磁碟管理

大綱

- 磁碟緩衝與快取
- 磁碟錯誤處理
- 磁碟陣列
- 磁碟排程
- Linux磁碟管理實作
- 本章重點回顧

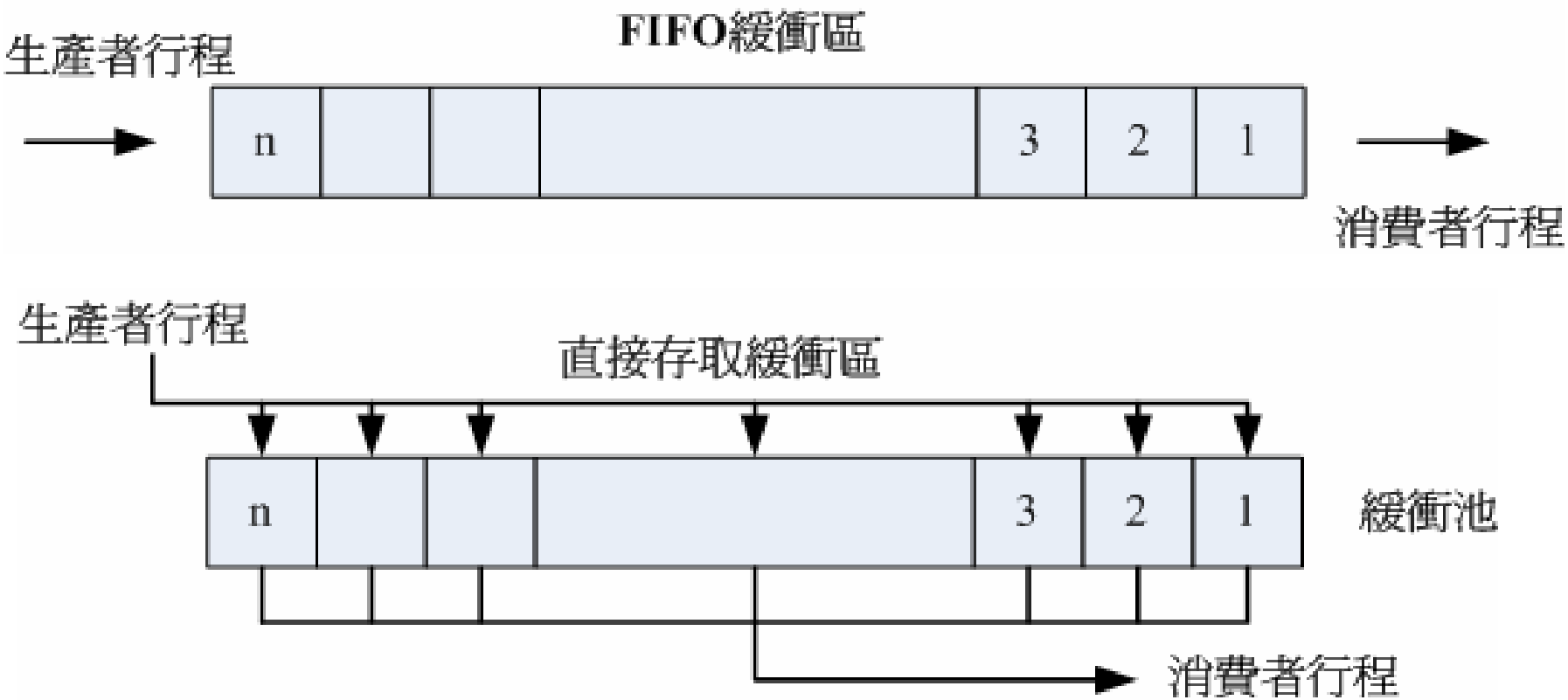
磁碟機的組成

- 磁碟機的組成：
  - 碟片、
  - 主軸馬達（Spindle Motor）、
  - 磁頭定位馬達（Head Positioner Motor）、
  - 磁頭驅動臂（Arm Actuator），
  - 與控制器等
  - 緩衝區(Buffer)
- 主要的傳輸介面
  - IDE、Ultra DMA、SCSI、SATA

磁碟緩衝區

- 組成與功能：
  - 在硬碟裡面的記憶體，可提供傳輸介面與儲存媒體之間的高速介面。存取頻率較高的資料可暫存於此區域內，加速資料的存取
- 主要類別：
  - FIFO緩衝區（First In – First Out Buffer）
    - 讓程序以非同步的方法運作
  - 直接存取緩衝區（Direct Access Buffer）
    - 資料直接由緩衝區取出，減少重複搜尋的動作

磁碟緩衝區的兩種類型

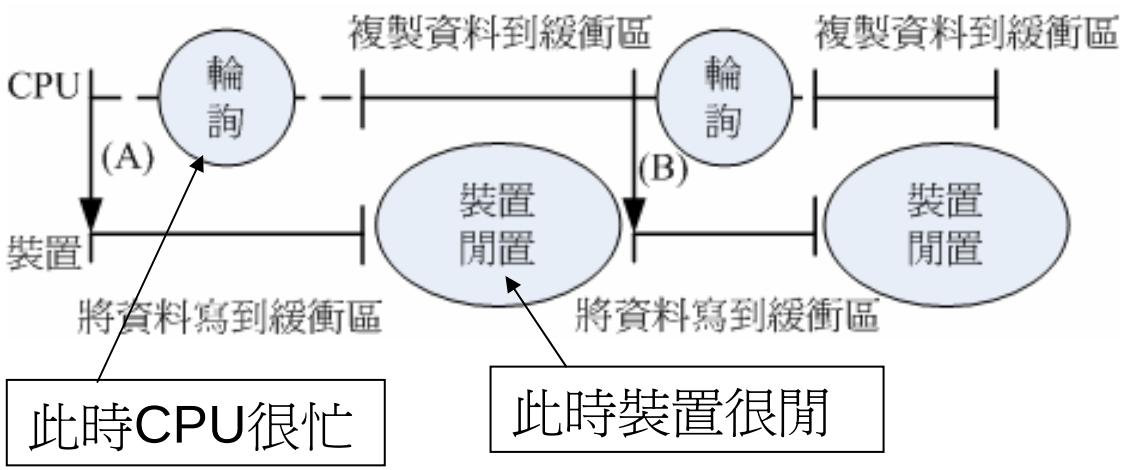


磁碟緩衝區的功能

- 使用緩衝區的目的：
  - 分離生產程序與消費程序的時間
  - 處理不同單位的資料
    - 磁碟的物理儲存量：磁區(sector)，一個512bytes
    - 檔案系統使用block，通常為 4Kbytes
    - 使用緩衝區先處理過區塊的配置，才寫入/讀出到磁碟，避免錯誤的發生。

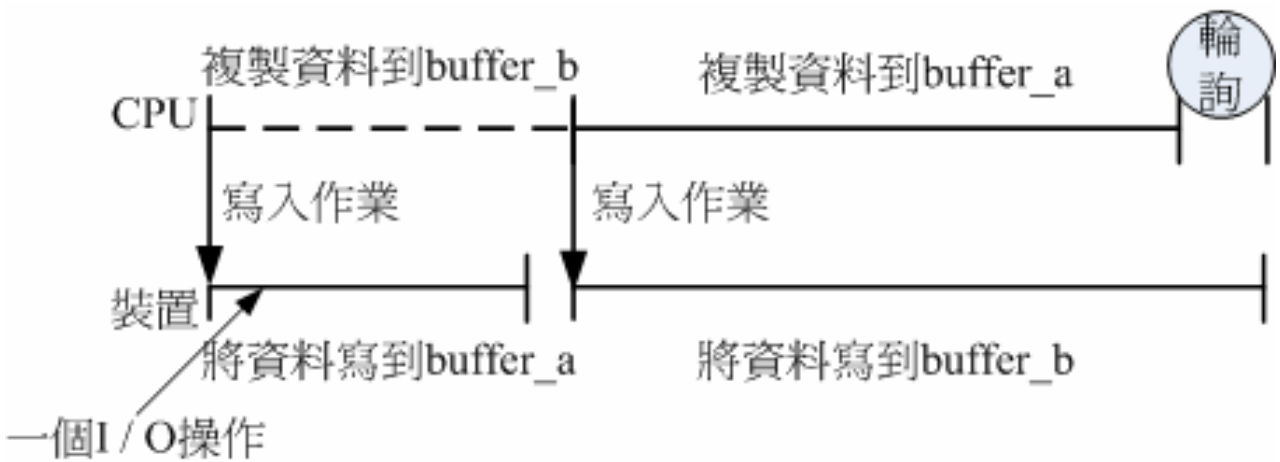
緩衝區的架構

- 作業系統如何使用磁碟緩衝區？四種架構
  - 單一緩衝區
    - 緩衝區一次只能被一個動作所使用
    - CPU必須要透過輪詢的方式來存取緩衝區
    - CPU會很忙碌，且緩衝區的使用率降低



緩衝區的架構-續

- 作業系統如何使用磁碟緩衝區？四種架構
  - 緩衝區置換
    - 將原有的緩衝區分隔成為兩個，
    - 一個負責I/O動作，一個負責 CPU 的查詢



緩衝區的架構-續

- 作業系統如何使用磁碟緩衝區？四種架構
  - 環狀緩衝區：
    - 透過提升CPU與I/O的重疊率改善運作效能
    - 但程序的執行時間不可預測，因此此架構少使用
  - 緩衝區快取
    - 基本上將緩衝區是唯一個快取空間，規劃出緩衝區的位址區段
    - Unix-Like的機器大多使用此類方式運作。

磁碟錯誤處理

- 錯誤的發生可以區分為：
  - 暫時性的（Transient）
    - 可能需要透過重試(Retry)才會正常作業時
    - 可以透過作業系統的工具避免這類『儲存媒體錯誤』
  - 永久性的（Persistant）錯誤
    - 例如磁碟壞軌，通常是物理性的損毀
    - 可以透過偵測工具將這些壞軌定義出來，不再使用該區段來規避這種物理錯誤。

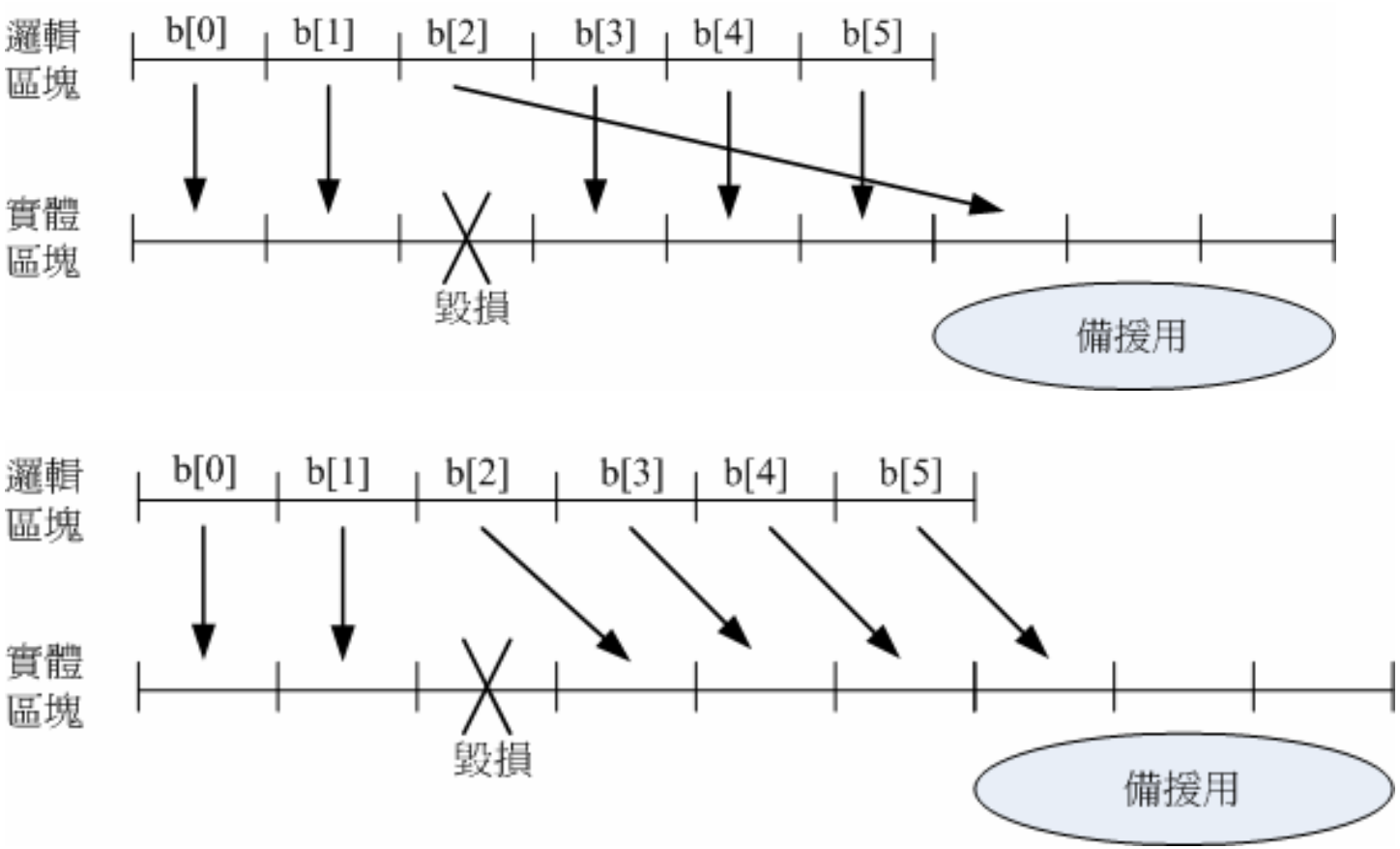
磁碟錯誤處理

- 磁碟的格式化行為：
  - 低階格式化（Low Level Format）
    - 將磁碟表面的磁性物質畫出磁軌
    - 並將磁軌分割出扇行成為磁區後才能繼續使用
    - 可以檢視出有問題的壞軌，並將之定位為壞軌
  - 與高階格式化（High Level Format）
    - 利用磁軌/磁柱/磁區等定義，格式化成為作業系統認識可讀的檔案系統。

磁碟錯誤處理

- 磁碟裝置毀損，常見的解決方法
  - 磁區轉遞（Sector Forwarding）
    - 將原本壞軌上的資料寫入備援區段中
  - 磁區順延（Sector Slipping）
    - 所有的資訊都往後順延一個位址

磁碟錯誤處理示意圖


邏輯
區塊
b[0]
b[1]
b[2]
b[3]
b[4]
b[5]
實體
區塊
毀損
備援用
邏輯
區塊
b[0]
b[1]
b[2]
b[3]
b[4]
b[5]
實體
區塊
毀損
備援用

磁碟陣列

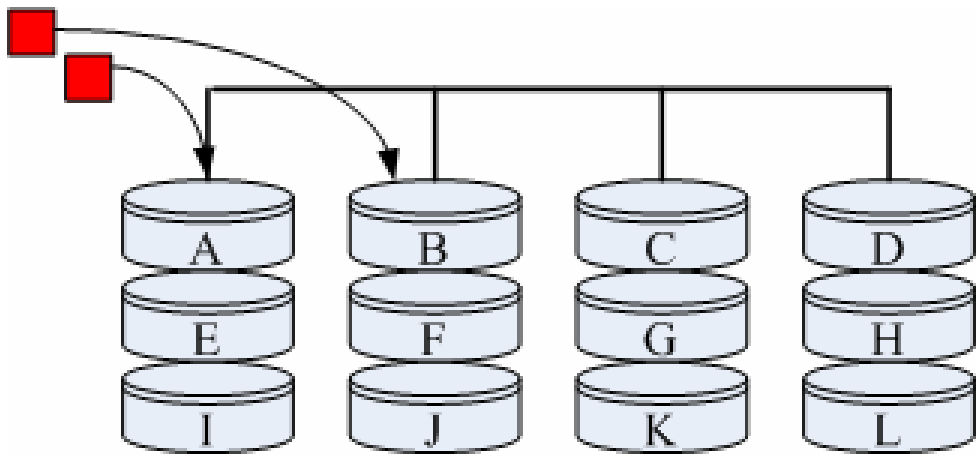
- 磁碟陣列
  - Redundant Array of Independent Disks，RAID
  - 主要的功能：
    - 將多個磁碟的容量整合成為一個大磁碟
    - 改善磁碟的讀/寫效能
    - 改善磁碟的資料安全性
      - 此安全為資料不會被損毀，而不是指網路security方面的安全概念。

磁碟陣列

- 不同等級的RAID技術：
  - RAID 0
  - RAID 1
  - RAID 2
  - RAID 3
  - RAID 4
  - RAID 5
  - RAID 6
  - RAID 0+1

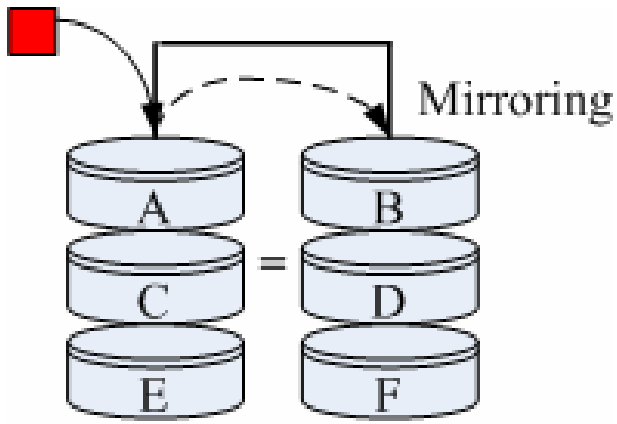
RAID0

- 技術： Striped Disk Array without Fault Tolerance
- 磁碟總容量：不變
- 效能：增加非常多
- 資料安全：非常差
- 需要硬碟數：至少兩顆



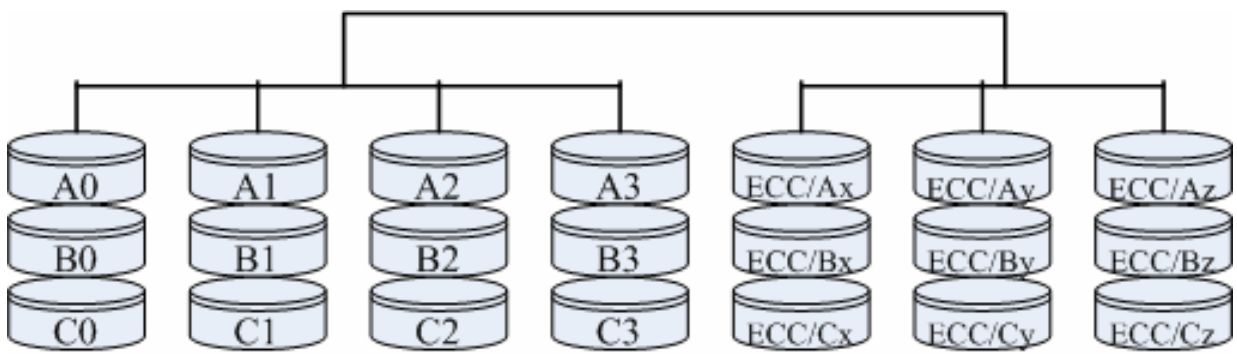
RAID1

- 技術： Mirroring & Duplexing
- 磁碟總容量：少一半
- 效能：幾乎不變
- 資料安全：非常好
- 需要硬碟數：2的倍數



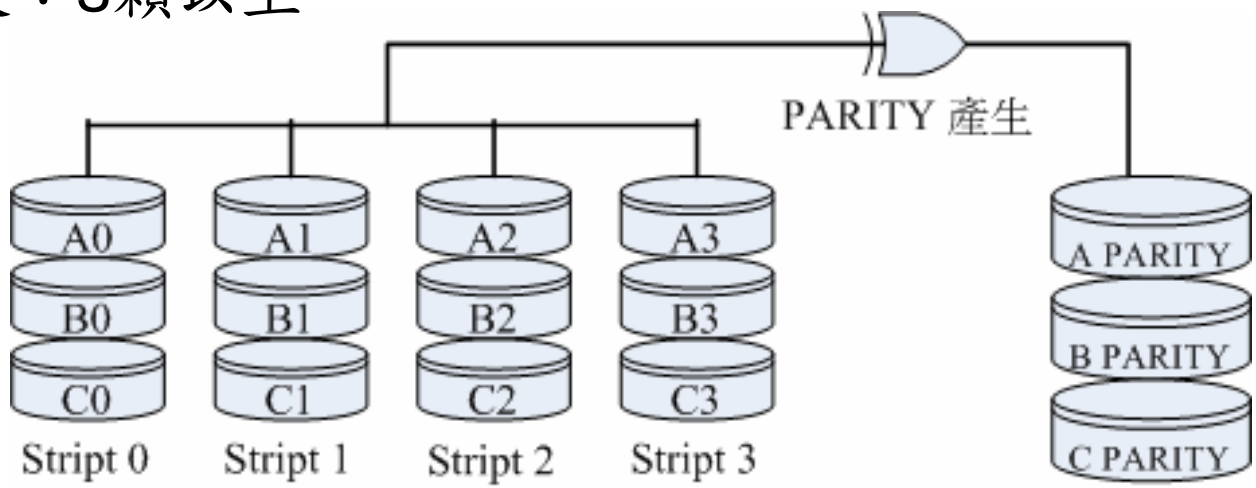
RAID2

- 技術： Hamming Code ECC
- 磁碟總容量：少一顆
- 效能：讀快/寫慢
- 資料安全：佳
- 需要硬碟數：3顆以上



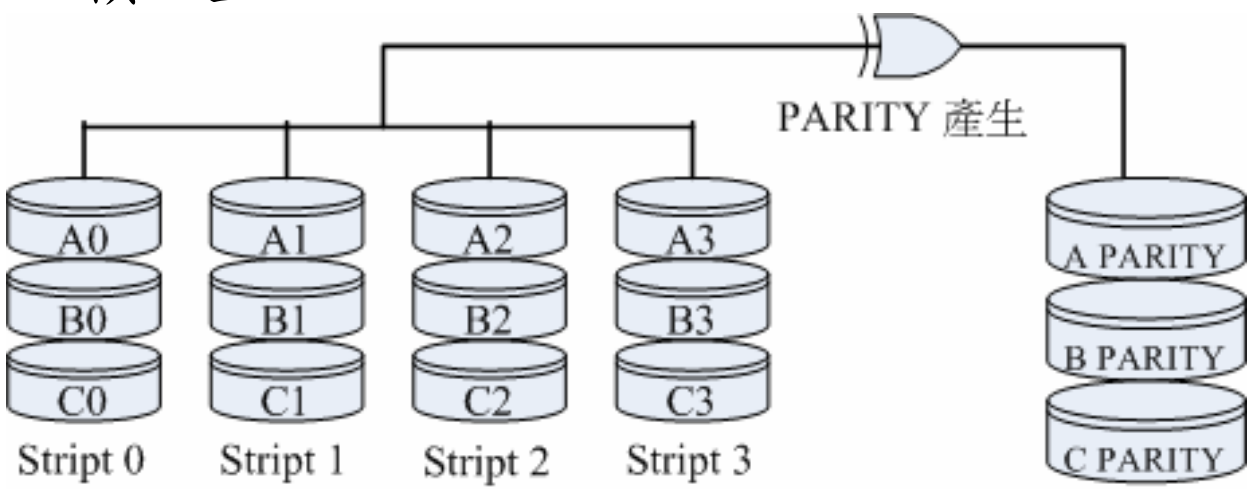
RAID3

- 技術： Parallel Transfer with Parity
- 磁碟總容量：少一顆
- 效能：讀快/寫慢
- 資料安全：佳
- 需要硬碟數：3顆以上



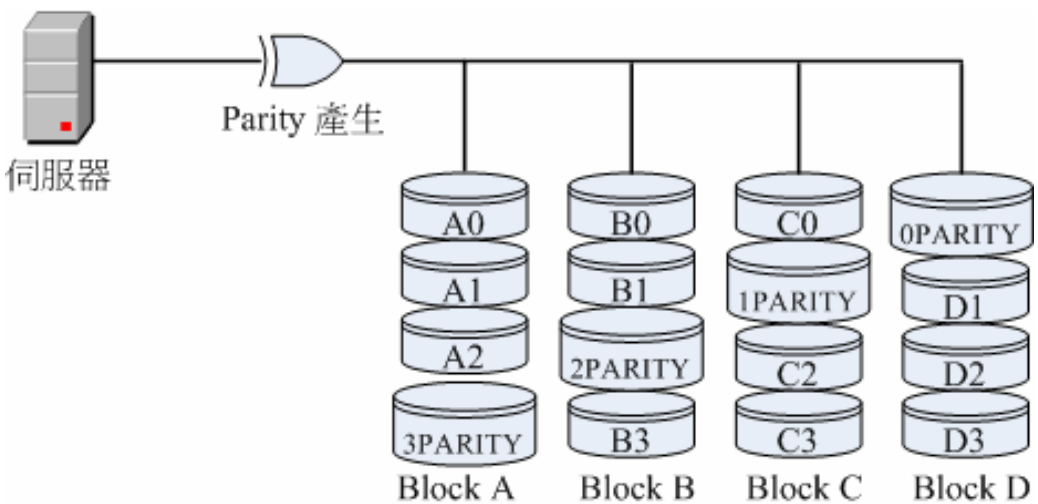
RAID4

- 技術： Independent Data Disks with Shared Parity Disk
- 磁碟總容量：少一顆
- 效能：讀快/寫慢
- 資料安全：佳
- 需要硬碟數：3顆以上



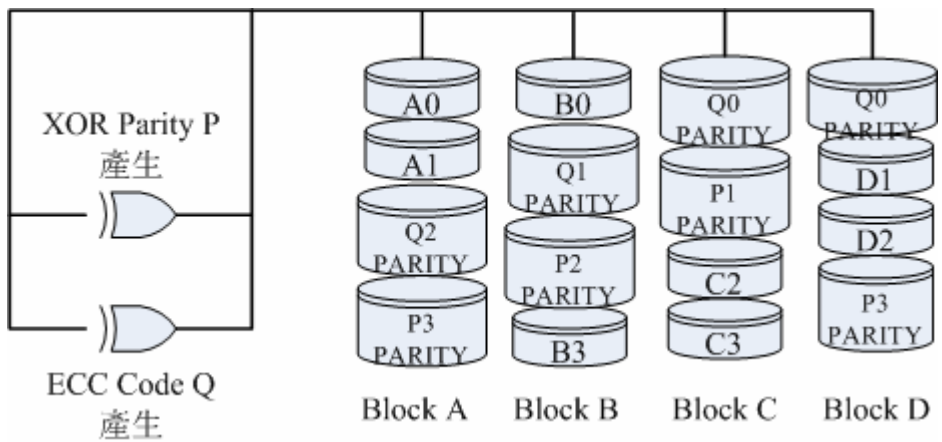
RAID5

- 技術： Independent Data Disks with Distributed Parity Blocks
- 磁碟總容量：少一顆
- 效能：讀快/寫慢
- 資料安全：佳
- 需要硬碟數：3顆以上



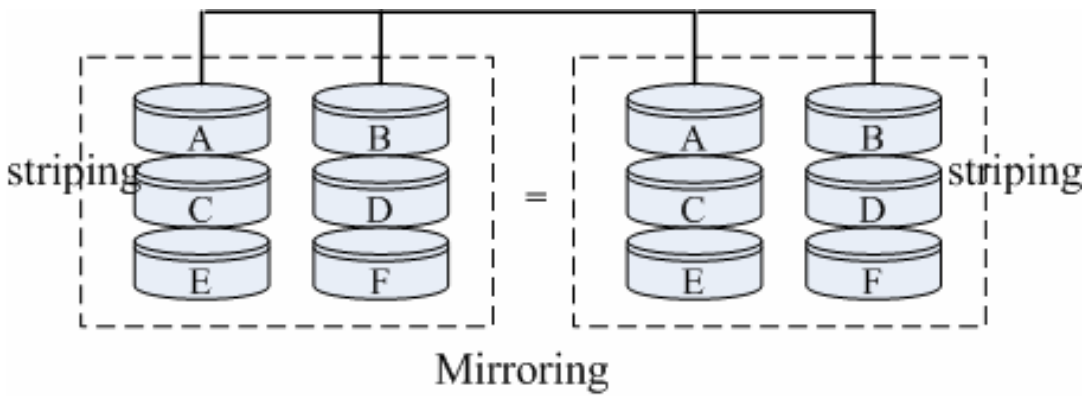
RAID6

- 技術： Independent Data Disks with Two Independent Distributed Parity Schemes
- 磁碟總容量：少兩顆
- 效能：讀快/寫慢
- 資料安全：佳
- 需要硬碟數：3顆以上



RAID0+1

- 技術： High Data Transfer Performance
- 磁碟總容量：少一半
- 效能：佳
- 資料安全：佳
- 需要硬碟數：四顆以上



RAID方案	所需磁碟數	可用容量	效能	容錯能力
JBOD	> 2	全部	不變	無
RAID 0	> 2	全部	最高	無
RAID 1	2	N/2	稍有提升	有
RAID 2	> 3	N-1	讀快寫慢	有
RAID 3	> 3	N-1	讀快寫慢	有
RAID 4	> 3	N-1	讀快寫慢	有
RAID 5	> 3	N-1	讀快寫慢	有
RAID 6	> 3	N-2	讀快寫慢	有
RAID 0+1	4的倍數	N/2	高	有

磁碟排程

- 原因：讀/寫磁碟上的資料會花費時間
  - 所以需要進行排程方面的最佳化，讓讀寫達到較佳效能

- 存取磁碟上的區塊資料所需花費的時間有：
  - 搜尋時間（Seek Time）
    - 磁碟移動到實際區塊上的時間
  - 旋轉時間（Rotational Latency Time）
    - 等待磁碟盤轉動的時間
  - 傳輸時間（Transfer Time）
    - 將資料複製出來，或寫入到磁碟區塊中的時間

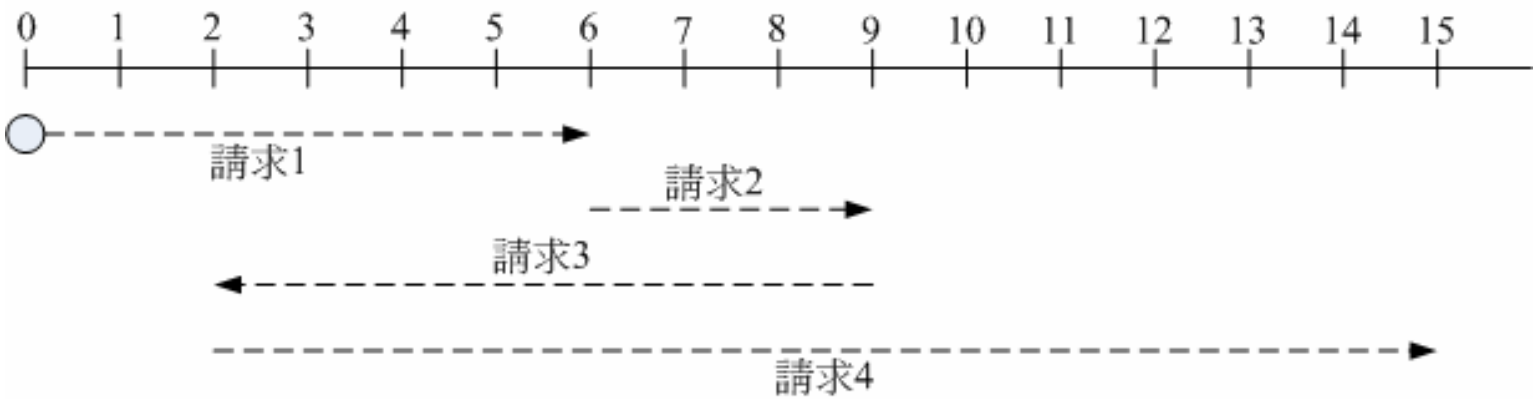
磁碟排程

- 磁碟效能的提升
  - 磁碟盤的搜尋時間與旋轉時間要越低越好
  - 磁碟轉動為同一方向，針對資料放在不同磁柱的資料讀寫，可透過讀寫頭排程器來處理。原則為：
    - 以磁碟整體效能為考量
    - 對所有同步存取需求的程序要具備公平性
    - 所執行的排程演算法需耗費的成本與效益考量

磁碟排程演算法-1

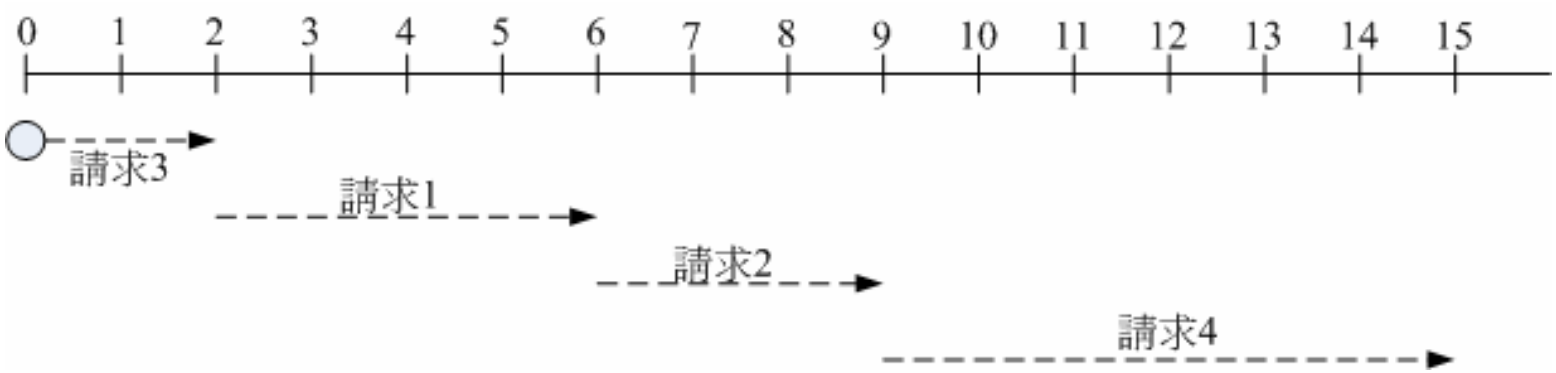
- 假設磁碟與存取需求環境
  - 讀取頭位於 0 位址
  - 資料要求的位址順序：6, 9, 2, 15

- 先來先服務（First In First Served，FIFS）
  - 0 →6→9→2→15的順序沒變；
  - 經過的磁柱總數：0+6+(9-6)+(9-2)+(15-2)=29



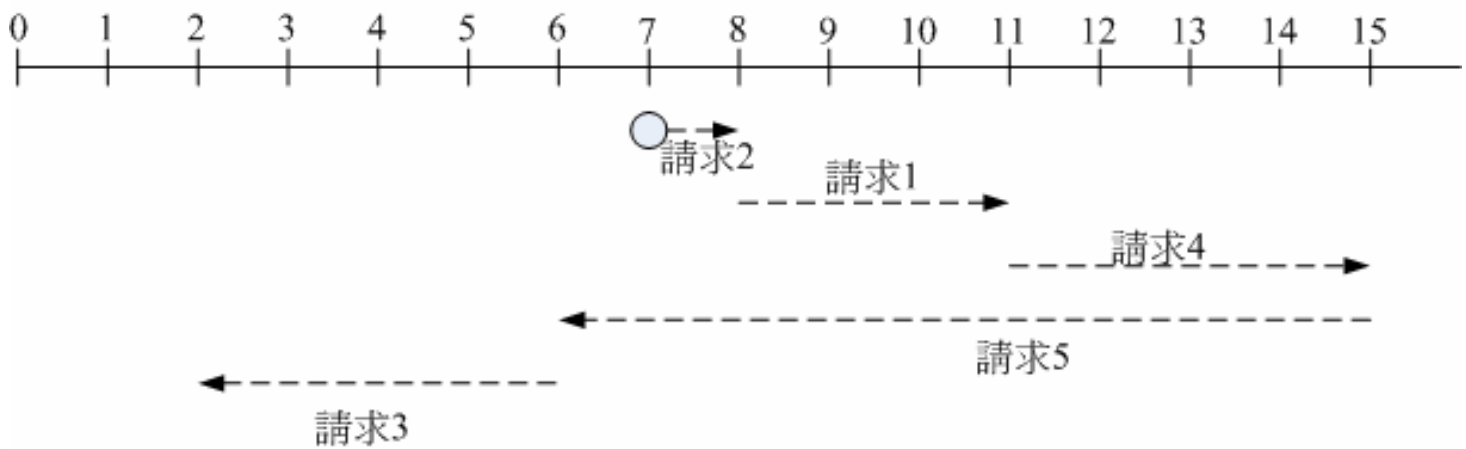
磁碟排程演算法-2

- 最短搜尋時間優先（Shortest Seek Time First，SSTF）
  - 先將磁碟請求的位址加以排序
  - 依據上述的排序來分別讀取，不是用佇列的順序喔
  - 經過磁柱數：0+2+(6-2)+(9-6)+(15-9)=15



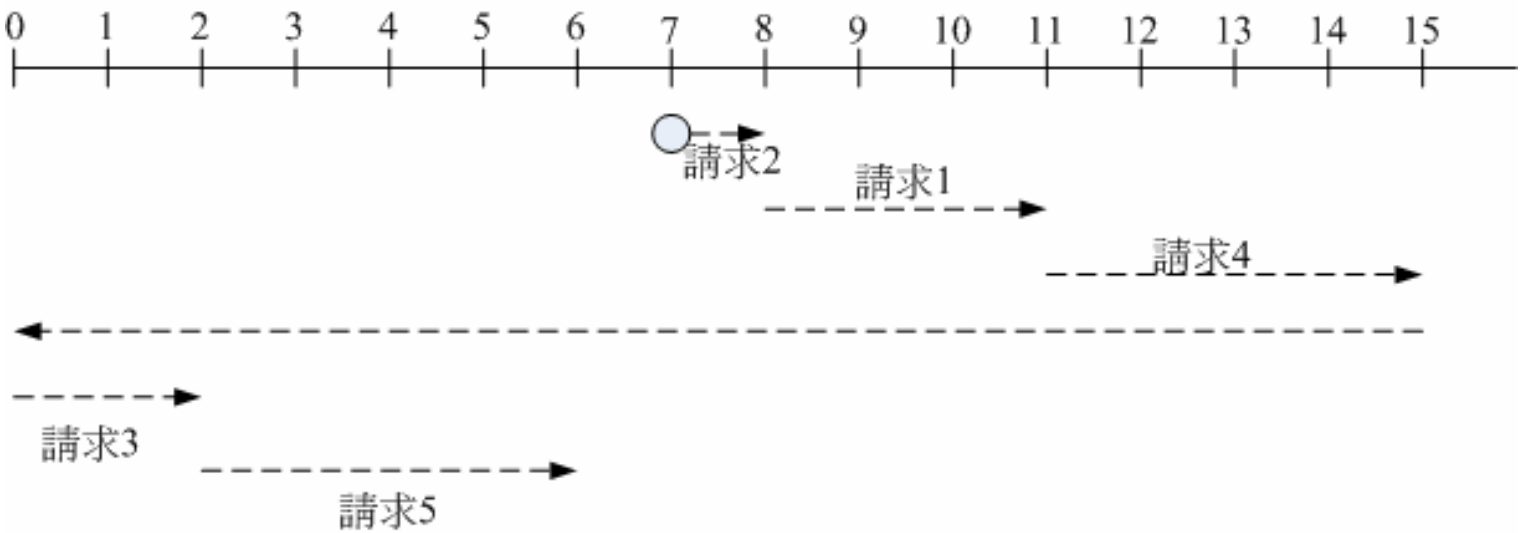
磁碟排程演算法-3

- 掃描法（Scan）
  - 改良SSTS的程序不公平而來的
  - 作法：讓磁頭在頭/尾磁柱兩端進行來回讀取的方式
  - 模擬環境：原始在7，要求有11, 8, 2, 15, 6等磁柱則
    - 7→8→11→15→6→2 (由小到最大，再到最小)



磁碟排程演算法-4

- 循環式掃描法（C-Scan）
  - 作法：讓與掃瞄法類似，但磁頭朝一固定方向存取
  - 模擬環境：原始在7，要求有11, 8, 2, 15, 6等磁柱則
    - 7→8→11→15→0→2→6 (由小到最大)



磁碟與目錄樹的使用

- 使用磁碟內的資料過程：
  - 1. 進行分割，或建立大檔案
  - 2. 進行格式化
  - 3. 建立掛載點
  - 4. 開始掛載

Linux磁碟檔名

- 裝置目錄：/dev/
  - 所有磁碟裝置的檔案都位於/dev目錄底下。
- 磁碟的檔名：
  - IDE硬碟：如右所示
  - SATA/USB/SCSI
    - /dev/sd[a-p] →依序

	Master	Slave
IDE-1	/dev/hda	/dev/hdb
IDE-2	/dev/hdc	/dev/hdd

Linux磁碟分割槽代號

- 以第一顆SATA硬碟為例(/dev/sda)
  - partition table在第一個磁區，只能記錄四筆
    - Primary (主要分割槽)：最多四個
    - Extended(延伸分割槽)：最多一個
      - P+E 最多也是只能有四個
    - Extended還能被分割出邏輯分割(Logical)
      - SATA類型的，邏輯分割最多到15號
      - IDE類型的，邏輯分割最多到63號
  - 第一個邏輯分割槽：/dev/sda5
    - 尚有其他/dev/sda[1-15]

Linux磁碟分割表查閱

- 利用 fdisk 來進行分割與查詢
  - fdisk /dev/sda
  - fdisk –l /dev/sda
- 建立大型檔案：
  - dd if=/dev/zero of=/tmp/ext3.img bs=1M count=100

進行格式化

- 格式化指令： mkfs [-t type] [裝置/檔案]
  - mkfs -t ext3 /dev/sdb5
  - mkfs –t ext3 /tmp/ext3.img　　格式化大檔案
  - mkfs –t vfat /dev/sdc1　　格式化隨身碟

建立掛載點

- 目錄樹與掛載點
  - 使用者只需要知道目錄樹位於何處
  - 磁碟透過掛載點連結到目錄樹架構
  - 掛載點為『某一目錄』
  - 掛載點建立：
    - mkdir /mnt/某個目錄
    - mkdir /mnt/loopdev

實際掛載

- 掛載指令： mount
  - mount [裝置] [掛載點]
  - mount –o loop [大檔案] [掛載點]
  - mount –o loop /tmp/ext3.img /mnt/loopdev
- 查閱檔案系統的指令
  - df [-ih]
  - mount
- 卸載的指令
  - umount [裝置/掛載點]

本章重點回顧

- 了解磁碟機基本的構造與磁碟緩衝和快取的功用。
- 了解當磁碟發生錯誤時的處理模式。
- 了解常見的磁碟陣列型態與其特性。
- 了解常用的磁碟排程演算法與運行方式。
- 透過Linux作業系統的磁碟管理工具，來說明磁碟裝置的管理實作。