|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Process communication 的方式 | 名稱 | 定義 | 重點 | 特色 |
| Shared memory | Process之間透過其shared variables的存取，達到溝通的目的 | shared variables的互斥存取確保data的正確性 | 1. programmer負擔重，OS無額外負擔，只需提供Shared memory 2. 適用於大量的資料溝通 |
| Message passing | 1. 建立communication link 2. 訊息可互相傳輸 3. 傳輸完後釋放communication link |  | 1. 是OS的負擔(負責管理communication link、提供傳輸/接收指令)，programmer負擔輕 2. 適用少量的資訊溝通 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Race condition problem | 定義 | 解決法 | | 優點 | 缺點 |
| Shared memory communication若未對共享變數提供互斥存取的機制，最終值會因為process不斷的交錯執行順序不同而有所不同 | Disable interrupt  (to CPU) | Process對共享變數進行存取之前先Disable interrupt，等完成後再Enable interrupt可確保存取期間CPU不會被preemptive(此期間的atomically executed) | 1. simple、easy implementation 2. 適用單一CPU   (uniprocessors environment) | 1. 不適用multiprocessors system，因為只Disable一顆CPU不夠，其他CPU上的process仍可存取共享變數，但是若Disable所有CPU會造成system performance大幅降低 2. 風險高，user process可以disable interrupt有可能造成CPU長期被占用及其他無法緊急中斷的危害 |
| Critical section design  (to data control) | 對共享data / resource提供互斥存取的控制 | 1. 適用在multiprocessors環境 2. 風險較小 | 不適用在uniprocessors(比Disable interrupt麻煩) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C.S(Critical section臨界區間)  的基本認知 | 程式架構 | C.S design主要是在設計entry section和exit section的控制程式 | 設計原則 | Mutual exclusion | 任何時間點只允許一個process在自己的C.S內活動，不可多個process在各自的C.S內活動 |
| 1. process中存取共享變數的敘述/指令的集合 2. Remainder section(R.S)除了C.S以外的區間統稱R.S 3. Process內C.S可能有多個 |  | Progress | 1. 不想進入C.S的process(or在R.S內活動的process)不可以阻礙其他process進入R.S(or不可參與進入C.S的決策) 2. 想進入C.S的process，決定一個process可以進入C.S的時間有限(no deadlock) |
| Bounded waiting | Process提出申請到他進入C.S的等待時間是有限的(若有n個process想進入C.S，最多等(n-1)次)(no starvation) |