CH1、導論

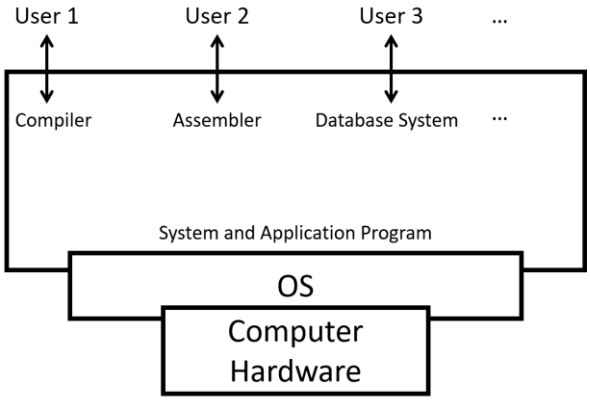
OS 介紹及 System Types 介紹

目錄：

Computer System Structure

主要由 4 個部分所組成

1. Hardware：CPU、Memory、I/O Devices
2. Operating System*(存在 RAM、使用者共享)*
3. Application Programs*(例如：Office、Word、Processor... 等)* (+ System program*(例如：Compiler、Assembler、Linking Loader、Debugger... 等)*)
4. Users：人或其他機器/系統

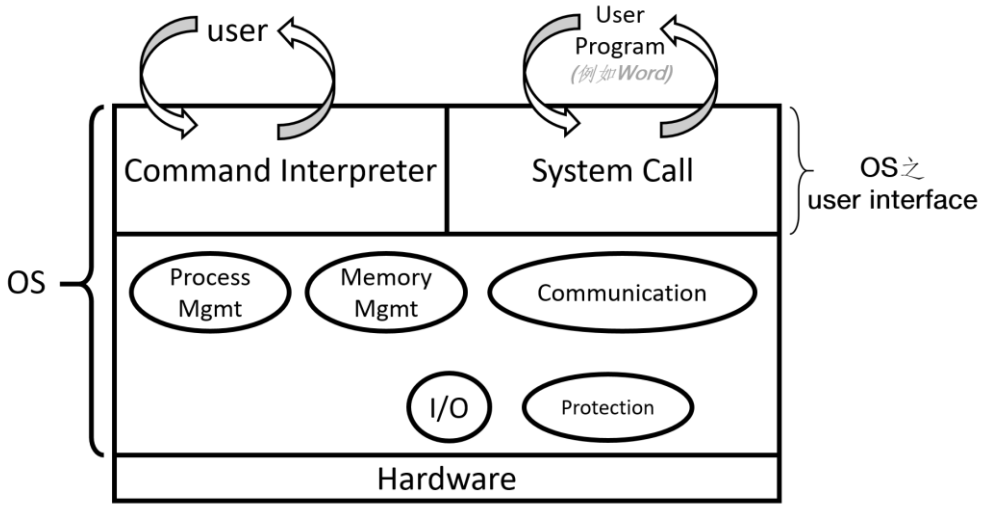


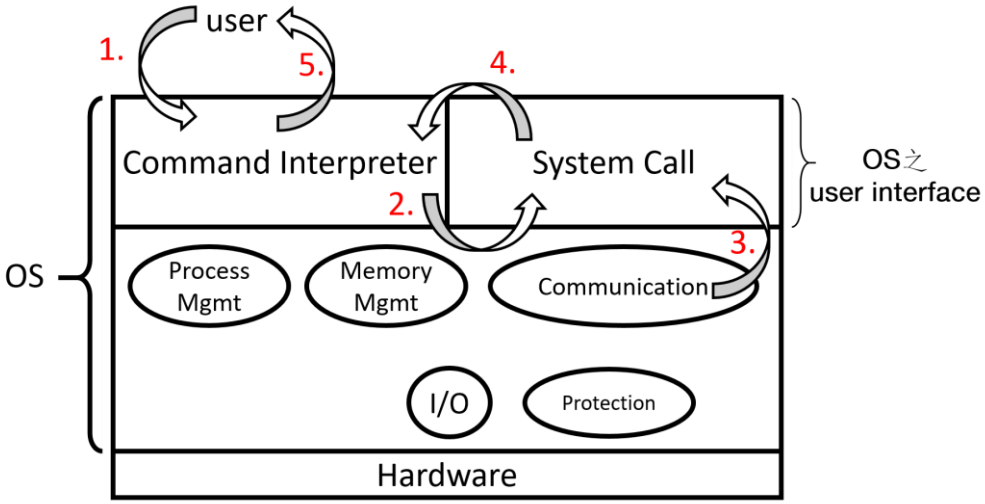
[補充]

1. Bare Machine(裸機)：純粹只有硬體組成，其上無其他 OS 及 System program
2. Extended Machine：Bare Machine 加入 OS/system program 或 app

OS 架構

[其他版本圖一]





kernel(核心)：OS 之重要服務之集合、且常駐(resident) in Memory.
雖無非常明確、嚴緊之定義，但可以以『總是在運行』作為定義參考

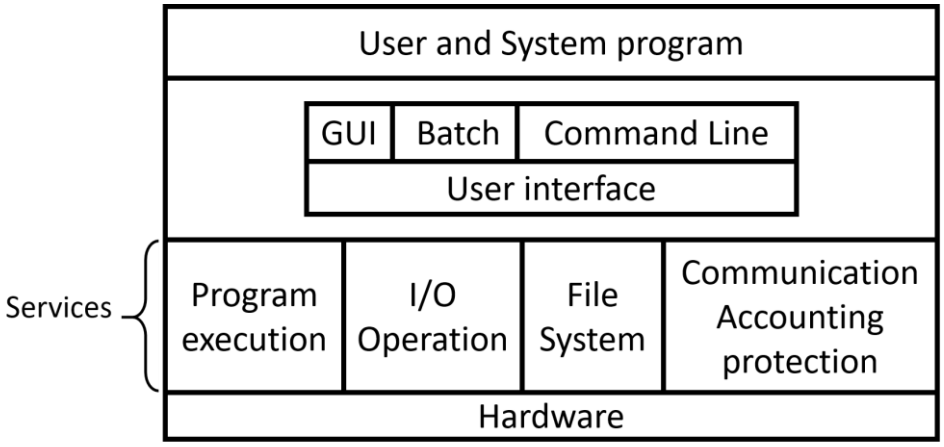
例：下列何者為 OS 的一部分？

a. The CPU Scheduler
b. Device Drivers
c. Compilers
d. Word processors
=>a，因為 b、c、d 皆可為『需要時再安裝/使用即可』

Command Interpreter(命令解譯器)，三種型式：

1. Menu
2. Text(Command/dos/UNIX/Linux)
3. GUI(Graphic User Interface)、Mac、Windows

[恐龍書版圖二]



OS 之 Roles(扮演角色或目的)

1. 提供一個讓 users 易於操作電腦之溝通介面
2. 提供一個讓 users programs 易於執行之環境
3. 作為一個 resource(資源)的管理者協調分配 resources*(CPU、Memory、I/O)*，期望 resources 可有效利用
4. 作為一個監督者，監控所有 process 執行，避免 process 之有意或無意之破壞，造成 system 重大危害

OS 追求『公平』與『效率』，雖然兩者基本上是矛盾的*(ex：FIFO(公平)與 SJF(效率)*

CPU 延遲時間過長原因與解決策略

一、(早期)人為設定/介入時間太長

1. 熟練的 System Administrator
2. 發展"Resident Monitor"
   i. Automatic Job Sequencing
   ii. Control Card & Command Interpreter
   iii. Device Drivers(驅動程式)：實現 Device Independent
   iv. Interrupt 機制

二、(始終存在)IO 運作太慢使得 CPU 等待時間過長

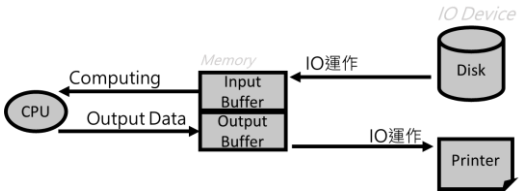
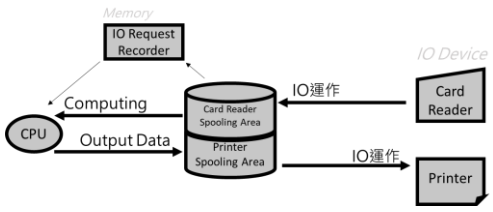
1. 使用較快的設備/機制介入 CPU 與 IO：Buffering、SPOOLing
2. 讓 CPU 總是 Busy，即讓 CPU 在多個工作間切換：Multiprogramming

CPU/IO-Bound

CPU-Bound	IO-Bound
CPU 計算量大、效能取決於 CPU 運算速度	IO 運作量大、效能取決於 IO 運作速度
若 CPU-Bound 與 IO-Bound Job 同時爭取 CPU 時，IO-Bound Job 會有較高之優先權*(IO-Bound Job 會較快完成)*	若 CPU-Bound 與 IO-Bound Job 同時爭取 IO 時，CPU-Bound Job 會有較高之優先權*(CPU-Bound Job 會較快完成)*

Buffering(緩衝)與 SPOOLing(線上週邊同時處理).

SPOOL：Simultaneous Peripheral Operation On-Line

	Buffering	SPOOLing
定義	將傳輸之資料暫時放置於 Memory 內	將 Disk 作為極大的緩衝區使用，CPU 會『誤以為』IO 已完成
硬體	Mem(有 SWAP)	少許 Mem*(紀錄 IO Request、無 SWAP)*、Disk
特色	一次只允許一種/自身 IO 在運作	可同時有多種 IO 在運作，可能產生死結*(多個 IO 通道)*
圖示		

Multiprogramming System

(一)系統允許多個 Jobs (process)同時執行，即是：

1. 主要目的：提高 CPU utilization
2. 作法：透過 Job Scheduling (or CPU Scheduling 技術) 達成

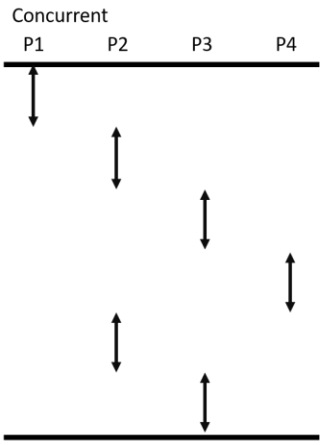
   Ex：當執行中的 process 在 waiting for I/O-Completed，則 OS 可將 CPU 切換給另一個 process 執行，避免 CPU idle，即只要系統內存夠多的 Jobs 存在，則 CPU idle 機會下降

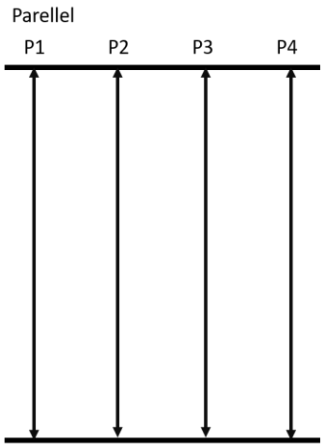
(二)Multiprogramming Degree

系統內之 process 數目，一般來說，若 Degree 愈高，則 CPU utilization 愈高。*(CH8 Virtual Memory 中的 Thrashing 例外)*

(三)多個 process 同時執行之方式有二：

1. Concurrent Execution：一顆 CPU，輪番交錯使用*(看起來像平行執行)*
2. Parallel Execution*(真的實際在平行執行)*





Time-Sharing System(分時系統)

(一)Def：又叫 Multitasking*[恐]*

It's a logical extension of multiprogramming system.

與 Multiprogramming 最大差異：CPU 的切換頻率極高(CPU switching highly frequently)

(二)特色：

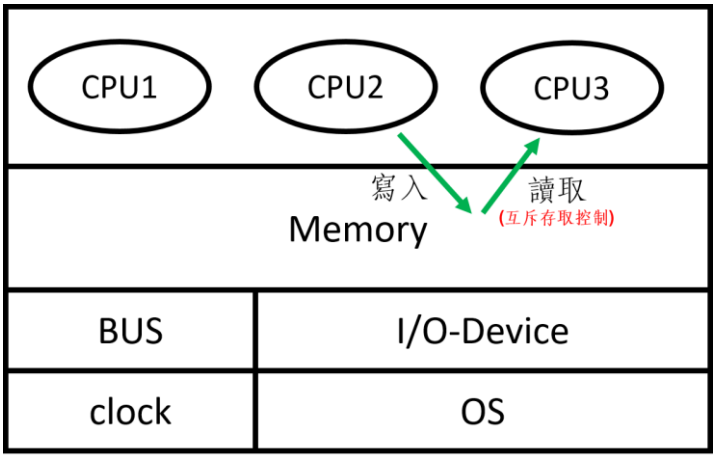
1. 對 user response 時間很短*(ex：小於 1 秒)*
2. 適用於"user interactive" computer/環境
3. CPU：CPU Scheduling 採用 RR 排班法*(CH4)*
4. Mem：使用 Virtual Memory 技術，擴展 logical memory space
5. IO：使用"spooling"技術，實現 I/O Device 之共用*(類似 buffering 技術)*

Multi-process System

(一)Def：又叫 Multiprocessing 或 Parallel System 或 Tightly-coupled system

主要特色如下：

1. 一個機器(或 Mother board)有多顆 processors (或 CPUs)
2. 這些 CPUs 彼此共享機器的 Memory BUS、I/O Devices、power supplier
3. 通常受同一個 clock 之時脈、同一個 OS 控制
4. processors 之間的溝通大部分採"Shared Memory"方式



(二)Benefit(好處)

1. Increased Throughput(產能增加)：因為可支持多個工作在不同 CPUs 上平行執行(parallel computing)，但 N 顆 CPUs 產能絕對小於 1 顆 CPU 的 N 倍，原因有二：
   甲、resources contention(資源競爭)
   乙、processor 間之 Communication 會抵消產能
2. Increased Reliability(可靠度的提升)：萬一一個 CPU 壞了，系統不會馬上停頓，因為其他 CPU 仍可運行。名詞解釋：
   甲、Graceful degradation(漸進式滅亡)：系統不致因為某些 HW/SW 元件故障而 Fail-soft
   乙、Fault-Tolerant System(容錯系統)：具有 graceful degradation 性質
3. Economy of Scale：(運算能力之擴充符合經濟效益)：
   甲、因為 N 顆 CPUs 在一部機器內與 N 部機器相比，成本較為便宜
   乙、這些 CPUs 共享同一機器之 Memory、BUS、I/O Devices

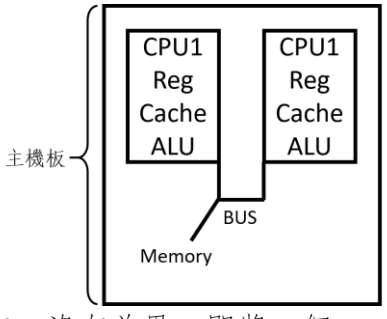
(三)Multiprocessor system：可再細分為 2 種 subtypes：

甲、SMP(Symmetric Multi-Processors，對稱式)：每個 Processor 之工作能力相同(identical)且每個 processor 皆有對等的權利來存取資源
優點： 1.可靠度較 ASMP 高
2.效能較高
缺點：SMP 的 OS 設計開發較複雜(因為互斥存取的機制)

乙、ASMP(Asymmetric Multi-Processors，非對稱式)：每個 processors 之工作能力不盡相同，通常採取 Master-Slave 架構(又稱 Boss-Employee*[恐]*)
Master processor 負責工作分派及資源管理、監督 Slave…等管理工作；其他 Slave processors 負責執行工作
優點：ASMP 的 OS 設計開發簡單(與 simple CPU OS 類似)
缺點： 1.可靠度低
2.效能較差(Master 是瓶頸)

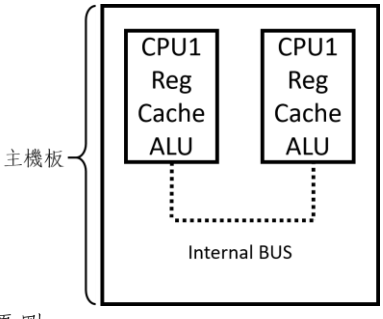
(四)Multiprocessors System vs Multicores CPU

硬體差異(主要)



就 OS 而言，沒有差異。即將一個 Core 視為一個 logical CPU 在看待。
Ex：主機板裝 4 顆雙核=8 顆 CPU 可用(對 OS 而言)

多核



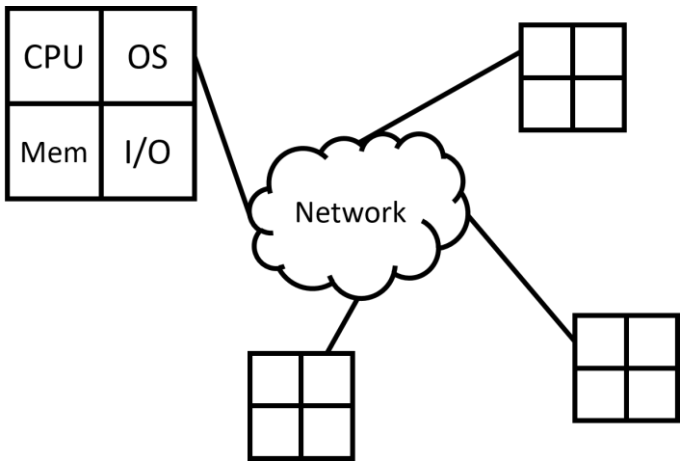
優點：

1. power saving
2. 速度較快，因為同一晶片內資料傳遞較快

Distributed System(分散式)

(一)Def：又叫 Loosely-coupled System

1. 多部機器彼此透過 Network*(LAN、WAN)*(或 Bus)相連
2. 每部機器之 CPU 有各自的 Memory、Bus、I/O，非共享
3. 各 CPU 之 clock 時脈控制不一定相同
4. 各 CPU 上之 OS 也不一定相同
5. 各機器之間的溝通大都採用"Message Passing"方式
   甲、建立線路
   乙、相互傳輸
   丙、釋放 link



(二)構建之理由(好處)

1. Increased Throughput (支持 parallel computing)
2. Increased Reliability
3. Resources sharing，所以成本降低

   支持"Clint-Server Computing Model"之實施
   1. Server：提供某些服務的機器*(ex：mail、file、DNS、printer、computing)*
   2. Client：本身不提供服務，且它需要某些服務時，則發請求給 Server，服務完後再將結果回傳

   "peer-to-peer(P2P)" model：peer 同時具有 Server and Client 角色
4. remote sites communication：需求被滿足。*(ex：email、FTP via internet)*

Real-Time System(即時系統)

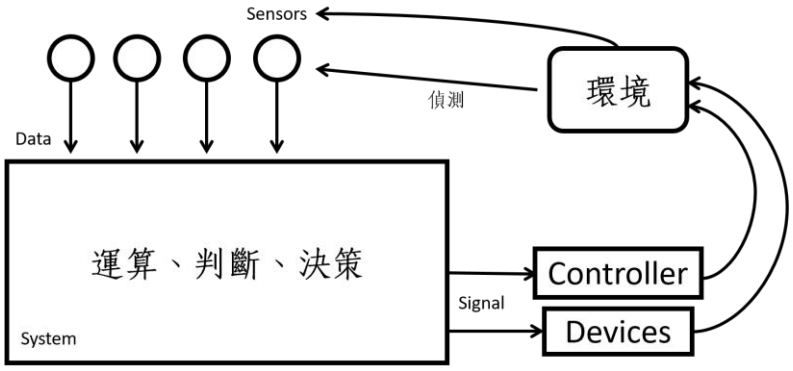
(一)分為 2 種：

1. Hard Real-Time
2. Soft Real-Time

(二)Hard Real-Time System

1. Def：This system must ensure the critical tasks complete on time。即時工作必須依規定的時間限制內完成，否則即算失敗
2. 例：軍事防衛系統、核能安控系統、工廠自動化生產、機器人控制

3. 圖示：



4. 設計考量
   甲、所有時間延遲之因素需列入考量
   sensor data 傳輸速度、運算速度、signal 傳輸…確保這些時間的加總能滿足時間 deadline 的要求
   乙、所有會造成處理過久或無法預測之設備或機制宜少採用或不用。
   例如：磁碟宜少用或不用、Virtual Memory 絕不採用
   丙、就 CPU Scheduling 設計而言，需先考慮 Scheduling 與否，再進行排程。*(ex：rate-monotonic、EDF Scheduling)*
   丁、Time-Sharing System 無法與之並存
   戊、OS 所造成的分派延遲宜降低
   一般實務上，Hard Real-Time System 鮮有 OS*(OS 幾乎不存在)*，尤其是 Embedded Real-Time System
   己、現行的常用系統 OS 不支援 Hard Real-Time 特性，通常是客製化設計*(ex：Linux、Unix、Windows、MAC、Solaris…等)*

(三)Soft Real-Time

1. Def：This system must ensure the real-time process get the highest priority than the others and retain this priority level until it completed.
2. 例：Multimedia system、simulation system、VR system
3. 系統設計考量
   甲、就 CPU Scheduling 設計而言
   1.需支持 preemptive priority scheduling
   2.不可提供”aging”
   乙、儘量降低 kernel 的 dispatch latency time
   丙、可支援 Virtual Memory 並存，但是要求 Real-Time process 的全部 pages 必須都待在 Memory 中，直到完工
   丁、與 Time-Sharing 可以並存(ex：Solaris)
   戊、一般商用 OS 皆支持 Soft Real-Time System 之特性

Batch System (批次系統)

(一)Def：將一些較不緊急、定期性失誤互動性之 Jobs 累積成堆，再分批送入系統處理

(二)例：庫存盤點、報稅、掃毒、磁碟重組、清算系統

(三)主要目的：提高 resource utilization，尤其是在冷門時期，不適合用在 Real-Time System、user-interactive app(*ex：game*)

Handheld System(手持式)

(一) 例：PDA、Smart Phone、iPad(平板)

(二) Hardware 天生的限制，帶來 Software 必須配合之處

1. Slower processor=>背後限制為：1.power 供應問題
   2.散熱處理問題
2. Memory space(RAM)有限=>1.運算宜簡單、不宜複雜
   2.程式 size 宜小、不用的 Memory space 立刻釋放
3. 螢幕很小=>顯示內容要有所刪減(ex：手機網站 design)

例題(是非題)：

1. *Multiprogramming 一定是 Multiprocessors？*
2. *Multiprocessors 一定是 Multiprogramming？*
3. *Multiprogramming 一定是 Multiusers？*
4. *Multiusers 一定是 Multiprogramming？*
5. *Time-Sharing 一定是 Multiprogramming？*
6. *On-Line System 一定是 Real-Time System？*

F、T、T、F、T、T、F(On-Line 有可能是 Batch System)