- Memory management
  - Binding
    - 定義
      - 決定程式執行的起始位置
    - Binding 時機
      - Static
        - Complier Time
        - Loading/Linking Time
        - 執行期間不可變動位置
          - 沒有位置轉換的問題
            - 不需硬體額外支援
        - 快速
      - Dynamic
        - Execution Time
        - 執行期間動態決定
          - 執行期間，位址可以變更
            - 轉換部分由 MMU(memory mgt. unit) 負責
              - 即：需要硬體的額外支援
            - 位址還要轉，執行效率就差了
          - 位址有邏輯位置、實體位置之別
        - 彈性
  - Dynamic Loading
    - 程式經過分割成片段，需要(被呼叫)的時候才載入
      - 也就是早期的 overlay 技術
      - Programmer 額外負擔
      - 沒有 O.S. 額外支援
      - 效率差
  - Dynamic Linking
    - 當模組被呼叫的時候，才載入並 linking 修正
      - 適用 Dynamic Library Linking (DLL)
      - 節省 load & linking time
      - 需要 O.S. 額外支援
        - linking 其他 process 的資料或函式的時候，process 之間必須經由 O.S. 請求/存取
  - Contiguous Memory mgt.
    - First Fit
      - 時間複雜度最低
    - Best Fit
      - 挑破碎最小的配置
      - 需要搜尋所有 list
      - 空間利用度最佳
    - Worst Fit
      - 挑破碎最大的配置
      - 需要搜尋所有 list
  - Fragmentation
    - External Fragmentation
      - 外部破碎
      - 例
        - 連續性配置、分段
      - 解決
        - Compaction
          - 壓縮、聚集
          - **必須是 Dynamic Binding**
        - Page
        - 多套 Base/Limit register for code sec. and data sec.
    - Internal Fragmentation
      - 內部破碎
      - 要五毛給一塊
      - 例
        - Paging
      - 解決
        - 減少分頁大小
  - Page memory mgt.
    - 概念
      - 1. 實體記憶體先劃分成多個 frame
        - 每個 frame 大小相同
      - 2. 虛擬的記憶體空間也畫分為多個 page
        - page 與 frame 大小相同
        - i.e. 行程佔用的記憶體空間以”頁”為單位
      - 3. page 與 page 之間不需連續配置
        - 行程佔用的”頁”，對應到實體記憶體中任意的框
          - 大概跟指標變數動態配置空間的概念類似
          - 指標變數依址取值
      - 4. process 私有一個 page table
        - 紀錄 page ID 對應的 frame ID
        - PCB 中儲存資訊的其中之一
    - 優點
      - 沒有外部破碎
      - 支援 memory sharing
        - 行程共享的部分指到同一個 frame 即可
      - 支援 memory protection
        - page table 加上 protection bit 紀錄 R/W 等
    - 缺點
      - 會內部破碎
        - 最糟情況
          - n pages + 1 Byte
            - 也就是說 **page 越大，破碎越嚴重**
      - 需要硬體支援
        - page table 製作、儲存
        - MMU 轉換位置
      - 有效存取時間拉長
        - 先存取分頁表，再存取資料
    - page table
      - 用 register 儲存
        - 快
        - 無法適用太大的 page table
      - 用 memory 儲存
        - 搭配一個 page table base register
        - 慢
        - 支援大 page table
      - TLB (Translation Lookaside Buffer)
        - 儲存”經常存取的”部分 page table
        - 重點
          - 對應關係＆示意圖
          - 有效存取時間等計算
    - 位址轉換
      - 邏輯位址
        - 依照 page size，切分為 p, d 兩個部分
        - page size 直接關係到 d 的 bits 數
          - 且 page 與 frame 大小相同
          - 所以跟實體位址中的 d 是同樣的
        - process size 直接關係到 p 的 bits 數
          - 也就是對 process 來說佔用了多少 pages
          - 或者 process 可佔用的最多 pages
      - 實體位址
        - 依照 page table 紀錄的 frame ID 與邏輯位址的 d 合成可得
          - 因為 frame 是實體記憶體的劃分
          - 有 frame ID 就知道是第幾個頁框
    - paging 重點考題
      - 通常需要搭配一個 entry 所需大小的資訊
      - page table size 計算
        - page table size 太大
          - 解決方法
            - Hierarchical Paging
              - 邏輯位址再切分為 P1, P2 兩部分
                - 依 P1 在第一層分頁表存取第二層分頁表
                - 依 P2 在對應的第二層分頁表中查到 frame ID
                - 合成取得實體位址
              - 有確實減少分頁表的佔用記憶體空間
              - 存取一次分頁表就是一次記憶體存取
                - 記憶體存取次數上升
              - aka.
                - multilevel paging
                - paging page table
                - forward-mapping page table
            - hashing page table
              - 略
            - inverted page table
              - 反轉在哪
                - 以實體記憶體為紀錄對象
                - 每個 entry 中紀錄實體記憶體被分配給哪個 Process 的第幾個 Page
                - <PID, page No.>
              - 有效解決分頁表太大的問題
              - 但是搜尋很長很慢
              - 喪失了 memory sharing 的優點
      - 畫圖？
  - Segment memory mgt.
    - 概念
      - 將 Process 分段
        - e.g.
          - code segment
          - data segment
          - stack segment
      - 一個”段”必須倍連續配置
      - 段與段之間可不連續
    - 實際操作
      - kernel 以 segment table 紀錄
        - 紀錄資訊分別有
          - base
          - limit
      - 邏輯位址劃分為 s, d
        - 每個段的大小不盡相同
          - 需檢查是否為合法存取
            - d < limit
    - 討論
      - 有外碎無內碎
      - 可支援 memory sharing/protection
      - 行程分段的部分較難實施
      - 需要硬體支援
        - 分段表
      - 需要另外檢查合法存取與否
        - 存取時間長