takahiry

Program2-Document

使用開發環境：

MacBook Pro (13-inch, 2019, Four Thunderbolt 3 ports) - macOS Catalina

Visual Studio Code

Python 3.7.3 64-bit

未完成功能

無

程式設計

功能：

實作了以下CPU scheduling方法：FCFS、RR、PSJF、NPSJF、Priority，均按老師的規則完成。

使用的data structure：

使用class Process來存讀取到的Process內容，裡面有ID、CPUburst、arrivalTime和priority，以及ready來記錄是否進入過ready queue。

甘特圖使用一個list隨時紀錄，若ProcessID超過10，則按該數字 - A的ASCII碼求得對應的大寫字母，再append進list。

使用processList存放所有的Process，是一個python list且沒有容量限制。

以下data structure具體用法寫在流程中。

FCFS：沒有用到data structure。

RR：readyState使用queue。size無限大，先進先出。

PSJF：readyState使用python的list，size無限大，可自由選擇位置插入。

NSJF：同PSJF

PP：readyState使用python的list，size無限大。

流程：

如老師要求，在input檔案中的第一行第一個int為method，若method為1-5，則執行對應的CPU scheduling，輸出對應的方法，如method為6，則在讀取檔案後，保存讀取內容，依次實作method1-5，最後使用專門的輸出function寫檔案。

turnAroundTime計算方式是完成時間 - arrivalTime

watingTime計算方式是turnAroundTime - CPU Burst

FCFS：先對processList依arrivalTime排序，然後只要依照該次序讓Process一個個執行完畢即可，紀錄turnAroundTime和watingTime。

RR：先把在初始時間就到達的Process push進readyState。當ready queue不為空時，pop一個process出來，若在timeSlice內該process做不完，則讓他佔滿這個TimeSlice，寫入甘特圖。若做得完，則只做需要的時長。然後判斷有沒有arrivalTime比currentTime小的process，有的話push進queue，最後如果當前runningProcess還沒完成，push進queue，loop。

PSJF：一樣先把初始時就有的Process push進list（readyState）。判斷哪個Process的CPU Burst最小，由它佔有CPU。只要它的CPU Burst不為零，則currentTime加1，CPU Burst減1，判斷這個時間節點有沒有新的Process近來，如果有，則比較它和runningProcess的CPU Burst，若新來的CPU Burst比較小，則換它執行，runningProcess回到隊列中排隊。

NSJF：同PSJF，唯一不同點是刪去判斷新來的節點要不要搶奪CPU的部分，有新來的節點直接丟進ReadyState。

PP：每次有要排隊的process時，通過lineUp函數決定新來的Process應該排在Ready Queue中的什麼位置。當readyState不為空時，從readyState pop出最前面的當作RunningProcess。一秒一秒的執行，如果在新的時間節點有process近來，則於running Process比較Priority大小，若新來的優先級比較高，則原來的running Process進入Ready Queue排隊，新來的process直接放到list中0的位置（即開頭）。

關於紀錄turnaroundTime和waitingTime：每次有process terminted時，去原始的processList中找到相同processID的process，將這兩個time紀錄在裡面，方便輸出。