作業系統-作業二

開發環境：VSCode　　開發語言：Python　　10827117 陳柏宇

FCFS

在FCFS中，需要做到的是，依照Process抵達的時間，按照順序做排程，先來的先處理，且不能搶奪CPU資源。

實作方法與流程

這一個排程法實作的流程，先將原本從檔案輸入的資料做複製，接著進入回圈，回圈中，先找出Arrival time最小的Process，接著先判斷目前的時間是否小於該Process的Arrival time，如果小於，則代表實際上該Process還未備妥，時間就加一秒，並輸出 - ，如果大於，就接著判斷該Process的CPU burst time是否等於0，如果等於，則代表該Process已經做完結束了，將其原本的資訊存入陣列當中，而如果不等於，則將時間加一秒並輸出Process的ID。最後，再依照固定的格式，把陣列中的資料輸出到檔案中。

RR

在RR中，需要做到的是，依照Process抵達的時間，還有Time slice的長短，來做排程，每一個Process用完時間片段後，就要換下一個在Queue中等待的Process。

實作方法與流程

這一個排程法中，一樣會先將原本從檔案輸入的資料做複製，接著再取出抵達時間最小的Process放入queue，接著進入回圈，如果queue不是空的，取佇列中的第一個元素，如果是空的，就將time加一秒，並輸出 - ，判斷結束後，會找出在剩下的Process中，有哪些Process的抵達時間等於現在的time，如果等於就將其放進queue。接著是做判斷，首先目前的時間是否小於該Process的Arrival time，如果小於，則代表實際上該Process還未備妥，時間就加一秒，並輸出 - ，如果大於，就接著判斷該Process的CPU burst time是否等於0，如果等於，則代表該Process已經做完結束了，將其原本的資訊存入陣列當中，而如果不等於，就判斷目前的時間片段是否等於最大的時間片段，如果等於就重置時間片段，並將這個Process放到queue最後面，如果不等於則將時間和時間片段加一秒並輸出Process的ID。最後，再依照固定的格式，把陣列中的資料輸出到檔案中。

SRTF

SRTF中，除了依照Process的抵達時間，還有該Process的CPU Burst time來做排程，且是可以奪取CPU資源，如果遇到相同的CPU Burst time，就處理抵達時間比較早的，如果抵達時間也都一樣，就直接看小的PID。

實作方法與流程

這一個排程法中，一樣會先將原本從檔案輸入的資料做複製，接著再取出抵達時間最小的Process放入queue，接著進入回圈，進入回圈後，先判斷是否有與當下時間相符的Process，如果有的話，就將其放進queue之中，完成後，要找出接下來要處理的Process，並且會判斷新取出來的和上一次回圈的Process的CPU Burst time還有ID，來決定是否要繼續處理原本的Process還是要處理新的Process，決定好要處理的Process後，就會開始判斷此Process怎麼處理，先判斷目前的時間是否小於該Process的Arrival time，如果小於，則代表實際上該Process還未備妥，時間就加一秒，並輸出 - ，如果大於，就接著判斷該Process的CPU burst time是否等於0，如果等於，則代表該Process已經做完結束了，將其原本的資訊存入陣列當中，而如果不等於，則將時間加一秒並輸出Process的ID。最後，再依照固定的格式，把陣列中的資料輸出到檔案中。

PPRR

在PPRR中，是依照每一個Process的優先程度，使用RR的排程法去做排程，時間片段用罄後，就要換下一個Process，當優先程度相同時，就要輪流使用時間片段，當有Time out 或是被搶奪後，都要重新將queue依照優先程度做排程。

實作方法與流程

這一個排程法中，一樣會先將原本從檔案輸入的資料做複製，接著進入回圈，進入回圈後，先判斷是否有與當下時間相符的Process，如果有的話，就將其放進queue之中，完成後，選出要執行的Process，而如果與上一次回圈的Process不相同，表示被搶奪，將上一個Process換到queue的最後面，接著就可以進入判斷要如何處理當下的Process，首先目前的時間是否小於該Process的Arrival time，如果小於，則代表實際上該Process還未備妥，時間就加一秒，並輸出 - ，如果大於，就接著判斷該Process的CPU burst time是否等於0，如果等於，則代表該Process已經做完結束了，將其原本的資訊存入陣列當中，而如果不等於，就判斷目前的時間片段是否等於最大的時間片段，如果等於就重置時間片段，並將這個Process放到queue最後面，如果不等於則將時間和時間片段加一秒並輸出Process的ID。最後，再依照固定的格式，把陣列中的資料輸出到檔案中。

HRRN

在HRRN中，依照反應時間比率，的高低來做排程，比率越高，擇優先度越高，且不可奪取CPU資源，而如果比率相同，則依抵達時間小的優先，如果兩個條件都相同，就依照PID來做判斷。

實作方法與流程

這一個排程法中，一樣會先將原本從檔案輸入的資料做複製，接著進入回圈，進入回圈後，依照規則取得正確的Process後，便可以開始判斷此Process怎麼處理，先判斷目前的時間是否小於該Process的Arrival time，如果小於，則代表實際上該Process還未備妥，時間就加一秒，並輸出 - ，如果大於，就接著判斷該Process的CPU burst time是否等於0，如果等於，則代表該Process已經做完結束了，將其原本的資訊存入陣列當中，而如果不等於，則將時間加一秒並輸出Process的ID。最後，再依照固定的格式，把陣列中的資料輸出到檔案中。

不同排程法的比較

Waiting time	FCFS	RR	SRTF	PPRR	HRRN
Input1	14.33	18.4	8.06	14.6	11.6
Input2	8.4	6.4	3	9.4	8.2
Input3	7	13	7	15	7
Turnaround	**FCFS**	**RR**	**SRTF**	**PPRR**	**HRRN**
Input1	18.2	22.2	11.9	18.5	15.5
Input2	13.2	11.2	7.8	14.2	13
Input3	24.1	29.1	24.1	30	24.1

結果與討論

FCFS是依照順序，可以避免掉餓死的狀況，但是在時間上並沒有特別的出色。而在RR中，雖然時間上來說比大多數的都還要久，但是每一個Process都有很平均的被分配到時間片段，不會造成餓死的情況。而SRTF雖然時間上來說算是最出色的，但是有可能會造成餓死的狀況。而在PPRR中，我認為RR加上了優先度，是有稍微的改善了RR的時間，且同樣保持了不會餓死的特性。至於HRRN，時間上也算是相當出色，同時，又會有反應時間比率的Aging機制，不會造成餓死。