|  |  |  |  |
| --- | --- | --- | --- |
| 名稱 | 定義 | 範例 | 分析 |
| FIFO | 最早到達的process先取得CPU |  | 1. 排班效能最差 2. 有Convoy Effect(護衛效應) 3. 公平 4. No Starvation 5. No Preemptive |
| SJF | 具最小CPU time的process取的CPU，同時有多個以FIFO為準 |  | 1. 排班效能最佳 2. Starvation 3. 不公平 4. 分No Preemptive及Preemptive，一般而言是No Preemptive |
| SRJF  (SRTF  /SRTN) | 是Preemptive的SJF，剩餘的CPU time最小的優先，若新到達的process其CPU time小於目前的running time的剩餘CPU time，則可以插隊執行 |  | 1. 和SJF相比SRJF的avg waiting time較小，但context switching也較多 2. Starvation 3. 不公平 4. Preemptive |
| priority | 1. 具有highest priority的process優先取得CPU，權值相同以FIFO為準 2. 分No Preemptive及Preemptive兩種 |  | 1. 是一個可參數化的排班法則，即給予不同的優先權高低定義，可產生不同的排班行為 2. Starvation 3. 不公平 4. 有No Preemptive及Preemptive兩種 |
| RR | Time-sharing system採用，OS會規定一個CPU time Quantum值，當process取得CPU後，若未能在Quantum內完成工作，則timer會發出time-out，強迫process放掉CPU，給其他process使用，而此process須等到下一輪再取CPU，每一輪以FIFO為準 |  | 1. 常用在Time-sharing system 2. 公平 3. No Starvation 4. Preemptive 5. RR排班效能取決Quantum大小(也為一個可參數化的法則) |
| Multilevel  Queue | 1. 把原先一條ready queqe變成多層的ready queqe 2. Queqe跟queqe之間有排班法則(通常採用Preemptive priority) 3. 每個queue也有自己的排班法則(eg: RR FIFO) 4. Process一但置入某queue，就不允許在不同queue間移動 |  | 1. 可參數化的項目眾多，故排班設計效能能調整的flexibility大幅提升   (eg: queue數目 / queue之間的排班法則 / 每個queue的法則)   1. Starvation 2. 不公平 3. Preemptive |
| Multilevel  Feedback  Queue | 類似Multilevel Queue  差別，允許process在queue間移動(可採用Aging) |  | 1. 可參數化的項目眾多，故排班設計效能能調整的flexibility大幅提升   (eg: queue數目 / queue之間的排班法則 / 每個queue的法則)   1. 不公平 2. Preemptive |

* Convoy Effect(護衛效應)—很多process均在等待一個需要很長CPU time的process完成，才有機會取得CPU，造成平均等待時間大幅增加的不良效應
* Starvation(飢餓現象)—process因長期無法取的執行程序所需的各項資源，以致遲遲無法完成工作形成”Indefinite Blocking”現象。(容易發生在不公平的環境，如再加上Preemptive更容易發生)
* Aging(用來解決Starvation)—隨著process在system的時間逐漸增加，system也逐漸調升此process的priority

NOTE: soft real-time system不可採用Aging

|  |  |  |  |
| --- | --- | --- | --- |
| No Preemptive | 1. 除非執行中的process自動釋放CPU，其他process才有機會取的CPU(不可插隊，任意搶奪CPU) 2. Running 🡪 wait 3. Running 🡪 exit | 優點 | 缺點 |
| 1. process完成的時間點可以預期 2. context switching較少 3. 發生race condition問題較少 | 1. 平均等待時間長 2. 不適用real-time system / time-sharing system |
| Preemptive | 1. 執行中的process可能被迫放棄CPU(可以插隊 eg: time-out / interrupt / 高優先權插隊) 2. Running 🡪 ready 3. wait🡪 ready | 1. 平均等待時間長 2. 不適用real-time system / time-sharing system | 1. process完成的時間點可以預期 2. context switching較少 3. 發生race condition問題較少 |