



NATIONAL QUEMOY UNIVERSITY

系統程式期末報告

資訊工程學系二年級

110710535

楊紓萍

系統程式期末報告

目錄

chapter 1

馮紐曼模型

操作系統的概念

進程

進程 API

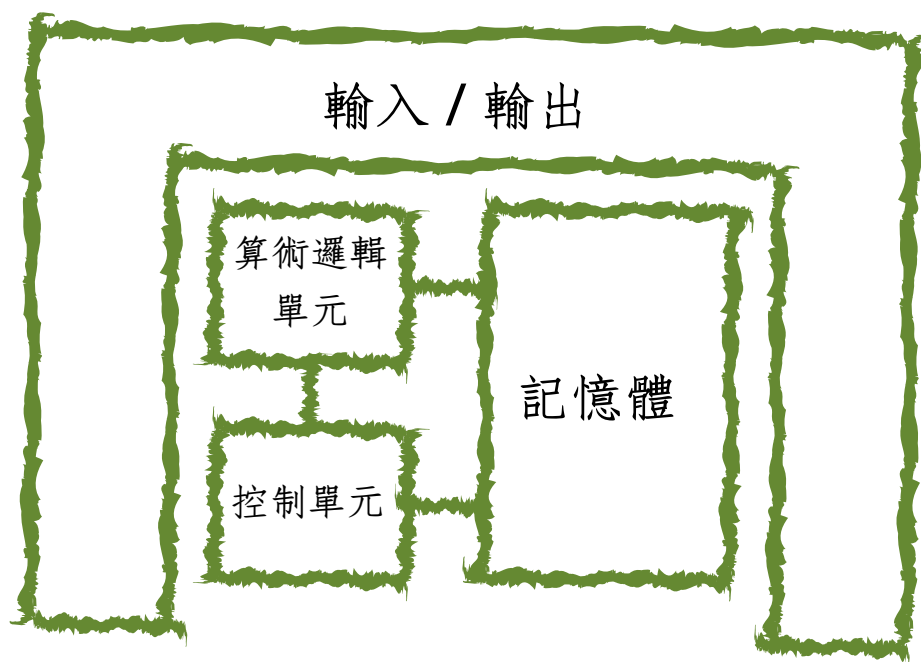
chapter 2

chapter 3

chapter 4

CHAPTER 1

馮紐曼模型



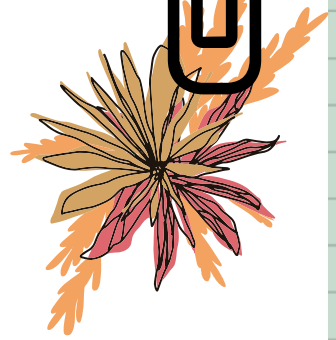
▲ 馮紐曼模型



CHAPTER 1



操作系統的概念

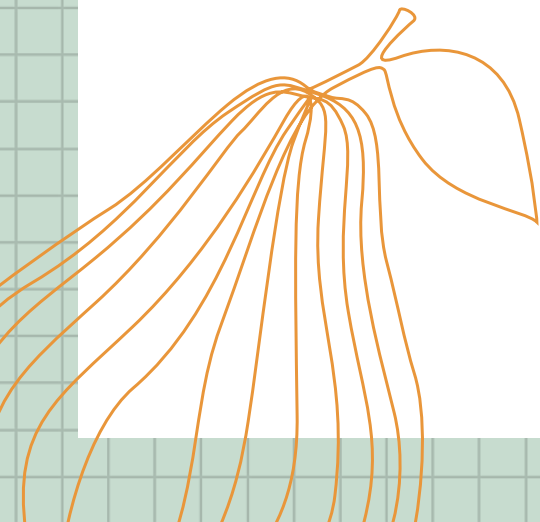


依據馮紐曼模型，當電腦要執行一個指令，他會先 fetch 這條指令，然後解碼並且執行，執行完畢後跳到下一條指令繼續執行，直到程式完成。

在這個執行過程中，可以利用操作系統讓程式變得易於使用，允許程式共享內存，甚至同時執行多個程式。

但是像是我們的硬體設備，像是磁碟這些都是物理性的資源，所以我們要先將他們虛擬化，將他們變得更通用，所以操作系統有時被稱為虛擬機。

因為虛擬化而讓許多程式得以運行，可以同時訪問自己的指令以及數據，也讓許多程式可以訪問硬體設備，所以操作系統有時也被稱做資源管理器。





CHAPTER 1


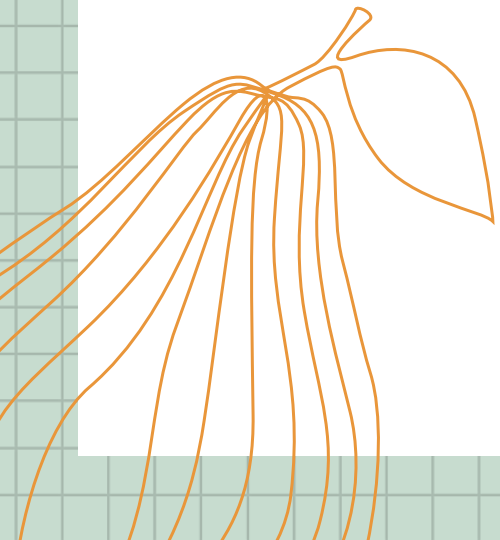


進程

進程的白話就是運行中的程式，程式本身只是程式而已，是操作系統讓他們全部運行起來。

就算電腦不是多核心，只有一個 **cpu** 我們一樣可以同時聽音樂、打電動或下載遊戲，而一個正常的電腦不可能只有這些程式在運行，還可能有一些背景程式或是其他程式，那既然只有一個 **cpu** 為什麼卻可以同時執行那麼多程式呢？

答案是操作系統會虛擬化 **cpu** 來提供好像有很多 **cpu** 的假象！但是硬體上的確只有一個 **cpu**，原理上要怎麼運行呢？運用 **time sharing** 技術讓一個進程只運行一段時間，然後切換到其他進程。也就是說，如果你想同時打遊戲、聽音樂、下載東西，那你的電腦會一下子跑遊戲，一下子放音樂，一下子再去下載東西，這樣你就可以同時做這些事情，但是相對的，每一個進程的速度就會比較慢，因為你的 **cpu** 是共享的。

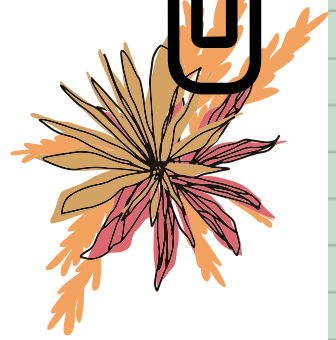




CHAPTER 1



進程 API



所有現代操作系統都必須提供這些API：

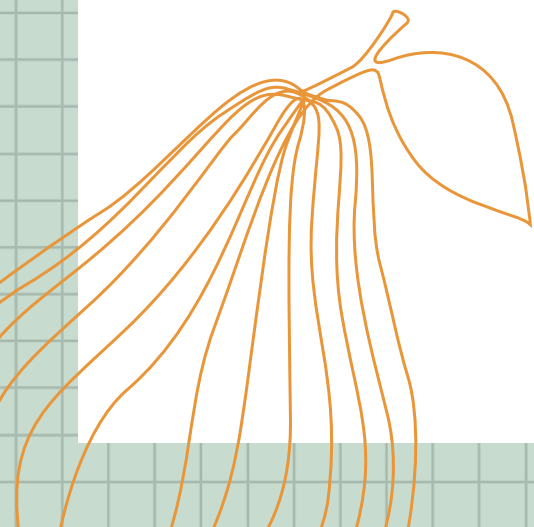
創建：操作系統要可以創立新的進程

銷毀：因為可以創造進程，也要可以銷毀進程，很多進程會在執行完後自動退出，如果他們沒有退出，而用戶希望他們終止，可以強制銷毀進程。

等待：有時需要等待進程停止，所以要一個等待接口

其他控制：除了銷毀或等待進程外，還可能需要其他控制(像是暫停)

狀態：通常有一些接口可以得知進程的狀態，像是運行多久

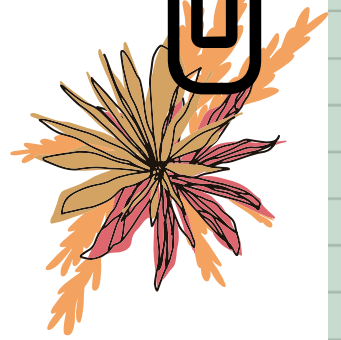




CHAPTER 1



進程創建細節

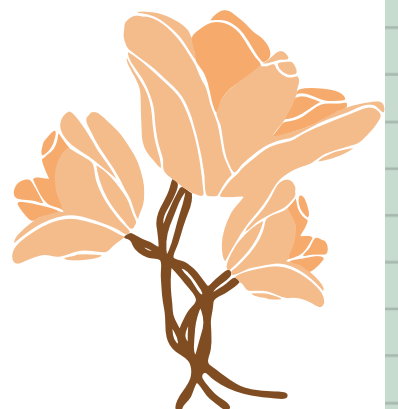
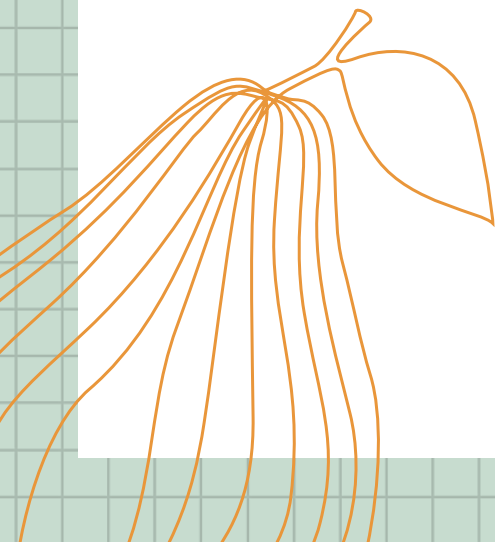


欸，可是程式是程式啊，要怎麼變成進程？

操作系統要先把程式跟一些有的沒的資料都先存到記憶體裡，加載到你電腦裡要存進程的這個位置，那些程式那些程式剛開始是以某種可以執行的格式在磁盤上(或是閃存的 SSD 上)，所以操作系統要把這些東西從磁盤內讀出來，再存到記憶體裡。

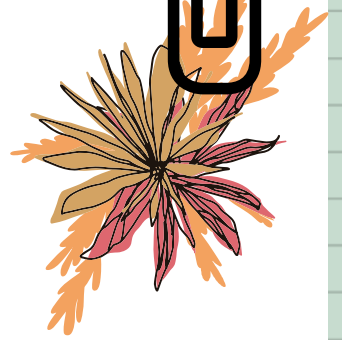
之前的系統加載過程會儘早完成(**eagerly**)，會在跑程式之前就做好，但是現在的系統不是，現在的系統是惰性(**lazily**)執行那個過程，有需要他才會加載。

把程式跟數據載進去後，操作系統還要把一些記憶體分給程式的運行時棧(**run-time stack** 或 **stack**)，操作系統可能會用參數來初始化這個 **stack**，用專業一點的講法來說，就是把參數填進去 **main()** 函數裡面，就是 **argc** 跟 **argv** 這兩個數組。





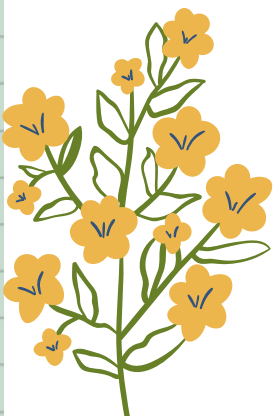
CHAPTER 1



進程創建細節

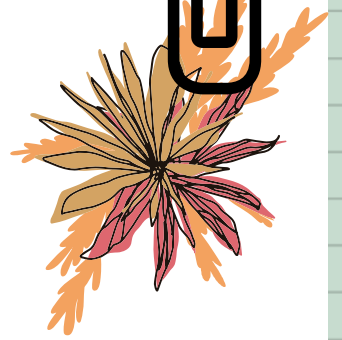
操作系統可能還要分一些記憶體給 heap，heap 在程式裡面用來顯示請求，可以調用 `malloc()` 來請求這樣的空間，然後調用 `free()` 來釋放他，一開始 heap 會很小，隨著程式運行，操作系統可能會分配更多記憶體給進程。

在我自己的理解裡面，這一段的意思是說，因為進程也是需要記憶體去儲存的嘛，所以操作系統就要分一些記憶體給進程，然後很多很多進程會疊成一堆(heap)，像是把很多書本疊在一起一樣，假設進程 = 書本，書本需要書架(記憶體)，去存放，所以要用 `malloc()` 來要求書架讓他放，然後用 `free()` 來釋放書架空間，一開始書很少占用的位置不大，如果書越放越多，就會越來越佔書架的位置。





CHAPTER 2



併發的概念

操作系統會同時處理很多事情，可能會有不只一個需要立即解決的問題同時出現，這件事情就是併發，併發問題也會發生在現在的多線程上。

