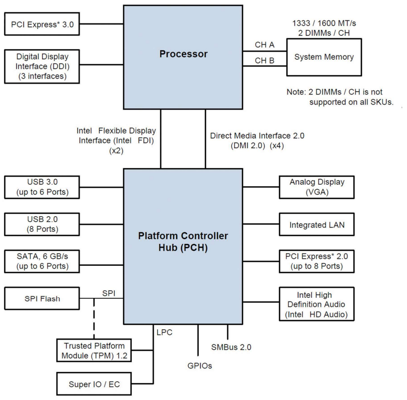
Welcome to OS @ UNSW

COMP3231/9201/3891/9283

(Extended) Operating Systems

Dr. Kevin Elphinstone

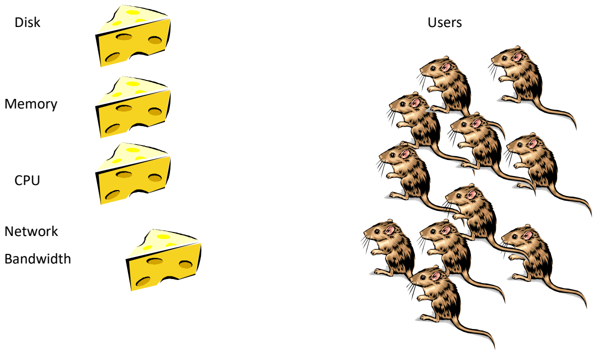
What is an Operating System?


PCI Express* 3.0
Digital Display Interface (DDI) (3 interfaces)
Processor
1333 / 1600 MT/s 2 DIMMs / CH
CH A
CH B
System Memory
Note: 2 DIMMs / CH is not supported on all SKUs.
Intel Flexible Display Interface (Intel FDI) (x2)
Direct Media Interface 2.0 (DMI 2.0) (x4)
USB 3.0 (up to 6 Ports)
USB 2.0 (8 Ports)
SATA, 6 GB/s (up to 6 Ports)
SPI Flash
SPI
Platform Controller Hub (PCH)
Analog Display (VGA)
Integrated LAN
PCI Express* 2.0 (up to 8 Ports)
Intel High Definition Audio (Intel HD Audio)
Trusted Platform Module (TPM) 1.2
LPC
SMBus 2.0
GPIOs
Super IO / EC

Block Diagram of Haswell Platform Architecture http://www.pcquest.com

Role 1: The Operating System is an Abstract Machine

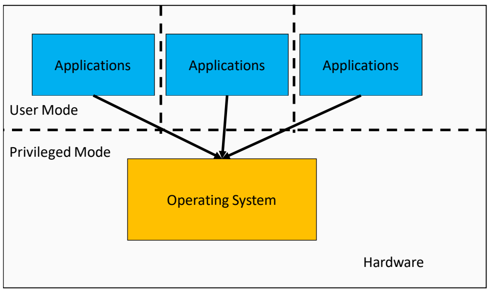
- Extends the basic hardware with added functionality
- Provides high-level abstractions
  - More programmer friendly
  - Common core for all applications
    - E.g. Filesystem instead of just registers on a disk controller
- It hides the details of the hardware
  - Makes application code portable



Role 2: The Operating System is a Resource Manager

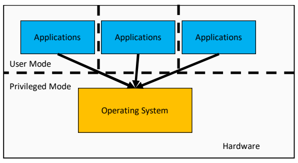
- Responsible for allocating resources to users and processes
- Must ensure
  - No Starvation
  - Progress
  - Allocation is according to some desired policy
    - First-come, first-served; Fair share; Weighted fair share; limits (quotas), etc…
  - Overall, that the system is efficiently used

Structural (Implementation) View: the Operating System is the software *Privileged* mode.



Course Aim

- A deep understanding of the key concepts and mechanisms of modern operating systems:
  - processes and process management, including threads and concurrency management,
  - physical and virtual memory management,
  - on-line storage methods (file systems)



Course Approach

- Operating system background and theory in the lectures
- Practical application of theory through challenging assignments
  - Implementing functionality in a rudimentary OS (OS/161)
  - Challenging as OSes are large and complex
- Tutorials to re-enforce concepts being taught and provide support for assignments
- Learn collaboratively through group assignments (the last 2 assignments)

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 experience
  - Capable UNIX command line users
  - Familiar with the git revision control system
  - Competent C programmers
    - Understand pointers, function pointers, memory allocation (malloc())
  - Comfortable navigating around an existing code base.
  - Able to debug an implementation.

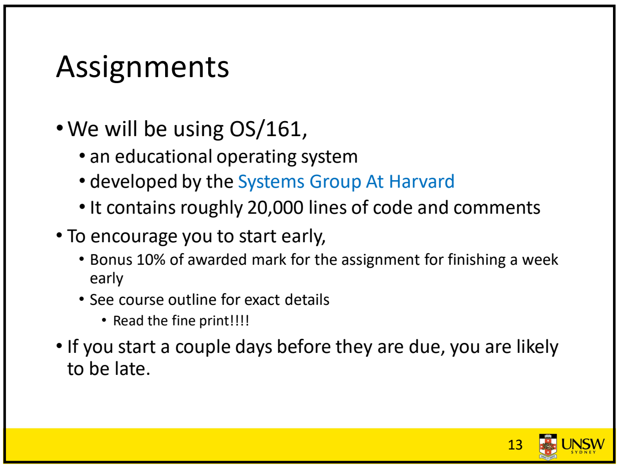
Why does this fail?

```
void set(int *x)
{
    *x = 1;
}
void main()
{
    int *a;
    set(a);
    printf(“%d\n”,*a);
}
```

Operating System Coding

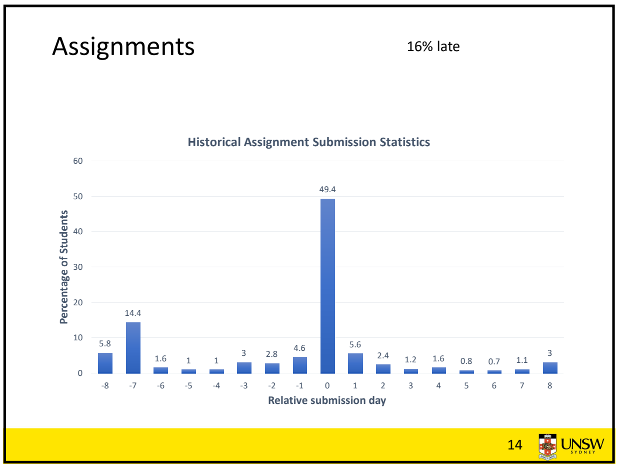
Why does this fail?

```
void set(int *x)
{
    *x = 1;
}
void main()
{
    int a;
    set(&a);
    printf(“%d\n”,a);
}
```



Assignments

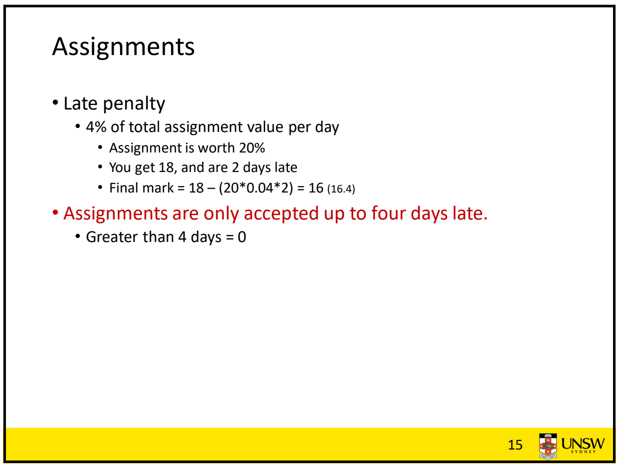
- We will be using OS/161,
  - an educational operating system
  - developed by the Systems Group At Harvard
  - It contains roughly 20,000 lines of code and comments
- To encourage you to start early,
  - Bonus 10% of awarded mark for the assignment for finishing a week early
  - See course outline for exact details
    - Read the fine print!!!!
- If you start a couple days before they are due, you are likely to be late.



Assignments

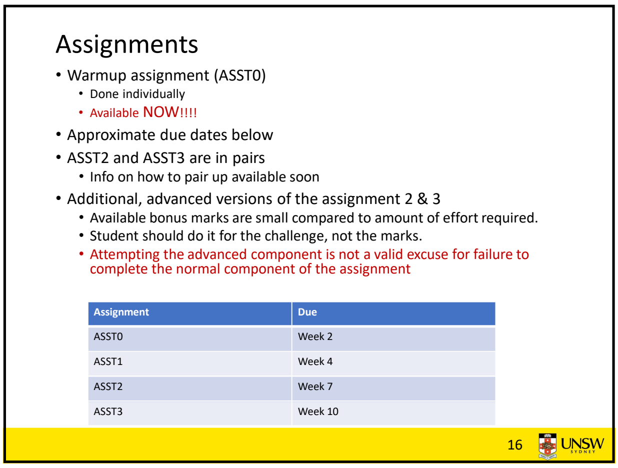
16% late

Historical Assignment Submission Statistics



Assignments

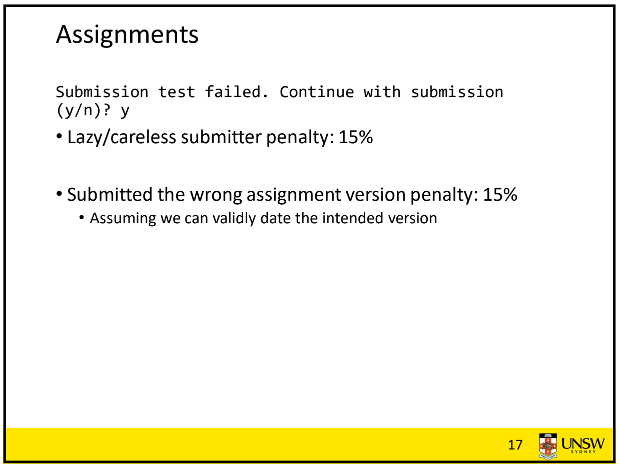
- Late penalty
  - 4% of total assignment value per day
    - Assignment is worth 20%
    - You get 18, and are 2 days late
    - Final mark = 18 – (20*0.04*2) = 16 (16.4)
- Assignments are only accepted up to four days late.
  - Greater than 4 days = 0



Assignments

- Warmup assignment (ASST0)
  - Done individually
  - Available NOW!!!!
- Approximate due dates below
- ASST2 and ASST3 are in pairs
  - Info on how to pair up available soon
- Additional, 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

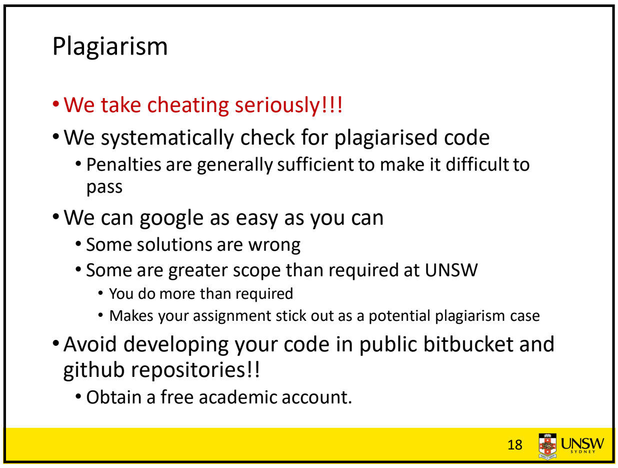
Assignment	Due
ASST0	Week 2
ASST1	Week 4
ASST2	Week 7
ASST3	Week 10



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



Plagiarism

- We take cheating seriously!!!
- We systematically check for plagiarised code
  - Penalties are generally sufficient 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
- Avoid developing your code in public bitbucket and github repositories!!
  - Obtain a free academic account.

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

Piazza 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
  - Apps for phone
- https://piazza.com/
  - Longer link on class web page
    - You will have received an invite from them to your UNSW email address.
    - 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
- 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.

Back to Operating Systems

Chapter 1 – 1.3

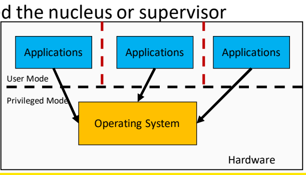
Chapter 1.5 – 1.9

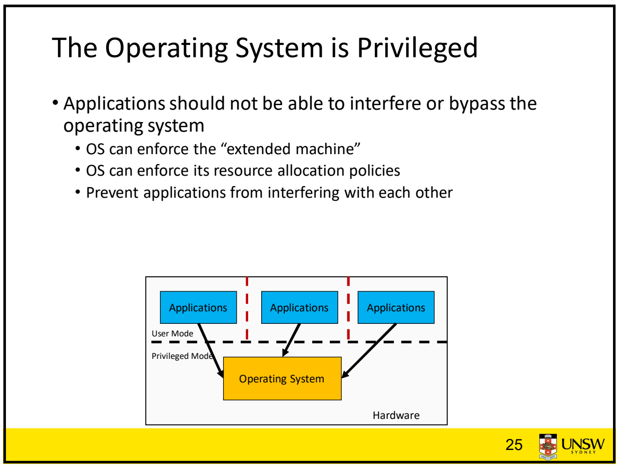
Learning Outcomes

- High-level understand what is an operating system and the role it plays
- A high-level understanding of the structure of operating systems, applications, and the relationship between them.

Operating System Kernel

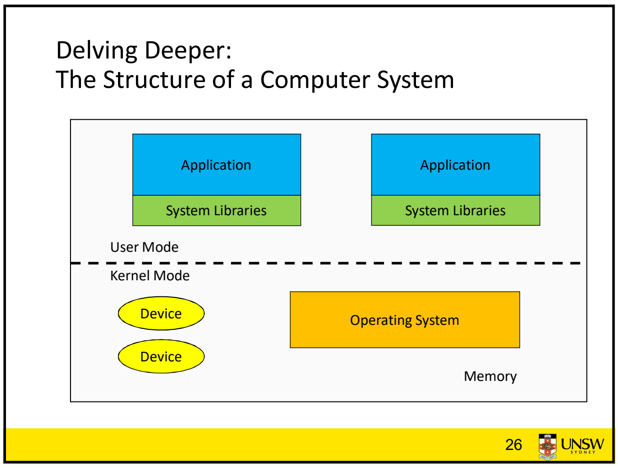
- Portion of the operating system that is running in *privileged mode*
- Usually resident (stays) in main memory
- Contains fundamental functionality
  - Whatever is required to implement other services
  - Whatever is required to provide security
- Contains most-frequently used functions
- Also called the nucleus or supervisor



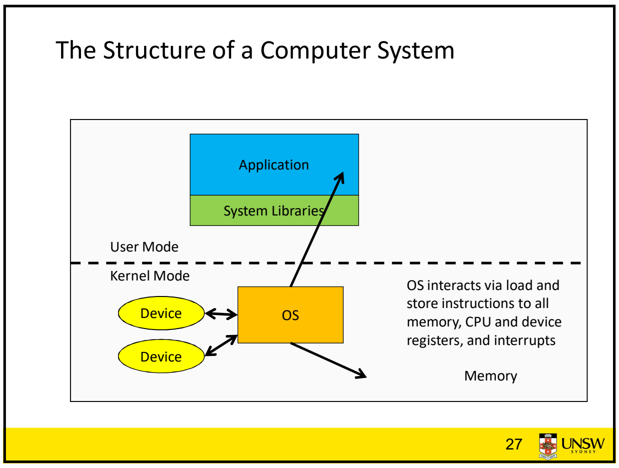


The Operating System is Privileged

- Applications should not be able to interfere or bypass the operating system
  - OS can enforce the "extended machine"
  - OS can enforce its resource allocation policies
  - Prevent applications from interfering with each other

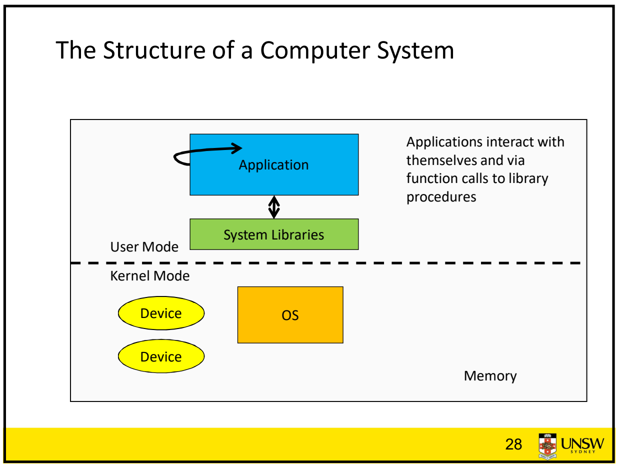


Delving Deeper: The Structure of a Computer System



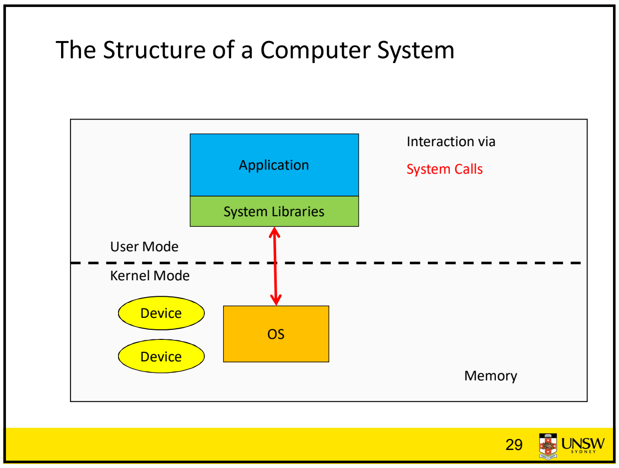
The Structure of a Computer System

OS interacts via load and store instructions to all memory, CPU and device registers, and interrupts



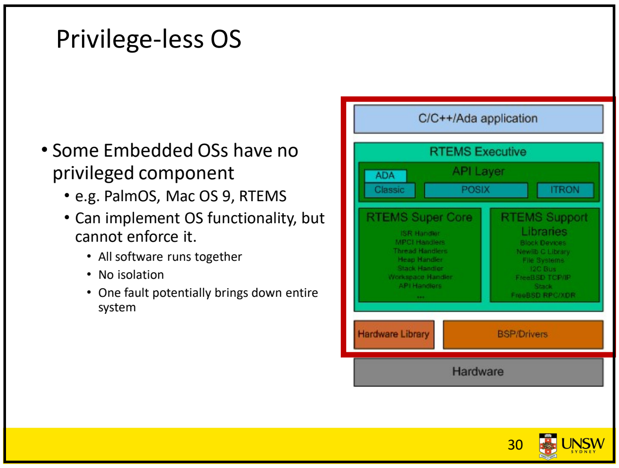
The Structure of a Computer System

Applications interact with themselves and via function calls to library procedures



The Structure of a Computer System

Interaction via System Calls



Privilege-less OS

- Some Embedded OSs have no privileged component
  - e.g. PalmOS, Mac OS 9, RTEMS
  - Can implement OS functionality, but cannot enforce it.
    - All software runs together
    - No isolation
    - One fault potentially brings down entire system

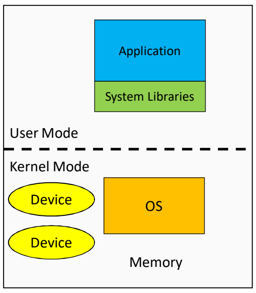
A Note on System Libraries

System libraries are just that, libraries of support functions (procedures, subroutines)

- Only a subset of library functions are actually systems calls
  - strcmp(), memcpy(), are pure library functions
    - manipulate memory within the application, or perform computation
  - open(), close(), read(), write() are system calls
    - they cross the user-kernel boundary, e.g. to read from disk device
    - Implementation mainly focused on passing request to OS and returning result to application
- System call functions are in the library for convenience
  - try `man syscalls` on Linux

Operating System Software

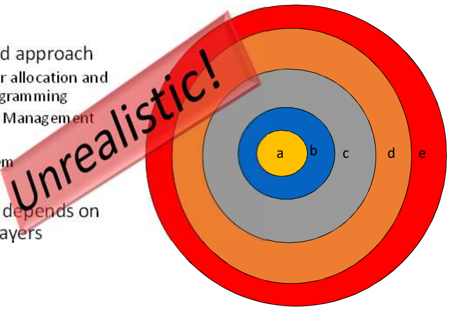
- Fundamentally, OS functions the same way as ordinary computer software
  - It is a program that is executed (just like applications)
  - It has more privileges
- Operating system relinquishes control of the processor to execute other programs
  - Reestablishes control after
    - System calls
    - Interrupts (especially timer interrupts)



Operating System Internal Structure?

Classic Operating System Structure

- The layered approach
  a) Processor allocation and multiprogramming
  b) Memory Management
  c) Devices
  d) File system
  e) Users
  - Each layer depends on the inner layers

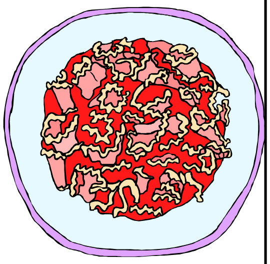


Operating System Structure

- In practice, layering is only a guide
  - Operating Systems have many interdependencies
    - Scheduling on virtual memory
    - Virtual memory (VM) on I/O to disk
    - VM on files (page to file)
    - Files on VM (memory mapped files)
    - And many more…

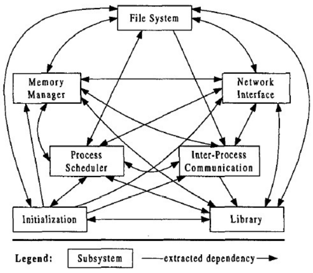
The Monolithic Operating System Structure

- Also called the “spaghetti nest” approach
  - Everything is tangled up with everything else.
- Linux, Windows, ….

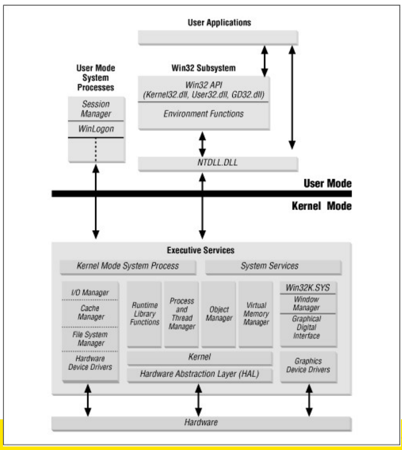


The Monolithic Operating System Structure

- However, some reasonable structure usually prevails



Bowman, I. T., Holt, R. C., and Brewster, N. V. 1999. Linux as a case study: its extracted software architecture. In *Proceedings of the 21st international Conference on Software Engineering* (Los Angeles, California, United States, May 16 - 22, 1999). ICSE '99. ACM, New York, NY, 555-563. DOI= http://doi.acm.org/10.1145/302405.302691



The end