概述

OS 的目的

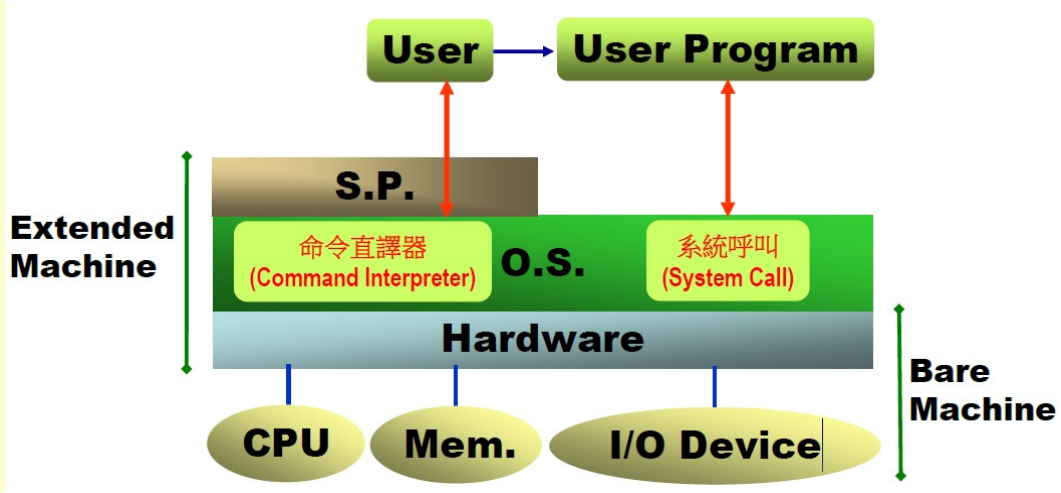
使用者觀點

- 使用者與硬體之間的**介面**
- **User Program 易於執行**的環境

系統觀點

- **資源分配者**
- **監控 User Program**, 防止不正常的運作

Bare Machine & Extended Machine



- **Bare Machine(裸機)**
  - **無系統軟體**
- **Extended Machine(延伸機器)**
  - **有系統軟體**

作業系統演進

- **CPU Idle Time** 過長
- **人為介入**，人工安排操作順序
- **CPU 速度 >> I/O 速度**
- 解決方法
  - **問題一：** 利用 **Resident Monitor(常駐監督程式)**
  - **問題二：**
    - 以**較快速的設備**介入 CPU 和 I/O 設備之間
      - Off-line
      - Buffering
      - Spooling
  - 讓 CPU **保持 Busy**
    - Multiprograming

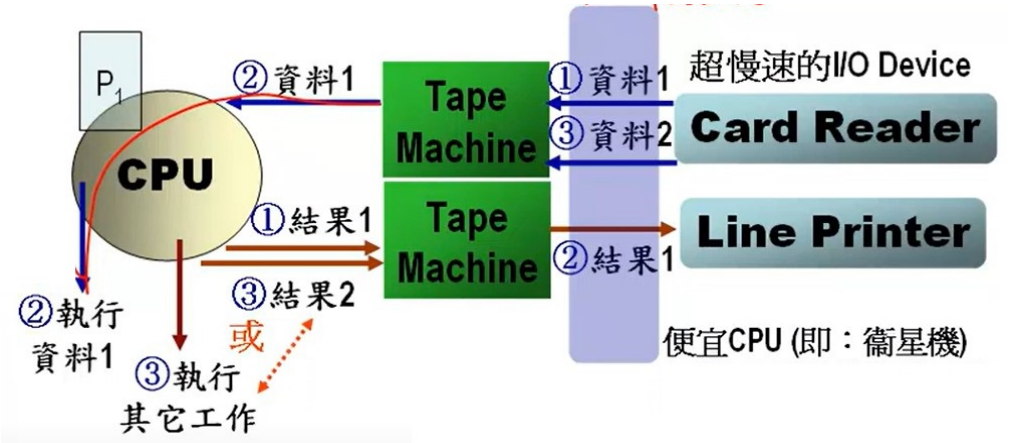
Resident Monitor

- 執行時需載入記憶體
- Control Card 紀錄工作執行順序，Resident Montior 對 Control Card 直譯
- Resident Monitor 負責工作的控制權轉換，**減少人為介入時間**

On-line

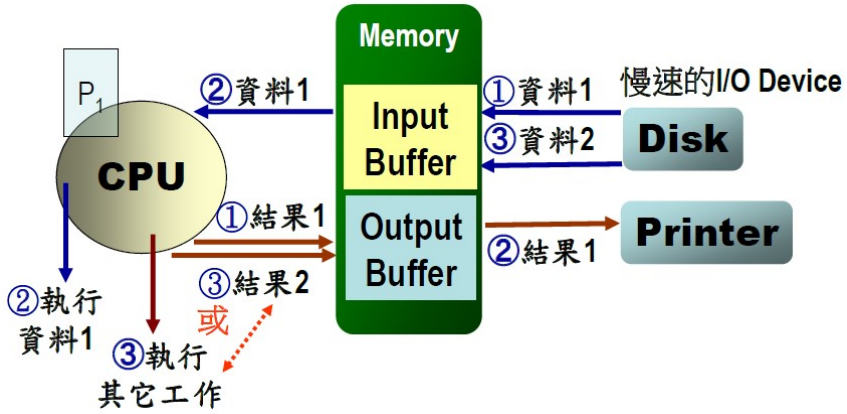
- CPU **直接連結**超慢速 I/O 設備

Off-line



- Tape Machine 和超慢速 I/O 設備溝通
  - 特用的 I/O 程式
  - **便宜的 CPU (衛星機)**
- CPU Computation(2)跟 I/O Operation(3)**重疊執行 (Overlay Execution)**
- Tape Machine 缺點
  - 需人為操作
  - 只能循序存取，讀前面的資料需要倒帶

Buffering



- CPU 和 I/O 設備中間使用 **Memory** 取代 Tape Machine
  - Input Buffer
  - Output Buffer
- CPU Computation 跟 I/O Operation **重疊執行 (Overlay Execution)**

I/O Bound Job

- **大量的 I/O 操作**，工作取決於 **I/O 設備的速度**
- CPU 總是面對**空的** Input Buffer 被迫等待
- CPU 總是面對**滿的** Output Buffer 被迫等待

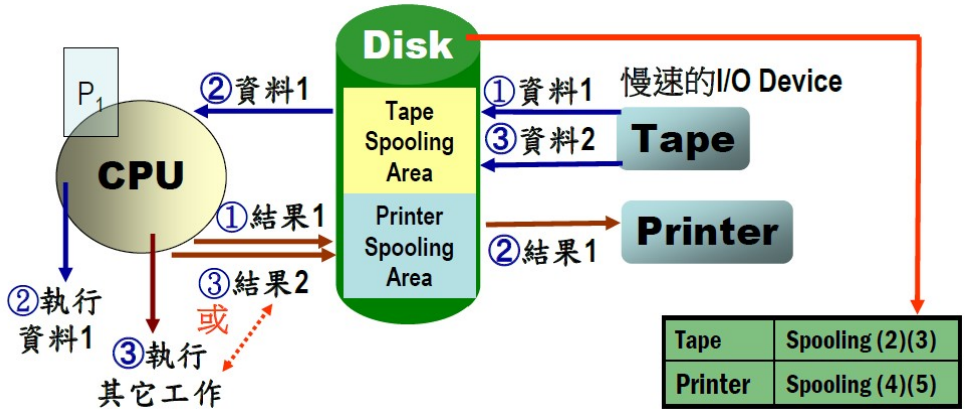
CPU Bound Job

- **大量的 CPU 操作**，工作取決於 **CPU 的速度**
- I/O 總是面對**滿的** Input Buffer 被迫等待
- I/O 總是面對**空的** Output Buffer 被迫等待

排班程式

- **系統內 I/O Bound Job 和 CPU Bound Job 必須均於混合**
- CPU 和 I/O 設備充分被利用

Spooling (線上同時周邊處理)



- **Simultaneous Peripheral Operation On-Line Processing**
- CPU 和 I/O 設備中間使用 **Disk**
  - Disk 個別建立專屬 **Spooling Area** 給每個 I/O 設備
  - e.g. Printer Spooling Area, Tape Spooling Area...etc
- CPU Computation 跟 I/O Operation **重疊執行 (Overlay Execution)**
- 建立 **Spooling Table** 紀錄 Input Data 在**硬碟的 Spooling Area 位址**
- **優點**
  - 讓**不同工作**的 CPU Computation 和 I/O Operation 同時執行
- **缺點**
  - 需要少量的 Memory 空間紀錄 I/O Request 執行狀況
  - 需要 Disk 空間紀錄 Spooling Table

Spooling & Buffering 比較

- Spooling **不同工作之間**的 CPU Computation 跟 I/O Operation 重疊執行
- Buffering **同一工作**的 CPU Computation 跟 I/O Operation 重疊執行

- Buffering 區域在 User Memory，工作執行結束釋放記憶體，導致 Output Buffer 與新的工作重疊，導致結果不正確
  - Lock User Memory Space
    - **整個 Process**
    - **局部區域**
    - 資料送往 OS Memory

系統類型

Multiprograming System

- **存在多組 Process 同時執行**(一段時間，不是一個時間點)
- 作法：**CPU Scheduling**

Concurrent

- **單一時間點**只有**一個 Process**，**一段時間內**有多個 Process
- 單 CPU
- Multiprograming System

Parallel

- **單一時間點**有**多個 Process**
- 多 CPU
- Multiprocessing System, Distributed System

Multiprograming OS 需求

- **CPU Scheduling**
  - 會有好幾個工作在記憶體準備執行，選擇其中之一交給 CPU 執行
- **Job Scheduling**
  - 決定哪些工作先放入記憶體，另一些暫時儲存在輔助儲存體
- **Memroy Management**
  - 將多個要執行的工作存放到記憶體準備執行

Multiprograming Degree

- 系統內待執行的 Process 數目
- **Multiprograming Degree 越高，CPU 利用率越高**
- 例外：**Thrashing(輾轉現象)**會造成 CPU 利用率下降
  - 當 Multiprograming Degree 超過某個程度，**造成所有行程忙於 Swap In/Swap Out**，CPU 反而因此 Idle

Time Sharing System

- Multiprograming System 的一種
- Resource Sharing **資源共享**技術給不同使用者
  - **Round Robin(RR) Scheduling**
  - 共享 **Memory Space**

    - 透過 **Spooling** 共享 I/O 設備
- 對所有 User **公平**
- 交談式系統
    - CPU 不斷在不同工作切換，使用者可以和正在執行的程式**交談**
    - e.g. 遊戲
- 要求系統**反應時間短**

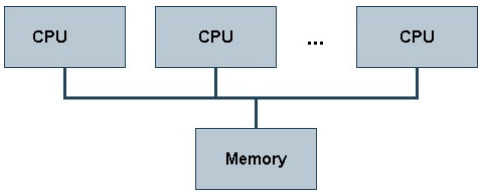
Distributed System

- **Tightly Coupled:** Multiprocessor System, Paralle System
- **Loosly Coupled:** Distributed

Tightly Coupled 緊密耦合

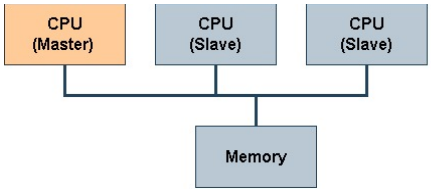
- CPU(或 Processor)數目 >= 2
- 共享**記憶體**、**I/O**、**Bus**
- **同一個 Clock 和 OS** 的控制與協調
- CPU 之間的溝通大部分採用 **Share Memory 技術(共享變數)**
- 也可以稱為 Multiprocessing System, Parallel System
- 可以將**多個 Process** 或**一個 Process 上的多個 SubTask** 配置到不同的 CPU，以 Parallel 方式執行

Symmetric Multiprocessing (SMP)



- 每一個 Processor 具有**相同功能**
- **可靠度高**當一個 Processor 損壞，未完成工作可轉移到其他 Processor
- **Load Balance 負載平衡**希望每一個 Processor 負載均等

Asymmetric Multiprocessing (ASMP)



- 一個 **Master Processor** 控制協調分配 Process 到 **Slave Processor** 運作
- 效能比 SMP 佳，可靠度較差

特性

- **Communication 通信**和 **Resource Contention 資源競爭**會產生額外消耗

Loosely Coupled (鬆散耦合)

- 大多稱為 **Distributed System**
- 每一步機器**有自己的 Processor, Memory Space, I/O Device, Clock,**etc

- 不一定受同一個 OS 控制
- **Message Passing 為主**，透過 **Network** 或**高速 Bus** 連接

Client-Server Model

- **Client** 發出 **Service Request --> Server** 提供 **service**

Peer to Peer Model

- 每個 Site 都可以是 Client/Server

網路作業系統 Network Operating System

- 提供 Network 基礎功能，OS + Network Operation
- User 自行決定工作要在哪個機器執行

分散是作業系統 Distributed Operating System

- 目標：讓使用者以存取自己資源的方式存取遠端資源
- 設計困難

建構分散是系統的優點

- 資源共享
- 加速機算
- 可靠度增加
- 通訊需求

即時系統 Real Time System

- 在**定義嚴謹的固定時間限制**之內完成工作，否則工作就算失效

硬性即時系統 Hard Real Time System

- 時間有嚴格限制
- 大部分的資料儲存在 ROM 或 RAM，不使用輔助儲存體和 Virtual Memory(處理時間過長)
- 工廠自動化、軍事管理、和安管理...etc
- 減少 OS 干預，降低 Dispatch Latency
- 與其他系統難以結合

軟性即時系統 Soft Real Time System

- 具有即時性的 Process 有最高的優先權
- 高優先權 Process 的優先權值一值維持到完成為止
- 多媒體系統、虛擬實境系統...etc
- 注意事項
  - OS 所造成的延遲時間要有限制
  - CPU 排班需支援 Priority Scheduling，且不能提供 Aging 技術
  - 應避免 Priority Inversion(優先權反轉)問題，可以利用 Priority Inheritance(優先權繼承)

優先權反轉

- 低優先權的 Process 釋放 CPU 但是仍占用資源，導致高優先權的 Process 無法執行工作

優先權繼承

- 暫時調高 Low Process 的優先權到和 Hight Process 一樣，等資源釋放後再把他的優先權值調回來

Clusteering System

- 和 Parallel System 一樣，集合多 CPU 完成工作，不同之處在於它是**集合多個個別系統完成工作**
- 共享儲存裝置
- 高可利用性

Asymmetric Clustering

- 有一台機器處於 **Hot Stand-by Mode**，負責監督其他 Server，若被監督的某台 Working Server 損壞，此主機就接管壞掉的 Server 重新執行未完成的應用程式

Symmetric Clusteering

- 多部機器同時執行應用程式，也彼此監督
- 具有較高的處理效能，可以掌握較多的硬體資源

Handheld System

計算環境

- **Traditional Compution**
  - 傳統作業系統的計算環境
- **Client-Server System**
  - 大略分為**計算伺服器**和**檔案伺服器**
- **Peer-to-Peer System**
  - 所有節點都可以當作客戶端或伺服器端
- **Web-Based Computing**
  - 透過瀏覽器與網際網路伺服器進行
  - 現今的作業系統都已內建網路功能