作業系統 hw2 文件

資訊四甲
10827108
鄞志良

一、開發環境

- 作業系統: windows
- 程式語言: C++
- 開發軟體: dev C++

二、實作方法與流程

主程式流程:

1. 使用者輸入檔名
2. 根據方法及時間片段，進入排序執行
3. 執行排序
4. 寫檔
5. 回到第一步輸入使用者資料

FCFS:

1. 根據抵達時間和 ID 排好最初的先後順序(nonsort)
2. 依照排成的時間，當時間一到就把符合的時間丟進準備程序(ready_queue)
3. 如果 ready_queue 沒東西時，此時就是沒有處理程序執行，把(-)記錄起來，接著再做第 2 步驟一次
4. 處理 ready_queue 的第零個處理程序，根據他需要做的時間，依序記錄起來，同時也要再做第 2 步驟
5. 重複 234 步，直到處理程序全部完成
6. 計算 waiting time 和 turnaround time

RR:

1. 根據抵達時間和 ID 排好最初的先後順序(nonsort)
2. 依照排成的時間，當時間一到就把符合的時間丟進準備程序(ready_queue)
3. 如果 ready_queue 沒東西時，此時就是沒有處理程序執行，把(-)記錄起來，接著再做第 2 步驟一次
4. 處理 ready_queue 的第零個處理程序，根據他需要做的時間，依序記錄起來，同時也要再做第 2 步驟，如果執行時間超過時間片段，將其記錄並丟入 ready_queue 待執行
5. 重複 234 步，直到處理程序全部完成
6. 計算 waiting time 和 turnaround time

SJF:

1. 根據抵達時間和 ID 排好最初的先後順序(nonsort)
2. 依照排成的時間，當時間一到就把符合的時間丟進準備程序(ready_queue)
3. 如果 ready_queue 沒東西時，此時就是沒有處理程序執行，把(-)記錄起來，接著再做第 2 步驟一次
4. 每次執行時需要依照處理程序需要做的時間更新 ready_queue，再處理 ready_queue 的第零個處理程序，根據他需要做的時間，依序記錄起來，同時也要再做第 2 步驟
5. 重複 234 步，直到處理程序全部完成
6. 計算 waiting time 和 turnaround time

HRRN:

1. 根據抵達時間和 ID 排好最初的先後順序(nonsort)
2. 依照排成的時間，當時間一到就把符合的時間丟進準備程序(ready_queue)
3. 如果 ready_queue 沒東西時，此時就是沒有處理程序執行，把(-)記錄起來，接著再做第 2 步驟一次
4. 每次執行時需要依照處理程序的反應時間比例更新 ready_queue，再處理 ready_queue 的第零個處理程序，根據他需要做的時間，依序記錄起來，同時也要再做第 2 步驟
5. 重複 234 步，直到處理程序全部完成
6. 計算 waiting time 和 turnaround time

SRTF:

1. 根據抵達時間和 ID 排好最初的先後順序(nonsort)
2. 依照排成的時間，當時間一到就把符合的時間丟進準備程序(ready_queue)
3. 如果 ready_queue 沒東西時，此時就是沒有處理程序執行，把(-)記錄起來，接著再做第 2 步驟一次
4. 每次執行時需要依照處理程序的還需要做的時間更新 ready_queue，再處理 ready_queue 的第零個處理程序，根據他需要做的時間，依序記錄起來，同時也要再做第 2 步驟
5. 如果中途有需要做的時間少的處理程序出現，必須讓他先做，將此時再執行的處理程序紀錄，並丟入 ready_queue
6. 重複 2345 步，直到處理程序全部完成
7. 計算 waiting time 和 turnaround time

PPRR:

1. 根據抵達時間和 ID 排好最初的先後順序(nonsort)
2. 依照排成的時間，當時間一到就把符合的時間丟進準備程序(ready_queue)

3. 如果 ready_queue 沒東西時，此時就是沒有處理程序執行，把(-)記錄起來，接著再做第 2 步驟一次
4. 每次執行時需要依照處理程序的優先程度更新 ready_queue，再處理 ready_queue 的第零個處理程序，根據他需要做的時間，依序記錄起來，同時也要再做第 2 步驟
5. 如果中途有處理程序的優先程度一樣，需要看時間片段，如果執行時間超過時間片段，將其記錄並丟入 ready_queue 待執行
6. 重複 2345 步，直到處理程序全部完成
7. 計算 waiting time 和 turnaround time

三、不同排成法比較

1. input1

Waiting Time

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

Turnaround Time

ID	FCFS	RR	SJF	SRTF	HRRN	PPRR
0	23	22	9	4	23	4
1	15	10	7	2	7	2
2	25	22	5	5	19	17
3	22	29	10	10	18	4
4	16	22	10	3	16	14
5	26	33	25	25	29	27
6	5	20	10	11	5	16

7	16	3	3	1	4	56
8	23	16	2	2	13	11
9	9	17	4	5	10	4
10	16	45	57	57	26	53
13	19	4	5	1	5	1
20	16	20	8	3	16	43
27	22	34	15	25	15	16
29	20	37	21	25	26	10

2. input2

Waiting Time

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

Turnaround Time

ID	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

3. input 3

Waiting Time

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

Turnaround Time

ID	FCFS	RR	SJF	SRTF	HRRN	PPRR
1	20	20	20	20	20	20

2	25	45	25	25	25	55
3	45	55	45	45	45	60
4	30	30	30	30	30	15
5	10	10	10	10	10	20
6	15	15	15	15	15	10

四、結果與討論

1.FCFS：FCFS 是一種非搶占式調度算法，按照任務到達隊列的先後順序進行處理。第一個到達的任務首先被處理，然後是隊列中的下一個任務，依此類推。該算法可能會導致稍後到達的任務等待時間過長，並且可能導致執行時間較長的任務性能不佳。

2.RR：RR 是一種搶占式調度算法，為隊列中的每個任務分配一個固定的時間片。每個任務在分配的時間片內執行，如果任務在時間片內沒有完成，則被搶占，執行隊列中的下一個任務。該算法可以讓資源的公平分配並防止飢餓。

3.SJF：SJF 是一種非搶占式調度算法，它對執行時間最短的任務進行優先排序。執行時間最短的任務最先執行，然後是隊列中下一個最短的任務，依此類推。該算法可以導致任務的最佳平均等待時間。但是，它可能導致長時間任務的飢餓。

4.SRTF：SRTF 是一種搶占式調度算法，它優先考慮剩餘執行時間最短的任務。該算法不斷地監視隊列中的任務，並搶占剩餘執行時間最短的任務。該算法可以為任務帶來最佳的平均等待時間，但是，由於頻繁的搶占，它可能會導致開銷增加。

5.HRRN：HRRN 是一種非搶占式調度算法，它根據任務的響應率對任務進行優先級排序，響應率是通過將任務的等待時間和執行時間之和除以其執行時間來計算的。響應率最高的任務最先執行，然後是隊列中次高的任務，依此類推。該算法可以為任務帶來最佳的平均等待時間，並為短任務帶來更好的性能。但是，它可能導致長時間任務的飢餓。

6.PPRR：PPRR 是一種搶占式調度算法，它為隊列中的每個任務分配一個優先級值。具有最高優先級的任務首先執行，然後是隊列中下一個最高優先級的任務，依此類推。具有相同優先級的任務使用循環算法執行。該算法可以導致資源的公平分配並防止飢餓。但是，由於頻繁的搶占，它可能會導致開銷增加。