Reviev:

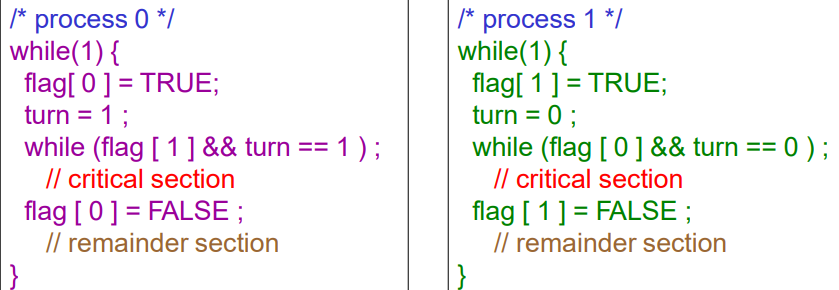
race condition: 共同競爭同一塊資源，counter++ 例子

\*解法:Critical Section演算法三要素

1. ME: 一次只能一個process在用
2. progress: 有權力，沒意願也要出來 (防止deadlock)
3. Bounded waited: 一定排的到 (防止straving)

\*CS演算法目的: 設計entry 與 exit

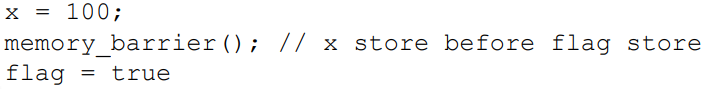
\*EX:兩個process的Peterson solution



Hardware Support:

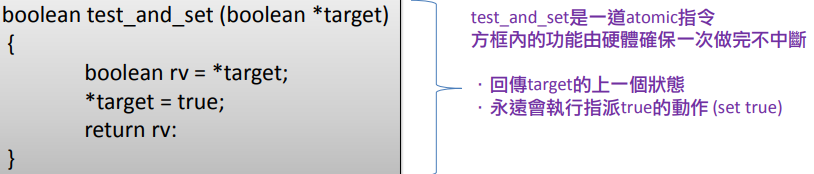
1. memory barrier:保證指令順序

在指令中間加上memory\_barrier()



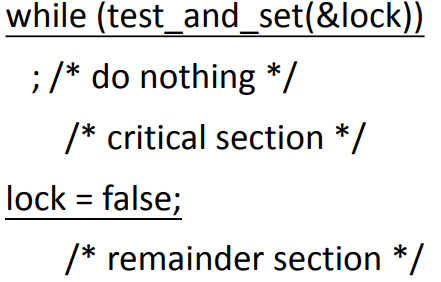
2.atomic instruction:保證不會被interrupt的基本單位指令

Test&Set(lock) : test看有沒有上鎖、將鎖set為true



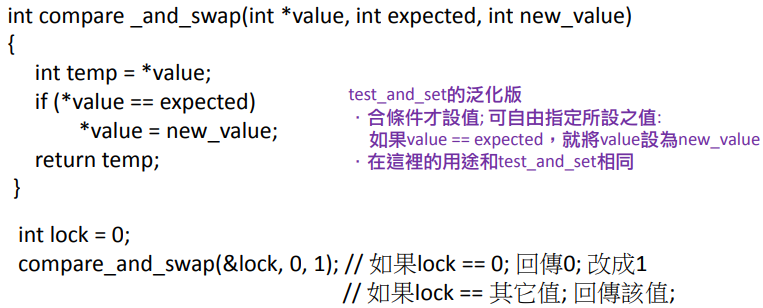
進去就上鎖、回傳上鎖前狀態

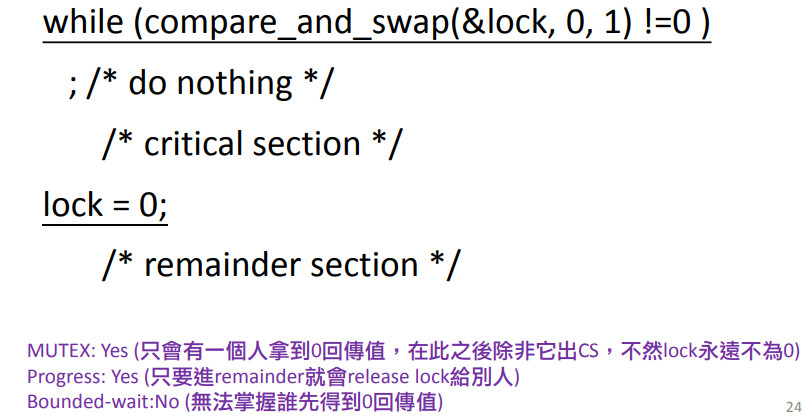
實作CS



注意: 沒有bounded waiting (像在搶廁所)

compare&swap(a,b) (多值版的test&set)





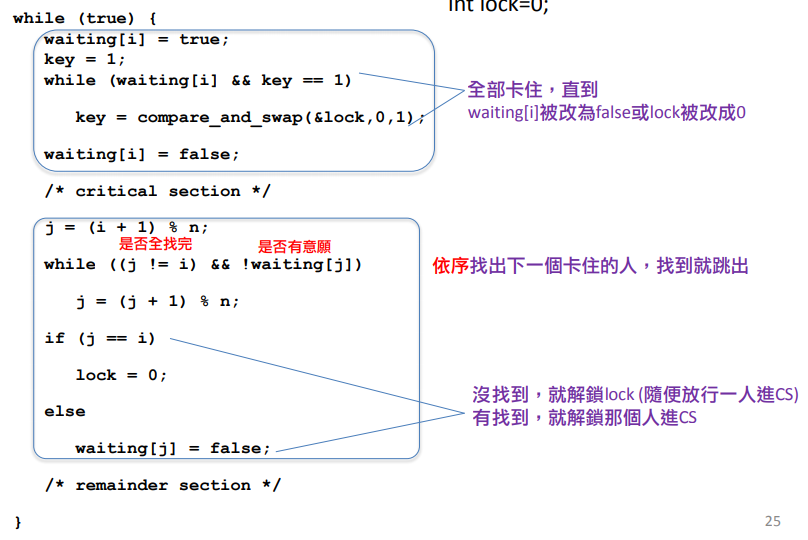
(比喻: 不斷輪流抽籤，抽到0才能進去)

\*Col: 第一個call function的人可進去; 出去時要解鎖

解決bounded waiting

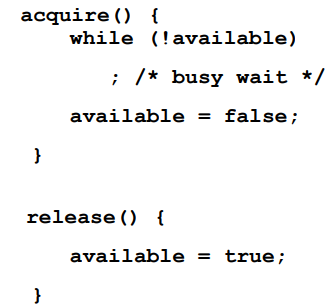
輪流選下一位進去

wating [n] 設為false => 可以進去 (會被上一個出去的人指定為false)

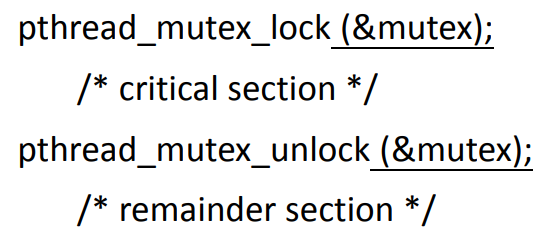


High level: 使用高階API來解決CS

Mutex: require & release ; 使用spin lock ; mutex 是 share data

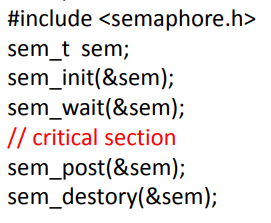


pthread實作



semaphore: 資源

計數器 + 隊列



Example: bounded buffer

