OS-HW2-CPU Scheduling

資訊三甲 10927145 李俐瑩

一、開發環境:

Visual Studio Code (Python)

二、實作方法、流程:

每個排程法都使用差不多的程式碼來完成，差別只在於是否須用到 time slice，以及選擇執行 process 的優先順序為何。

- FCFS: (不可奪取)

  將 txt 檔讀進來後使用 list 存放資料，並在每個資料後面加上一個 cpuburst，代表剩餘的執行時間，以用來判斷是否已執行完畢。先將資料由 arrival time 由小到大排序，用一個迴圈跑到所有資料都執行完後停止，在迴圈內設立一個時間(time)，若有 data 的 arrival time 等於 time，就將這筆資料放入 ready queue，並設立一個名為 now 的 list 指向現在執行的 process，若 now 的剩餘執行時間==0，就在這筆資料後面加上 time-arrival time 代表 turnarround time，並取用 ready queue 的下一筆資料，取出紀錄 waiting time;若現在執行的還沒結束，就讓剩餘時間-1 並執行下一次迴圈，而每執行一次迴圈就記錄現在執行的 pid 繪製成甘特圖。

- RR(可奪取)

  和 FCFS 流程只有兩點不同，第一點是多設立了一個現在執行的

process 執行了多久，以用來判斷是否 timeout，若 timeout 就重新回到 ready queue 排隊。而第二點是 waiting time 改用結束時間-cpu burst time-arival time，其餘作法與 FCFS 相同。

- SJF(不可奪取)

SJF 最大的差別是使用了 cpu burst time 來決定執行的優先順序，且為不可搶奪的，其餘做法相同。

- SRTF(可奪取)

SRTF 則是改用剩餘時間來排執行的優先順序，且迴圈內要不斷判斷 ready queue 內是否有剩餘時間更小的 process。

- HRRN(不可奪取)

HRRN 改為在選擇下一位執行的 process 前，先計算反應時間比率 (arrival time+waiting time)/cpu burst time，計算完之後選擇反應時間最大者，若相同就選 arrival time 最小的，若相同在改用 pid 最小的，其餘做法也與前面差不多。

- PPRR(可奪取)

PPRR 改用 priorty 大小作排序，每次選擇都取 priority 最小的，若有相同的就採用 RR 的方式進行，且每次迴圈內都要判斷 ready queue 是否有 priority 更小的，若沒有就執行到結束，而若有相同的就執行到 timeout 後重新排隊。

三、不同排程法的比較:

平均等待時間: 五個不同排程法的 average waiting time 比較

	FCFS	RR	SJF	SRTF	HRRN	PPRR
Input1	14.333	18.4	8.867	8.067	11.6	14.667
Input2	8.4	6.4	8.2	3	8.2	9.4
Input3	6.667	11.667	6.667	6.667	6.667	12.5
Input4	3.75	5.5	3.5	3.25	3.75	4.5

同一檔案內 process 間的 turnarround time 使用不同排程法之比較

	FCFS	RR	SJF	SRTF	HRRN	PPRR
1	11	24	11	24	11	11
2	12	4	12	2	12	23
3	13	5	15	3	15	11
4	13	8	10	3	10	11
5	17	15	17	7	17	15

- FCFS:

  FCFS 的平均等待時間長短主要取決於 arrival time，若一個 process 抵達時可以剛好進入 running state，則等待時間就會變短，但若是太早抵達就要在佇列裡等待一段時間，等待時間就會跟著被拉長。而 turnarround time 就再加上一個 cpu burst time 來判斷，若 cpu burst

time 越高，turnarround time 也會跟著變高。

- RR:

  在四個 input data 內使用 RR 的平均等待時間幾乎都是最高的，根據他的特性可以知道，因為用完 time slice 後又要重新回到 waiting state，因此根據所有 process 的 cpu burst 和 time slice 相對之值，可以得知當 cpu burst 幾乎都小於 time slice 時可以減少平均等待時間。

- SJF:

  SJF 因為是不可奪取的，所以若是現在抵達的 process 擁有很大的 cpu burst time，那其他人一樣要在佇列裡等待很長一段時間，但因為每次都選擇執行最短的 process，所以其他 process 的等待時間會較低，因此使用 SJF 來執行基本上會擁有最小的平均等待時間。

- SRTF:

  SRTF 是可奪取的，因此若有執行更快的 process 就會被奪取，所以在 SJF 受到 arrival time 的影響在這裡就不會發生，因此可以比 SJF 擁有更短的平均等待時間。

- HRRN:

  這個排程法是用反應時間比率來選擇優先度的，因此等待時間越久反應時間就越大，就越有機會換他執行，因此平均等待時間也能減少。

- PPRR:

這個與 RR 的差別在於多判斷了一個 priority，因此與使用者設定 priority 有很大的關係。

四、結果與比較:

- FCFS:

這個是不可奪取的排程法，依照 arrival time 來選擇換誰執行，所以就算要等很久最終也一定會執行到，不會造成餓死的情況，但若是有一個擁有非常大 cpu burst time 的 process 檔在前面執行，那後面的 process 平均等待時間就會非常久。

- RR:

因為限制了 time slice，所以每個 process 都擁有機會輪流使用 CPU，但就會根據 time slice 大小的選擇來影響，若選擇過大就與 FCFS 沒什麼兩樣，而若是選擇太小又會影響到效率，時間都被 context switch 占滿。

- SJF:

此為不可奪取的排程法，且每次都選擇執行時間最短的，因此執行時間較長的 process 可能無法執行到，造成餓死的情況，但因為每次都選擇執行時間最短的，因此單看平均等待時間來比較，這會是一個最好的排程法。

- SRTF:

Waiting time 一定比 SJF 來的更小，因為改成了可奪取的，因此發現了更小 cpu burst time 的 process 就會被奪取，讓平均等待時間變更低。

- HRRN:

  因為增加了反應時間比率的條件，讓這個排程法不會產生餓死的情況。

- PPRR:

  Priority 是自己設定的，因此可以調整越需要的越快執行，可以根據使用著的偏好來選擇，但可能發生餓死的情況。

綜合以上敘述我認為各個排程法有好有壞，要根據不同的 process 來決定更適合使用何者。