Welcome to OS @ UNSW

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

1

Q & A

2

Back to Operating Systems

Chapter 1 – 1.3
Chapter 1.5 – 1.9

3

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.

4

What is an Operating System?

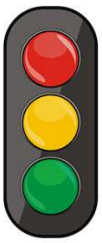




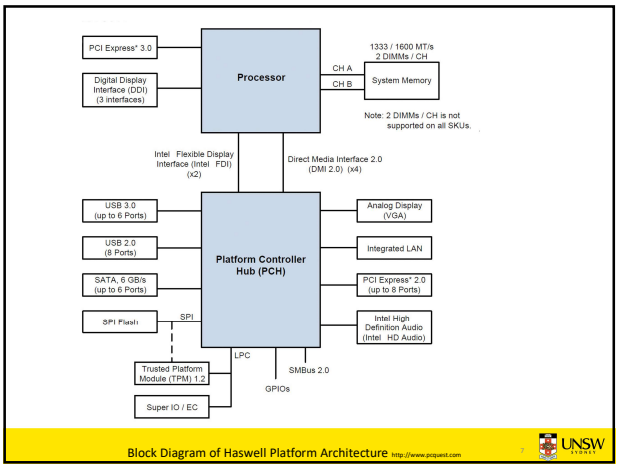
5

What is a traffic light?

- A signalling device that controls the flow of traffic
  - Defined in terms of the **role** it plays
- A signalling device consisting of three lights mounted at an intersection
  - Defined in terms of what it **is**



6

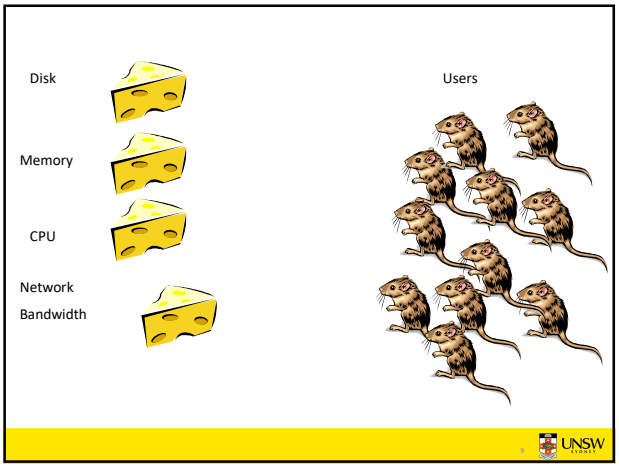


7

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

8



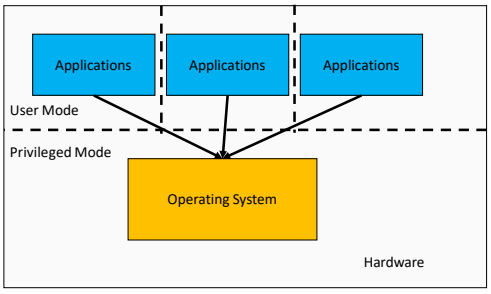
9

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

10

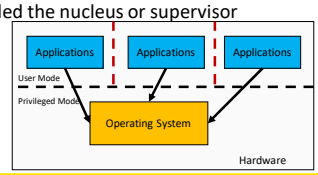
Structural (Implementation) View: the Operating System ***is*** the software *Privileged* mode.



11

Operating System Kernel

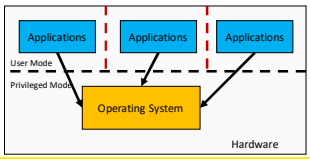
- Portion of the operating system that is running in *privileged mode*
- Usually resident (stays) in main memory
- Contains fundamental functionality
  - Whatever is required to implement other services
  - Whatever is required to provide security
- Contains most-frequently used functions
- Also called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



12

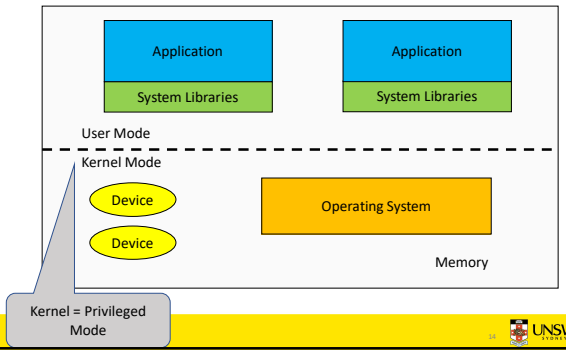
The Operating System is Privileged

- 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



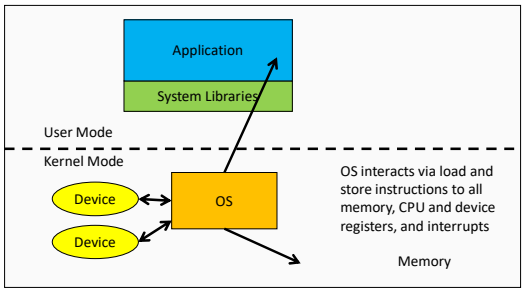
13

Delving Deeper: The Structure of a Computer System



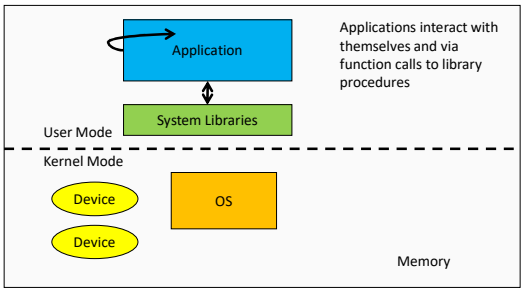
14

The Structure of a Computer System



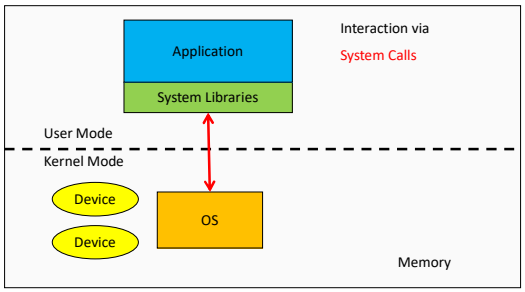
15

The Structure of a Computer System



16

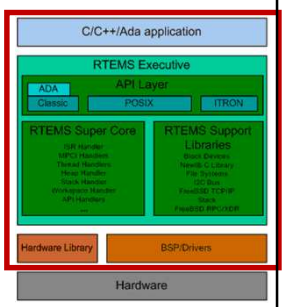
The Structure of a Computer System



17

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



18

A Note on System Libraries

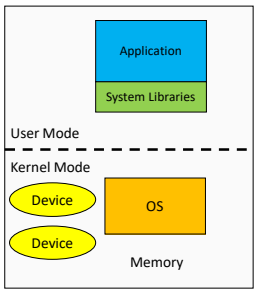
System libraries are just that, libraries of support functions (procedures, subroutines)

- Only a subset of library functions are actually system 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

19

Operating System Software

- Fundamentally, OS functions the same way as ordinary computer software
  - It is machine code that is executed (same machine instructions as application)
  - It has more privileges (extra instructions and access)
- Operating system relinquishes control of the processor to execute other programs
  - Reestablishes control after
    - System calls
    - Interrupts (especially timer interrupts)



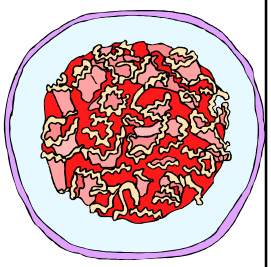
20

Operating System Internal Structure?

21

The Monolithic Operating System Structure

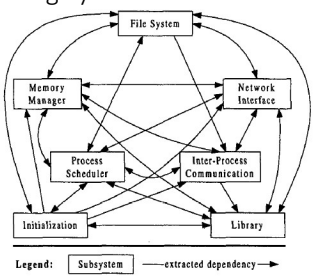
- Also called the "spaghetti nest" approach
  - Everything is tangled up with everything else.
- Linux, Windows, ….



22

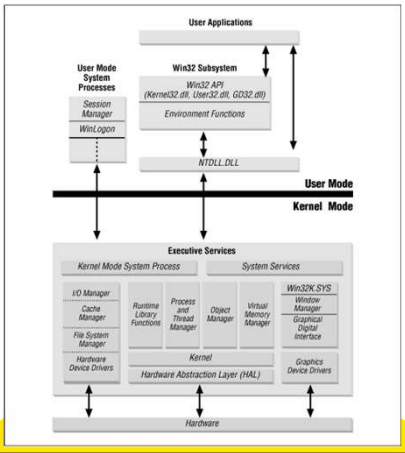
The Monolithic Operating System Structure

- However, some reasonable structure usually prevails

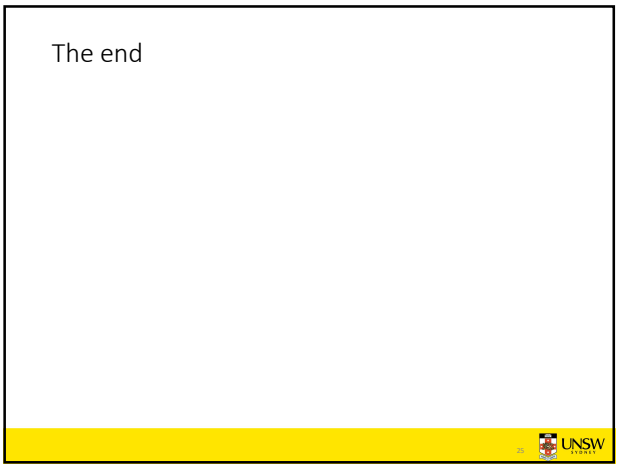


Bowman, I. T., Holt, R. C., and Brewster, N. V. 1999. Linux as a case study: its extracted software architecture. In *Proceedings of the 21st international Conference on Software Engineering* (Los Angeles, California, United States, May 16 - 22, 1999). ICSE '99. ACM, New York, NY, 555-563. DOI= http://doi.acm.org/10.1145/302405.302691

23



24



25