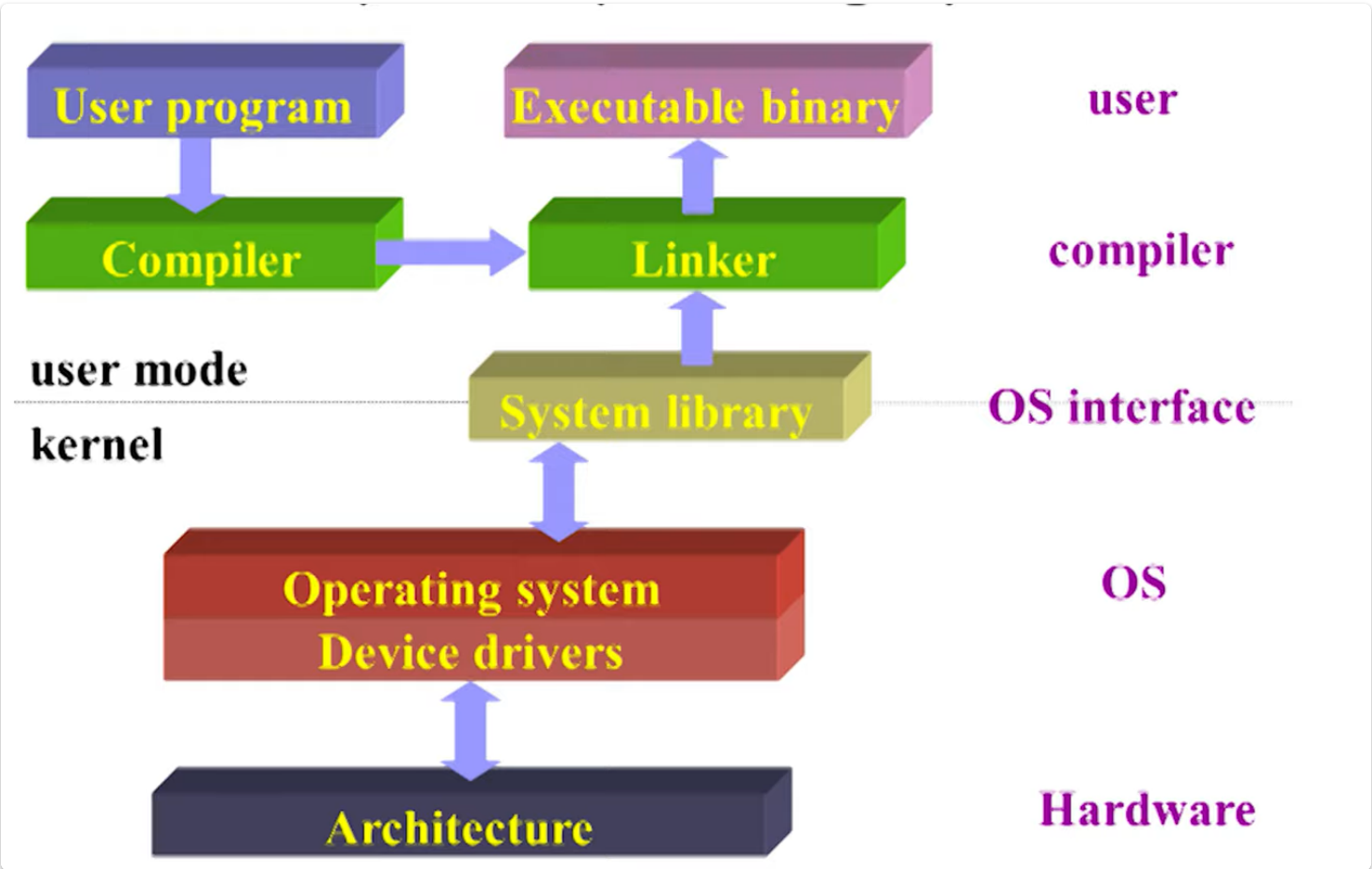
OS_110_CH1

What is OS

- User
- Application : solve the computing problems
- OS : control 控制 and coordinate 分配 the hardware/resource
- Hardware : provide basic computing resources (CPU, memory)
- 是一個 permanent software，開啟時就會一直看到

General-Purpose Operating Systems



Definition of an Operating System

- Resource allocator
- Control program
- Kernel (OS 的別名)

Goals of an Operating System

- Convenience 方便
- Efficiency 效率
  兩個會是 contradictory 矛盾
  在過去，效率比較重要

Important of an Operating System

- System API are only interface between user applications and hardware
- OS 不能有任何 BUG
- OS 的擁有者控制著該軟體和硬體行業

Modern Operating Systems

- x86
  - Linux
  - Windows
- PowerPC
  - Mac OS
- Smartphone
  - Android, iOS, Windows10 Mobile, Ubuntu Touch
- Embedded 嵌入式
  - Embedded Linux, Windows CE
  - Raspberry, Xbox

Computer-System Organization

- Goal : Concurrent execution of CPUs and devices competing for memory cycles

- Device controller
  - data register, status register
  - Busy/wait output
  - is in charge of a particular device type 負責
  - has a local buffer
  - I/O is from the device to controller's local buffer
  - CPU moves data from/to memory to/from local buffers in device controllers

Interrupt

- Interrupt allow a device to change the flow of control in the CPU

- 當 I/O 呼叫 interrupt 時，CPU 會被中斷，然後跳到指定 route 處理，然後在跳回去處理原本做的事情

- 可以讓 CPU 跟 I/O 同時做事情

- 流程
  - CPU 跟 Controller 指派 I/O 工作，然後 CPU 可以繼續做自己的事
  - Controller 處理 I/O 工作，當完成時，會向 CPU 發出 interrupt
  - CPU 收到會中止動作，處理，然後返回原本工作

- Hardware sending interrupt to CPU, we call signal

- when Software occur error or service (system call), it will sending interrupt

- Software sendeing interrupt to CPU, we call trap

- 流程
    - Hardware
        - CPU 在做事情時，發生了硬體 interrupt
        - 就會中斷且跑到 interrupt vector 找對應的事情(service routines)
        - 回到 CPU 原本的事情
    - Software
        - CPU 在做事情時，發生了軟體 interrupt
        - 用 switch case 接 interrupt，執行對應的事情
        - 回到 CPU 原本的事情

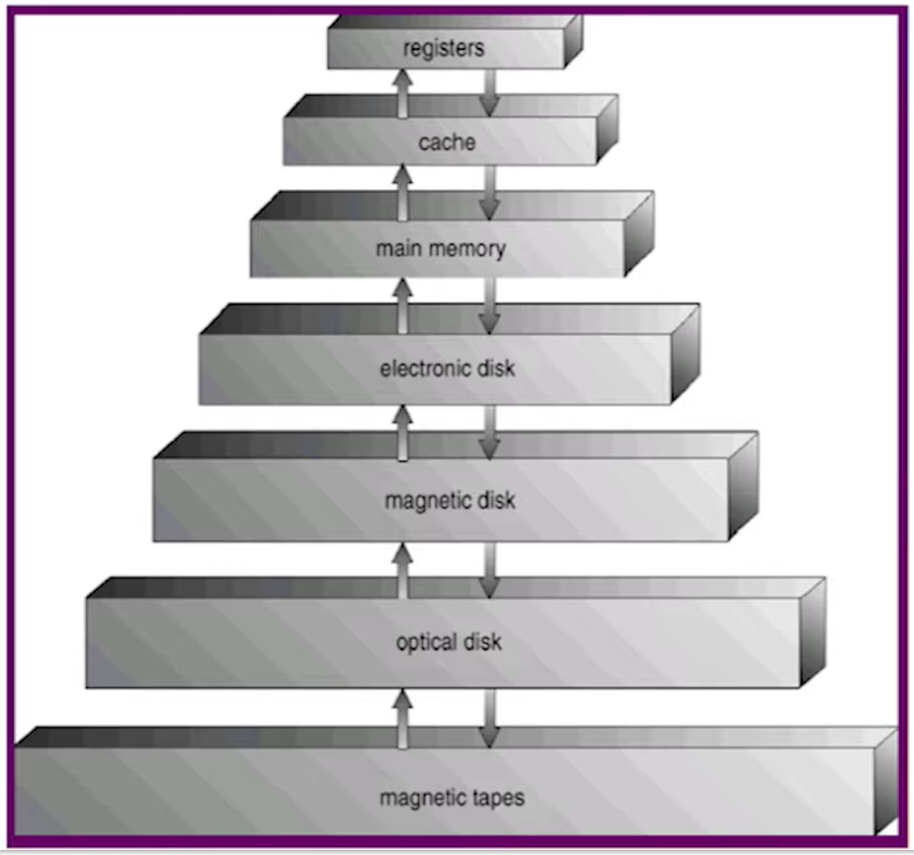
Common function of Interrupts

- 兩種會跑不同的 OS function
- interrupt vector, service routines
- 必須要記住原本的 address 的位址
- 被重複打斷的話，會產生很多問題
- 所以會有系統會在被 interrupt 時，會將後面的 interrupt 禁用，以免產生迴圈

Storage-Device

Storage-Device Hierarchy

Storage-Device Hierarchy



- 越上面速度越快，反之
- 越上面空間越小，反之
- 越上面價格越貴，反之
- main memory 分層，上下分隔 Volatility 揮發性
  - registers, cache, main memory are **volatility**
  - 簡單來說，在 main memory 以上的設備，都是揮發性裝置
- Main memory
  - CPU can access directly
  - RAM
  - Volatility
- Secondary

- large nonvolatile storage

RAM (Random-Access Memory)

- DRAM：沒有那麼快、較便宜
- SRAM：快、貴
- 製成影響

Disk Mechanism

- Disk：讀寫頭、連續資料也可以很快
- SSD：效能高、不連段資料很快

Cache

- 暫存常用到的資料
- 必須一致性
- 他只是一個複製的資料

Coherency and Consistency Issue

- 在更改資料的時候，只更改 cache 的，可能會導致資料不一致性
- 所以如果在有多個程式同時修改記憶體，就會有衝突
- Single task accessing
  - 沒問題
- Multi-task accessing
  - 有多程式同時讀取同記憶體，可能會有一致性衝突
- Distributed system
  - 多了還有網路問題，所以更麻煩

Hardware Protection

可以使得程式互相不影響執行

Dual model

- User mode

- Kernel mode (Monitor mode) : OS 執行的
- 用一個 bit 去控制
- 當 interrupt 發生，就會切換 bit 的值

Privileged instruction 特權指令

- 只能在 Kernel mode 時執行，若不是會報錯
- Requested by users (System mode)

I/O Protection

- 所有的 I/O 都是 Privileged instruction

Memory Protection

- 為了保護 Interrupt vector 跟 interrupt service routines
- 也保護各程式的資料訪問跟 over-write
- 紀錄兩個 register 檢查
  - Base register
  - Limit register
- 可以改變這個 register，一定是 Privileged instruction

CPU Protection

- 保護 CPU 當中不會有程式霸佔，在裡面有迴圈
- 用 Timer 保護，Time sharing

- Load-timer 也一定是 Privileged instruction