競爭與同步解決方案

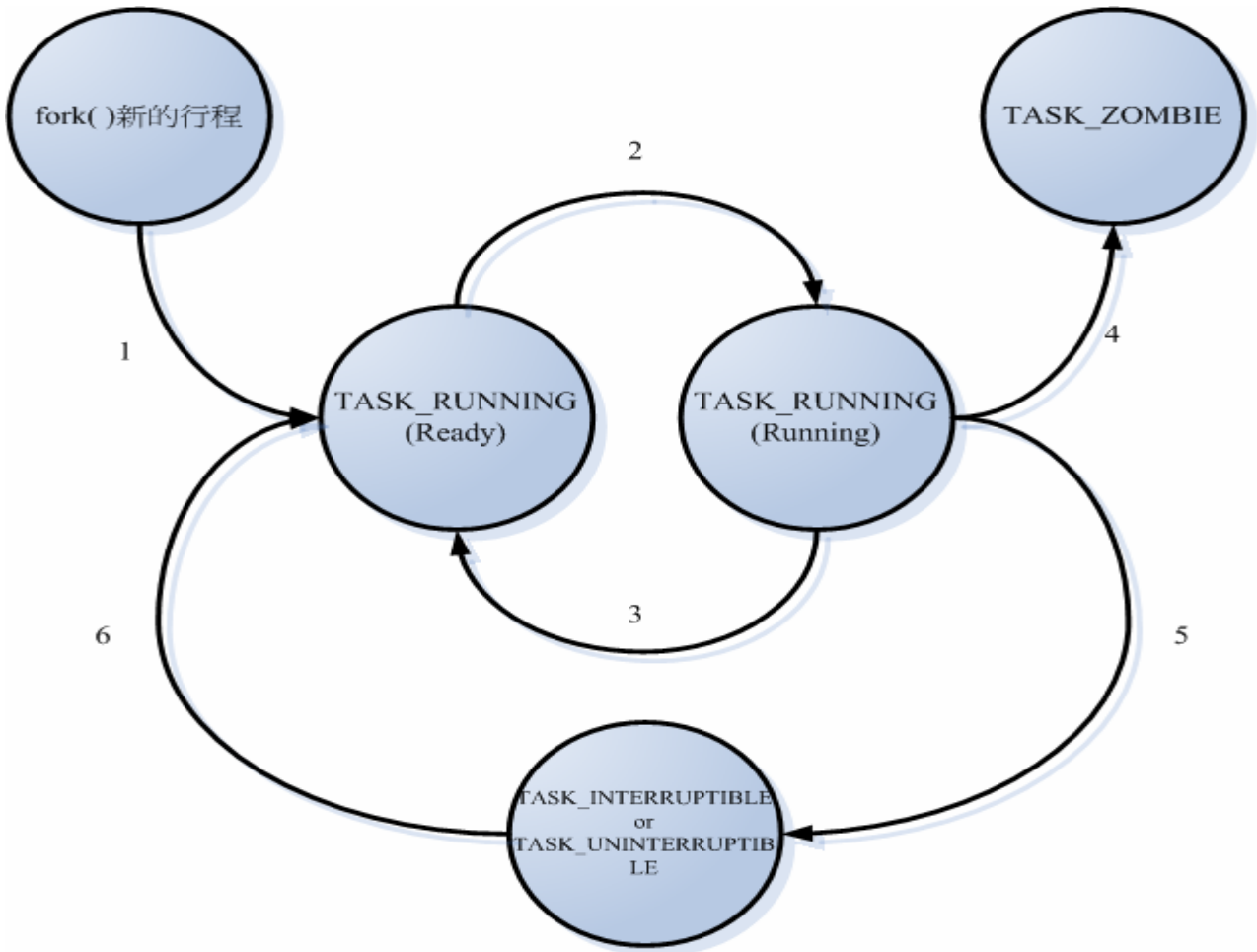
大綱

- Linux核心提供的號誌應用
- 死結的預防與處理
- 死結的避免與銀行家演算法
- 忽略死結與死結狀態恢復
- 核心的鎖定與先佔式的失效
- 本章重點回顧

號誌的運用

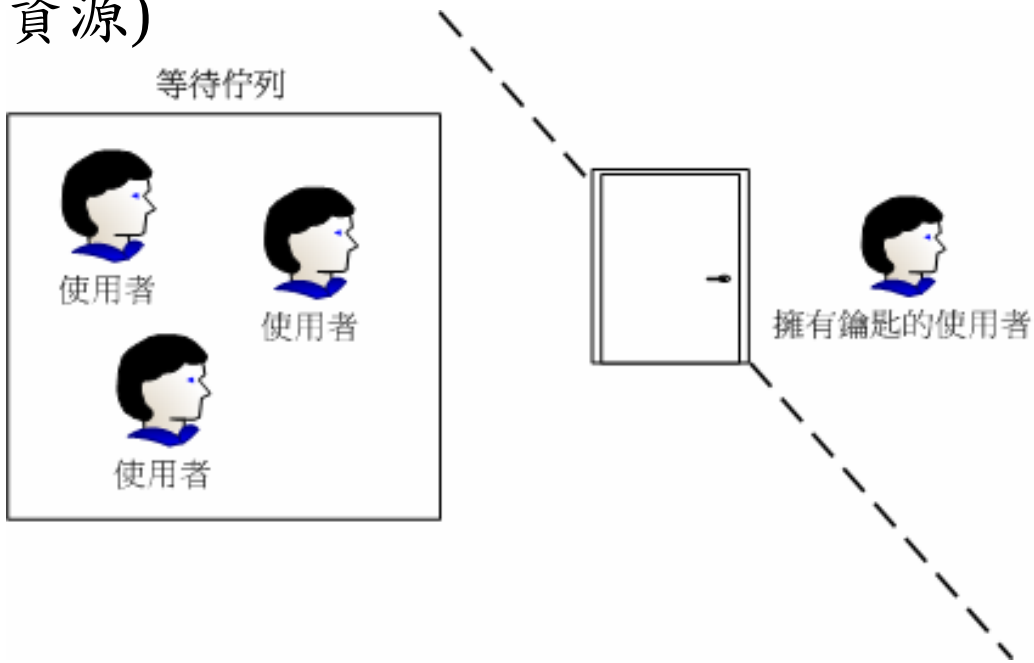
- 同步技術可能的問題：程序很忙碌
- 解決的方案：
  - 讓程序進入等待狀態，該狀態在休眠中
  - 處理器不會理會休眠的程序
    - 所以有較多資源可以運作其他程序所需

- 如下圖中，第五個步驟就是給予休眠的號誌！



Linux核心提供的號誌應用

- 號誌的運用 - 門與鑰匙範例
  - 只有一把鑰匙(臨界資源)
  - 每次只能一人使用
    - 每個人(程序)



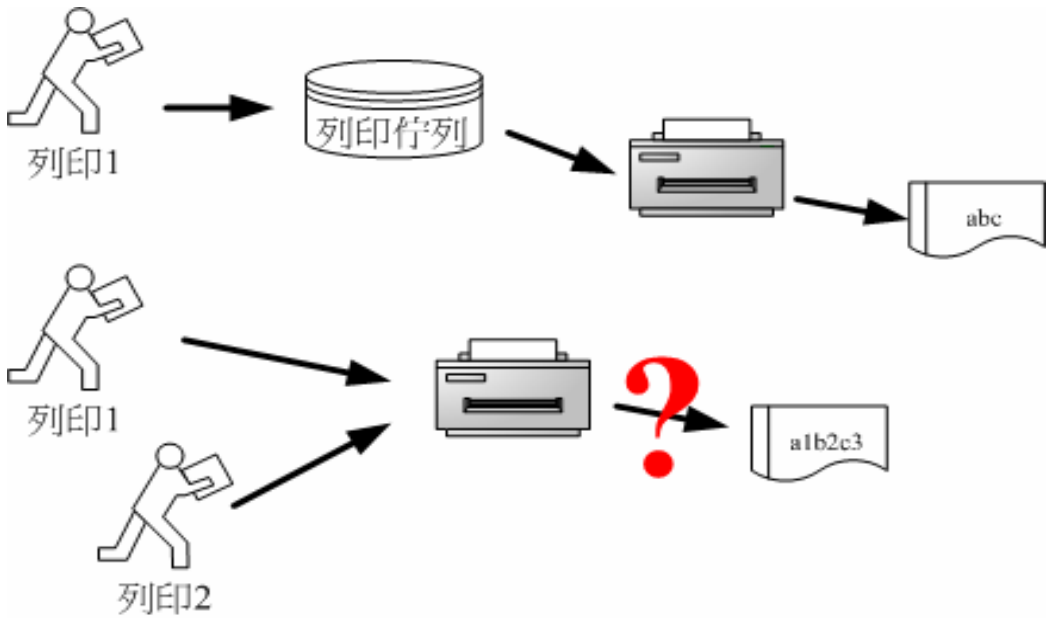
程序在休眠的情況

- 程序被放入等待佇列中，必須等到被喚醒才能夠到Ready的佇列中等待執行，
  - 所以號誌較適合需較大鎖定時間的程序中
- 號誌不太適合需求較短鎖定時間的程序，反而會造成系統的額外負荷(一直在喚醒)
- 號誌是讓程序的執行被鎖定，而非中斷程序
- 當另外一個程序獲得相同的休眠號誌時，原先的程序不會造成死結，依舊在等待佇列中休眠

死結的預防與處理

- 作業系統在處理死結的四種方式：
  - 預防（Prevention）→不允許死結的產生
    - 掌握系統，讓四個必備條件之一不會發生即可
  - 避免（Avoidance）→預先假設死結的發生狀態
    - 針對某些可能發生死結的資源進行限制，以免發生死結
  - 偵測與回覆（Detection and Recovery）
    - 定時或偶發的檢查系統內是否有死結，若有，則嘗試結束程序
    - 但會讓CPU撥出一些時間來處理這個程式段
  - 完全不理
    - 完全不理會，直到系統當機，並重新啟動後才恢復正常...

- 主要預防方法：就是確保作業系統於運行期間中，<u>**四大條件並不會同時成立**</u>。
  - 相互排斥的處理
    - 目的是讓不可共用的資源，變成可以共用
    - 但某些資源不能有共用的特性，例如印表機
    - 拿掉佇列，資料就錯亂了。
    - 此路不通！



- 佔用與等待 的處理
  - 原因：一個程序可能會佔用多個資源的情況
  - 解決：作業系統僅允許程序本身只能擁有一個資源的使用權，而不能同時具備多個資源的使用權。
  - 方案：
    - 要求行程執行前，須先對全部的資源進行請求並獲得分配。
    - 在請求額外的資源前，須先釋放原本持有的資源。
  - 缺點：
    - 系統的資源使用率會降低，甚至發生飢餓！

- 不可先佔 的處理
  - 讓作業系統的程序具有先佔式功能即可
    - 程序可以搶先取得資源使用權，將原本使用資源的程序手中搶過資源，以避免死結的發生。
  - 但系統資源必須要具備先佔式的特性(CPU,RAM是支援的！)
- 循環等待的處理
  - 針對所有的系統資源型態進行排序，使得作業系統僅允許程序可以對比目前有的資源編號更大的資源題出存取要求。
    - 因為資源有排序過，就不會產生循環等待的問題！

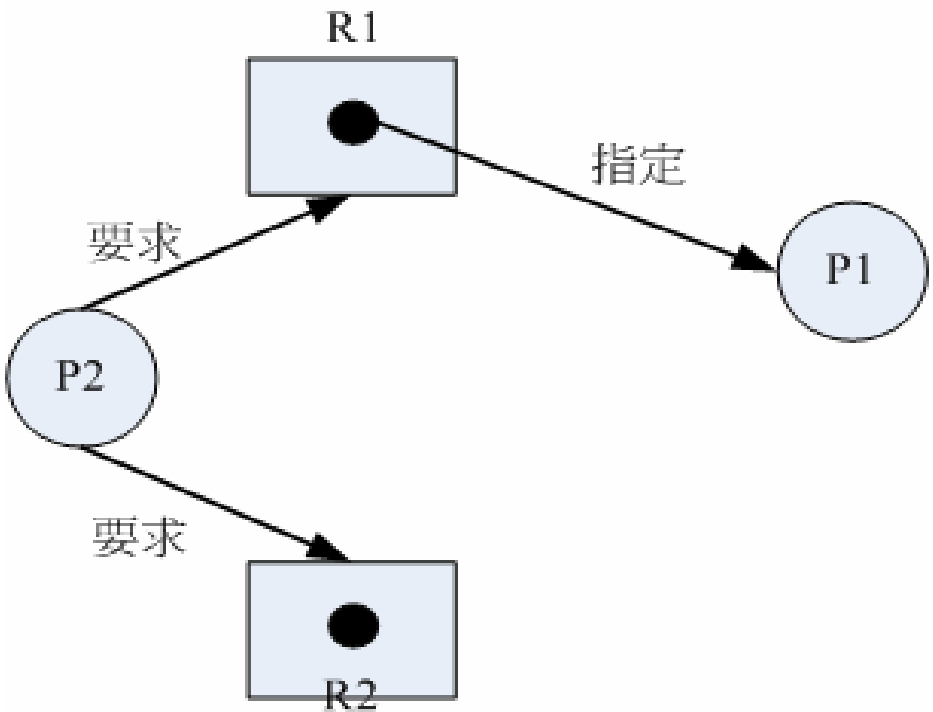
死結的避免與銀行家演算法

- 死結的避免（Deadlock Avoidance），就是作業系統本身可以依據
  - 現有可使用的系統資源，
  - 現有已被行程擁有的資源狀況，
  - 未來其他行程的要求，
  - 釋放佔有的資源
  - 種種因素的考量下，來決定是否同意目前提出資源存取要求的行程，是否可以擁有資源。

資源配置圖的使用

- 透過資源配置圖來決定安全與不安全
  - 安全：
    - 作業系統會依據某些次序分配資源給每個題出要求的程序，
    - 所有程序都能夠順利的執行完畢
  - 不安全：
    - 沒有這樣的次序分配情況，就稱為不安全

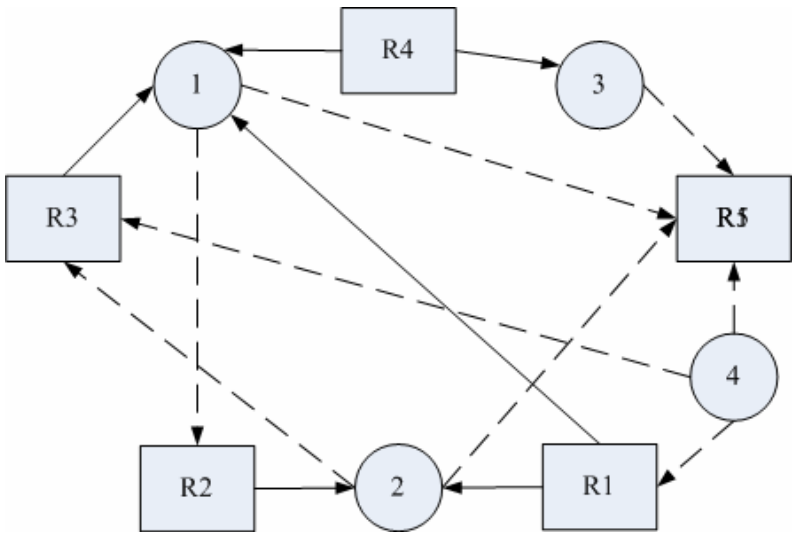
- 安全狀態的資源分配順序
  - 當作業系統呈現在安全狀態中時，系統是絕對不會產生死結。
  - 當作業系統呈現在不安全狀態中時，系統不一定會產生死結。
  - 死結的產生一定是由不安全的狀態中所衍生的。
  - 只要使作業系統呈現在安全狀態中，系統便可以避免死結的產生。



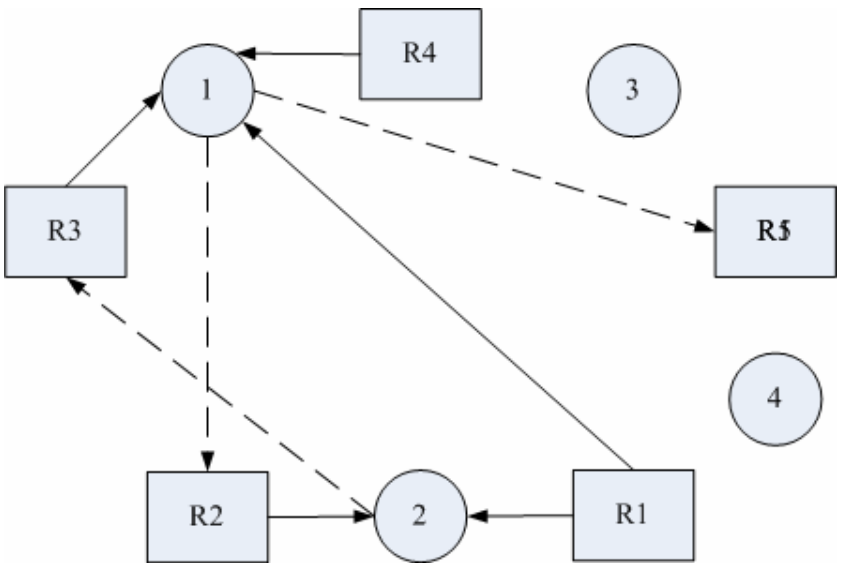
- 銀行家演算法（Banker’s Algorithm）
  - 銀行家就是作業系統本身，會隨時記錄：
    - 所有客戶(程序)
    - 的借款項目(已佔用的資源數量)
    - 最大可運用資金(目前可用的系統資源)
    - 找到一個順序(資源的配置序列)
  - 以滿足所有客戶的借款需求。若有此次序，則
    - 可保證任何時刻都能夠借錢(資源可以進行分配)
    - 也沒有人還不了錢(沒有死結的發生)

忽略死結與死結狀態恢復

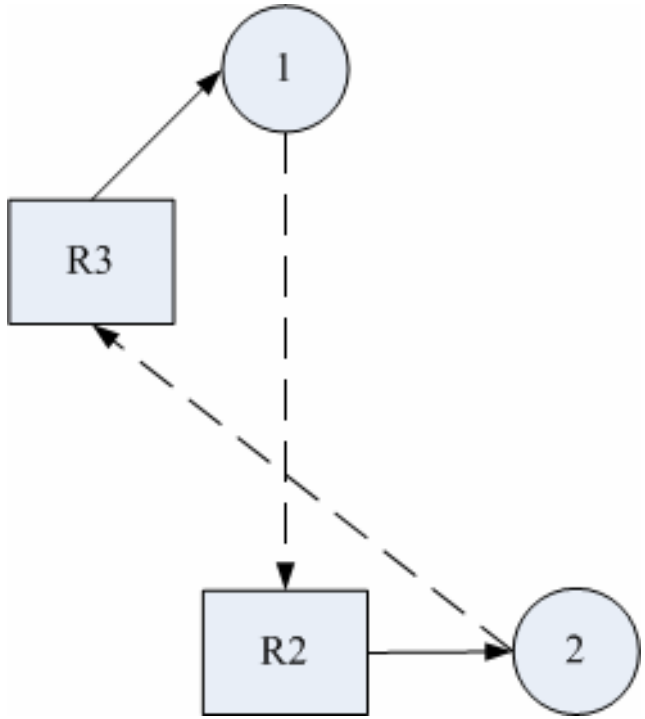
- 作業系統處理死結問題的另一個方法：
  - 定時檢查系統的中每個程序的狀態，
  - 偵測死結的模式，並處理這個死結現象



死結發生了



取消掉程序 3, 4看看



最終處理的結果

- 作業系統如何處理佔據資源的程序？
  - 結束造成死結的程序
    - 利用暴力法，結束這個死結的所有程序
    - 一個一個去除程序，但可能造成作業系統內部狀態的不一致性（Inconsistent）
  - 強制程序釋放佔有的系統資源
    - 讓程序回到原先的狀態，而釋放系統資源
    - 但每次都產生死結，用此方法就會產生飢餓（Starvation），因為該程序就是會產生死結嘛！

核心的鎖定與先佔式的失效

- 沒有一個適當的鎖定機制運行著的影響，便會在系統中產生競爭現象（Race Conditions）
- Linux核心中最普遍的鎖定機制 - spinlock
- 除了spinlock之外，Linux作業系統還提供一個透過kernel_flag所展現的核心鎖定機制

本章重點回顧

- 了解Linux作業系統核心中所提供的號誌運用方式。
- 了解到死結的預防與處理方式。
- 了解到死結的避免與銀行家演算法的運行方式。
- 了解到Linux作業系統提供的鎖定機制與先佔式功能的失效所造成的影響。