended schedule\*




- Lectures
  - Weeks 1-5, 7-9
- Tutorials
  - Weeks 2-5, 7-10
- Assignments Due
  - ASST0 – Week 2
  - ASST1 – Week 4
  - ASST2 – Week 7
  - ASST3 – Week 10

\* Subject to change



8


## Overview of Course Outline



9

## Prerequisites


- Data structures and algorithms
  - COMP2521, COMP9024 or COMP1927
  - Stacks, queues, hash tables, lists, trees, heaps,....
- Computer systems
  - COMP1521, DPST1092, COMP2121, COMP9032 or ELEC2142
  - Computer systems architecture
  - Assembly programming
  - Mapping of high-level procedural language to assembly language
  - Interrupts



10

## Assumed Knowledge

- Computing Theory and Background
  - Basic computer architecture
    - CPUs, memory, buses, registers, machine instructions, interrupts/exceptions.
  - Common CS algorithms and data structures
    - Links 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.



11



## Why does this fail?

```

void set(int *x)
{
    *x = 1;
}

void thingy ()
{
    int *a;
    set(a);
    printf("%d\n", *a);
}

```

12

## Why does this fail?

```
void set(int *x)
{
    *x = 1;
}

void thingy ()
{
    int a;
    set(&a);
    printf("%d\n", a);
}
```



13

## Lectures

- Common for all courses (3231/3891/9201/9283)
- 2 \* 2 hrs each week
- The lecture notes will be available on the course web site
  - <http://www.cse.unsw.edu.au/~cs3231>
  - Available prior to lectures, when possible.
  - Slide numbers for note taking, when not.
- Lectures will be a mix of live streaming and pre-recorded
  - Will announce in advance
  - Video will be available afterwards in both cases



14

Week	Topic	Book Ref	Print	Download
1	Course Introduction		Print	Download
	Operating Systems Overview	1	Print	Download
	Processes And Threads	2-2.2	Print	Download
	Concurrency and Synchronisation	2.3-2.3.7, 2.5	Print	Download
2	Deadlock	6 - 6.7	Print	Download
	Process and Thread Implementation	2.2 - 2.2.5	Print	Download
3	System Calls and RISC00 Overview	1.6	Print	Download
	Computer Hardware, Memory Hierarchy, and Caching	1.3	Print	Download
4	File Management	4	Print	Download
	File Management Part 2	4	Print	Download
	File Management (continued)		Print	Download
5	Case Study: Ext2		Print	Download
	Case study: Ext1		Print	Download
	Memory Management	3	Print	Download
6	ASST2 Overview Video		Print	Download
	Flexibility Week		Print	Download



15

## Extended OS Comp3891/9283

Starts in week 1

- A combination of:
  - Examination of topics in more depth
  - Looking at research in areas (past/present)
  - OS/161 internals in more depth
- Separate Assessment
  - 80%-ish of final exam common with base course
  - 20%-ish targeted to extended students
  - **Advanced assignment components part of the assessment**
- Assumes the tutorials are not challenging enough
  - Effectively replaces the tutorial with extra interactive lecture.



16

## Tutorials

- **Start in week 2**
- A mix of online and f2f
  - Depends on tutorial you enrolled in
- 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
    - There is intentionally more questions than can be covered
    - Review the questions beforehand



17

## 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



18

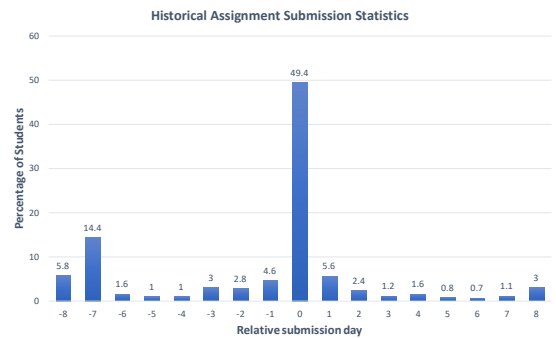
## Assignments

- Don't underestimate the time needed to do the assignments.
  - 80% is understanding
  - 20% programming
- Avoid
  - 1% understanding
  - 9% programming
  - 90% debugging
- If you start a couple days before they are due, you will be late.
- To encourage you to start early,
  - Bonus 2% of awarded mark per day early, capped at 10%
  - See course outline for exact details
    - Read the fine print!!!!



19

## Assignment Submission Times 16% late



20

## Assignments

- Late penalty
  - 4% of total assignment value per day
    - Assignment is worth 20%
    - You get 18, and are 2 days late
    - Final mark =  $18 - (20 * 0.04 * 2) = 16$  (16.4)
- Assignments are only accepted up to one week late. >5 days = 0



21

## 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

Assignment	Due
ASST0	Week 2
ASST1	Week 4
ASST2	Week 7
ASST3	Week 10



22

## 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.



23

## Assignments

- Submission test failed. Continue with submission (y/n)? y
- Lazy/careless submitter penalty: 15%
  - Submitted the wrong assignment version penalty: 15%
    - Assuming we can validly date the intended version



24

## Assignments

- To help you with the assignments
  - We dedicate a tutorial per-assignment to discuss issues related to the assignment
  - Prepare for them!!!!



25

## 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 2<sup>nd</sup> if you do the 3<sup>rd</sup> 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



26

## 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



27

## 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



28

## Assessment\*

- |   |  |
|---|--|
| • Exam Mark Component <ul style="list-style-type: none"><li>• Max mark of 100</li></ul> | • Class Mark Component <ul style="list-style-type: none"><li>• Max mark of 100</li></ul> |
| • Based solely on the final exam  | • 100% Assignments   |

\* Course outline is authoritative.



29

## 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.



30



## 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](mailto:cs3231@cse.unsw.edu.au)
  - Don't post private threads in Ed
- 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.



37

## What next?

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



### Startup Checklist

- Watch the online intro lecture
  - Bring any questions to the first lecture.
- Join Piazza (you should have received an invite sent to [z1D@unsw.edu.au](mailto:z1D@unsw.edu.au))
- Review assignment 0
- Choose where you plan to do your assignment work (desktop, laptop, and at CSE).
  - Make sure the toolchain works on where you plan to work (see [Setup Overview](#))
- Set up git (see [Setup Overview](#))
- Choose an editor capable of code browsing (see [Setup Overview](#)).
- Complete ASSTO



38