RM和EDF排程器

說明：

撰寫RM與EDF排程機制，從task.txt檔案中讀取週期性任務的資料，之後模擬時間的前進，在當有新工作抵達放入linked list的ready queue。當新工作抵達或是工作結束時，從ready queue中挑選優先權最高的工作執行。

輸入檔格式(文字格式)task.txt

每一行即為一個週期性的task，第一個數字為週期，第二個為執行時間，且都為正整數，且以逗號隔開，週期與相對截限時間不一定相等。任務個數不固定。

Ex:

0, 5, 2, 1 (phase time, period, relative deadline, execution time)

2, 3, 2, 1

輸出檔案格式

(1)如果是使用EDF排程方法，請先以Schedulability test判斷是否可排。

。如果是RM，請以

(2)請依據排程演算法(EDF或是RM)計算出每個單位時間由哪一個task在使用，或是畫出甘特圖。任務編號請從1開始。

Ex:

以RM為例，

0 T1

1 T2

2 T1

3 T2

4 T1

5

以時間3為例，代表時間間隔3-4為任務T2的工作所使用

繳交時間：

11/17

繳交內容：

原始程式、2-3頁報告

PS: 請以**linked list**實作存放系統中等待被執行工作的ready queue。需要改變目前優先權最高工作之執行，只有在新工作抵達或是工作執行結束。struct Task{

int TID;

int Phase;

int Period;

int WCET; //worst-case execution time

int RDeadline; //relative deadline

float Utilization;

};

struct Job{

int release\_time;

int remain\_execution\_time;

int absolute\_deadline;

int TID; //屬於哪個Task的工作

};

int Total\_Job\_Number=0;

int Miss\_Deadline\_Job\_Number=0;

Pseudocode

1: 先將task.txt中的每個任務資料讀入，並儲存在適當資料結構中。N為任務總個數

2: 求得所有任務之週期的最小公倍數LCM，與所有任務之最大Phase Time MaxPH

3: 將系統的就緒佇列Q初始化成空佇列

4: 初始化時間Clock為0;

5: While(Clock<(LCM+MaxPH))

{

6: 判斷Q中的每一個工作是否能在它的絕對截限時間之前完成，((d-Clock-rem\_exe)>0)，若不能，則輸出該工作訊息，並紀錄miss工作加1和刪除

7: 針對每一個任務判斷目前時間Clock是否為它的工作之抵達時間((Clock-Phase)%pi)==0。若是，則在Q中加入工作，並紀錄進入系統的工作個數加1

8: 當有新工作抵達或是結束時，從Q中找到優先權最高的工作，並將其執行時間減1。如果執行時間已為0，則刪除該工作。RM與EDF在尋找優先權最高的工作時，所參考的參數不同，RM檢視週期而EDF檢視絕對截限時間。否則，將正在執行中的工作執行時間減一。

9: Clock++;

}