OS_110_Ch6

Background

- Concurrent access data 造成 data inconsistency
- 所以需要 ensure the orderly execution
- 當有 share memory 的時候，就會有可能有同步問題

Race Condition

- 結果的值會 depend on 最後一個修改值的人，則會造成中間的過程可能不正確

- 為了解決同步
  - 在 single-processor machine，可以用 **disable interrupt** or **non-preemptive CPU scheduling**
  - 大多用 **critical section problem**

Critical Section

- 為一個 **protocol**
- 問題：有多 processes 競爭 shared data
- 寫一個區塊，讓區塊只有一個 process 執行

Requirements

1. **Mutual Exclusion**：只有一個 process 可以在 CS 執行
2. **Progress**：要確保空的時候可以進去，且可以正常執行
3. **Bounded Waiting**：等待的時間是要有限的

Software Solution

- Peterson's Soloution for Two Processes 雙方同意
  - 有 flag 跟 token 可以確認對方要不要用

- Bakery Algorithm for N processes 像是抽號碼牌
  - FCFS
  - 有可能會抽到同樣的號碼牌，會比較 PID
  - 要有一個 choosing 的功能，判別是否再抽號碼牌
- Pthread Lock / Mutex Routines
- Condition Variables
  - 代表可以告知某些事情發生，可以開始執行其他事

Synchronization Hardware

- atomic instructions
  - as one uninterrupt unit
  - TestAndSet (var.), Swap (a,b)

TestAndSet

- 會回傳當下的 lock 狀況，且把呼叫的人鎖起來，下一個會收到已經鎖起來
- 離開的時候會解鎖

Swap

- 會將鎖頭交換給別人，給別人有機會執行
- 先 Call 的人就會拿到鎖

Semaphores

Classical Problems of Synchronization

Monitors

Atomic Transactions