OS_HW1 結果報告資訊三乙10727224 李敏嘉

一、開發環境及開發平台
(語言: C++)
IDE: DEV C++
Windows 10

二、實作方法和流程
以下五種方法皆有相同的規則:
- 所有方法都先依照Arrival Time先後次序排序。設一個Timer由0開始算起，每次都+1，代表過去了一單位時間。
- 當前時間以後才到達的Processes皆不能執行。
- 若該方法有使用Arrival Time先後次序作為優先等級，則若多個Process的Arrival Time相同，則比較Process 的ID，由小至大排序。
- 若該方法有使用Time Slice作為執行一個Process的時間，若Time Out的話，該Process將回到佇列尾端排隊。
- 當Process執行完成，則記錄完成時間，完成時間為當前時間+1。(當前時間為Process最後一單位時間的開始執行時間，所以當前時間再加一就是完成時間)
- Turnaround Time = 完成時間 – 到達時間
- Waiting Time = Turnaround Time – CPU Burst

五個方法皆有相同實作方法、Data structure:(如下)
- 因為不能確定資料數，所以使用動態陣列--vector存所有的Process。
- Vector中所存的每個Process都是一個struct，其中紀錄該Process的ID, CPU Burst，Arrival Time, Priority, Finish Time，Turnaround Time, Waiting Time。
- 使用string存目前執行的Process是哪個(此string就是甘特圖)。
- 用int timer 當計時器，每過一單位時間就+1，而每次被執行的Process，其CPU Burst就減一。直到其CPU Burst 為0，代表該Process執行完成。

FCFS(First Come First Serve)
流程:
1. 依照每個Process的Arrival Time先後次序排序，並將結果存入List中。
2. 從排好序的List中依序取出Process執行，此Process執行完成才換下一個Process執行。
3. 若Process執行完成，則紀錄完成時間，並存入fcfs_list中(作為輸出用)。
4. 待排好序的List中的所有Process執行完畢，計算fcfs_list中所有Process的Turnaround Time及Waiting Time。
5. 將fcfs_list依Process ID由小至大輸出結果。

實作方法、data structure:
- FCFS中有兩個vector，一個是負責排執行順序的sorted_list，另一個是輸出結果用的fcfs_list。每執行一個Process，則sorted_list的size便會減一，直到其size為0，表示所有Process皆執行過了。

RR(Round Robin)
流程:
1. 依照每個Process的Arrival Time先後次序排序，並將結果存入List中。
2. 從排好序的List(sorted_list)中依序取出Process執行，每次只能執行一個Time Slice，Time Out就回到佇列尾端排隊，並取下一個Process執行。
3. 若Process完成，則紀錄完成時間，並存入rr_list中(作為輸出用)。
4. sort_list中的所有Process執行完畢，計算rr_list中所有Process的Turnaround Time及Waiting Time。
5. 將rr_list依Process ID由小至大輸出結果。

實作方法、data structure:
- RR中有兩個vector，一個是負責排執行順序的sorted_list，另一個是輸出結果用的rr_list。sorted_list使用方式同FCFS的sorted_list。
- RR會多使用一個queue來存正在等待執行的Process(此queue代表Ready queue)，新到的Process皆會先到Ready Queue中(尾端)排隊。每次皆從Ready Queue中取top(第一個)Process執行。
- 有一個變數(integer)用來記錄該Process已經執行多久，若超過Time Slice則換要換Process執行，並重置此integer。

SRTF(Shortest Remaining Time First):
流程:
1. 依照每個Process的Arrival Time先後次序排序，並將結果存入sorted_list中。
2. 所有等待執行的Process皆從sorted_list取出，並存在ready_list中，每過一單位時間，就會從ready_list中挑選剩餘CPU Burst最小的Process出來執行。
3. 若Process完成，則紀錄完成時間，並存入srtf_list中(作為輸出用)。若尚未完成則再放回ready_list中。
4. ready_list中的所有Process執行完畢，計算srtf_list中所有Process的Turnaround Time及Waiting Time。
5. 將srtf_list依Process ID由小至大輸出結果。

實作方法、data structure:
- SRTF中有三個vector，一個是負責排執行順序的sorted_list，一個是存等待被執行的Process的ready_list，另一個是輸出結果用的rr_list。sorted_list同FCFS的sorted_list。
- 當有新來的Process，則從sorted_list中取出，並放至ready_list中。每次都存入ready_list的最後面，其擺放順序並不影響結果。(因為每次要挑Process來執行時，都會看過每個Process一遍)
- CountRemaining(): 負責找出ready_list中所有Process中 剩餘CPU Curst最小的那個Process，每次皆從頭開始找，每找到”目前”最小的Process就暫存起來，找到後，回傳該Process在ready_list中的index。此Function的時間複雜度:O(n)

PPRR (Preemptive Priority + RR)
流程:
1. 依照每個Process的Arrival Time先後次序排序，並將結果存入sorted_list中。
2. 所有等待執行的Process皆從sorted_list取出，並依priority找到相應的queue存入

queue(存在multi_queue中)。
3. 每次皆取multi_queue的第一個queue中的第一個Process執行，執行一次，若此Process尚未執行完成且Time Slice尚未用完，就放回原來的位置(原本queue的最前面)，但若是被奪取或是TimeOut，則放回原本queue的尾端。
4. 若Process完成，則紀錄完成時間，並存入pprr_list中(作為輸出用)。
5. multi_queue中的所有Process執行完畢，計算pprr_list中所有Process的Turnaround Time及Waiting Time。
6. 將pprr_list依Process ID由小至大輸出結果。

實作方法、data structure:
- PPRR中有兩個vector，一個是負責排執行順序的sorted_list，另一個是輸出結果用的pprr_list。sorted_list使用方式同FCFS的sort_list。
- PPRR還使用了一個multi_queue: 使用一個vector，vector中每個節點存的是一個queue，和該queue的priority，也就是說vector中存有多個不同priority的queue。該queue中存的是Process，同RR(Round Robin)的queue。
- 每個新來的Process會依照其priority找到相同priority的queue存入其中，若找不到則建立新的queue，並插入應該在的位置。(priority越小的queue會存在vector的越前面)，也因為需要使用"插入"的做法，所以選擇用vector來實作multi_queue
- 將新來的Process存入multi_queue時，會順便判斷它的priority是否優先於目前所執行的Process的priorty，並回傳一個布林-- isPreempted，表示是否被奪取。

HRRN (Highest Response Ratio Next)
流程:
1. 依照每個Process的Arrival Time先後次序排序，並將結果存入sorted_list中。
2. 所有等待執行的Process皆從sorted_list取出，並存在ready_list中。
3. 每次皆從ready_list中取出一個response ratio最大的Process來執行。
   response ratio = (waiting time + cpu burst) / cpu burst
4. 正在執行的Process執行完畢，才能再從ready_list中取出下一個response ratio最大的Process來執行。執行完畢的Process計算完成時間後，存入hrrn_list中。
5. ready_list中的所有Process執行完畢，計算pprr_list中所有Process的Turnaround Time及Waiting Time。
6. 將hrrn_list依Process ID由小至大輸出結果。

實作方法、data structure:
- HRRN中有兩個vector，一個是負責排執行順序的sorted_list，另一個是輸出結果用的hrrn_list。sorted_list使用方式同FCFS的sort_list。
- Ratio_Calc() : 此function負責回傳目前ready_list中Response Ratio最大的Process的index(在ready_list中的index)。

三、不同排程法的比較

數據input1執行結果: waiting time

ID	FCFS	RR	SRTF	PPRR	HRRN
0	19	18	0	0	19
1	13	8	0	0	5
2	22	19	2	14	16
3	18	25	6	0	14
4	13	19	0	11	13
5	20	27	19	21	23
6	0	15	6	11	0
7	15	2	0	55	3
8	21	14	0	9	11
9	5	13	1	0	6
10	8	37	49	45	18
13	18	3	0	0	4
20	13	17	0	40	13
27	16	28	19	10	9
29	14	31	19	4	20

平均等待時間:

FCFS	RR	SRTF	PPRR	HRRN
14.33333	18.4	8.066667	14.66667	11.6

SRTF < HRRN < FCFS < PPRR < RR

數據input2執行結果: waiting time

ID	FCFS	RR	SRTF	PPRR	HRRN
1	0	13	13	0	0
2	10	2	0	21	10
3	10	2	0	8	12
4	11	6	1	9	8
5	11	9	1	9	11

平均等待時間:

FCFS	RR	SRTF	PPRR	HRRN
8.4	6.4	3	9.4	8.2

SRTF < RR < HRRN < FCFS < PPRR

數據input3執行結果: waiting time

waiting

ID	FCFS	RR	SRTF	PPRR	HRRN
1	0	0	0	0	0
2	0	20	0	30	0
3	20	30	20	35	20
4	15	15	15	0	15
5	0	0	0	10	0
6	5	5	5	0	5

平均等待時間:

FCFS	RR	SRTF	PPRR	HRRN
6.666667	11.66667	6.666667	12.5	6.666667

FCFS = SRTF = HRRN < RR < PPRR

結果討論:

由上面數據可看出，PPRR平均等待時間會最長，而SRTF的平均等待時間一直都是最短的。

FCFS:平均等待時間在所有排程中不一定是最長的也不一定是最短的，若Process中存在CPU Burst時間很長的Process，平均等待時間就變得很長。

RR: 每次正在執行的Process能用的時間就只有一個Time Slice，Time Out就會取下一個Process執行。所以若有n個Process，則平均等待時間不會超過(n-1)個Time Slice。

SRTF:因為將CPU Burst較長的Process延遲執行，進而降低其他(CPU Burst較短的)Process的等待時間，所以可以提高整體平均等待時間。

PPRR:若先不看Priority的部分，其餘部分皆和RR排程法相同，而PPRR主要是在RR排程法的基礎上，先做優先權高的Process，但優先權高的Process的CPU Burst若很高時，則會讓較低優先權的process等待更久，進而拉長平均等待時間，所以這個排程方法會使平均等待時間較長。

HRRN: 每次都計算Response Ratio，Response Ration = (Waiting time + CPU Burst) / CPU Burst 。由公式可看出若Waiting time較長，則會提高優先等級，讓已經等很久的Process盡量不要再繼續延長等待時間，提早去執行;而CPU Burst較大的Process就會使其降低優先等級，類似於SRTF的方式，將低其他CPU Burst較短的Process的等待時間。這種方式雖然可以降低平均等待時間，但平均等待時間卻不是最小的，我想原因就出在它是Non-Preemptive，一旦被執行，就要等待該Process執行完成才能選下一個Process執行，所以並不能比SRTF還要快。若每個Process的Arrival Time差距很大(如:數據input3)，則HRRN排程就等於是FCFS的做法了。

四、未完成的功能:無