RM、EDF 與、strict LST 排程器

A1105534 張宏宇

一、引言

本專題實作了三種排程演算法，包括 Rate Monotonic (RM)、Earliest Deadline First (EDF)、和 strict Least Slack Time (strict LST)，以模擬週期性任務的排程。

從 `inputs/test*.txt` 文件中讀取週期性任務的資料，依照排程演算法的邏輯在時間前進時動態決定任務的執行順序，並輸出每個 Clock 的任務執行狀況。

二、輸出格式

位於 `outputs/{SchedulerName}/` 資料夾中，使用 SchedulerName 分類輸出資料夾。

若 txt 與 json 內容為 **“無法排程任務”** 與 **“[]”** 則為無法排程，並且不會有圖片輸出。

- TXT：每一個 clock 的執行任務輸出
- JSON：包含 Dead 和 Arrival 時間點
- PNG：甘特圖 (也一併附於報告中)

三、實作報告

我將這些排程器都從 **“`Scheduler trait`”** Implement，這樣可以方便地進行調用，內部定義了兩個待 Implement 的 func，分別為**可排程測試**與 ReadyQueue 排序。

在模擬過程中，對於每個 test.txt，會遍歷所有的排程器進行執行，在每個排程器中首先進行可排程性測試，確保任務可以被成功排程。如果任務不可排程，則會輸出相應的提示並跳過該任務文件。

如果任務可排程，模擬將進入循環，這裡的 Clock 會從 0 開始，直到達到計算出的結束時間。在每個 Clock，會檢查是否有任務到達，並將其加入 ReadyQueue，以及檢查是否錯過截止時間，並將其從 ReadyQueue 中移除。

之後會根據排程器的策略對 ReadyQueue 進行排序，並執行最高優先權的任務。執行後，更新該任務的剩餘執行時間，若執行完成，則將其從 ReadyQueue 中移除。

這個專題不僅讓我學會了如何實現排程算法，還提高了我對 RTOS 的理解。通過這次作業，我對如何設計和實現一個完整的排程器有了更深入的認識。

圖例:

- 綠線為該 Job release time
- 紅線為該 Job miss deadline

RM

Test1

無法排程

Test2

無法排程

Test3

無法排程

Test4

無法排程

Test5

無法排程

Test6

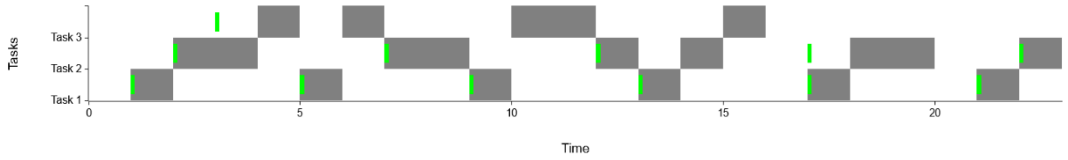
無法排程

圖例:

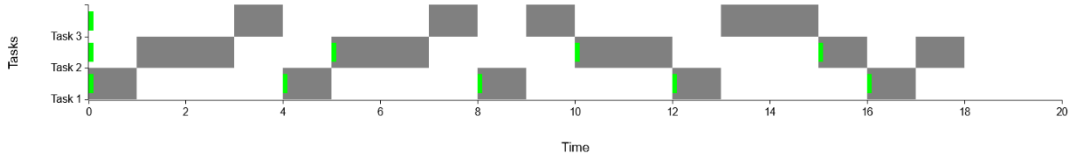
- 綠線為該 Job release time
- 紅線為該 Job miss deadline

EDF

Test1



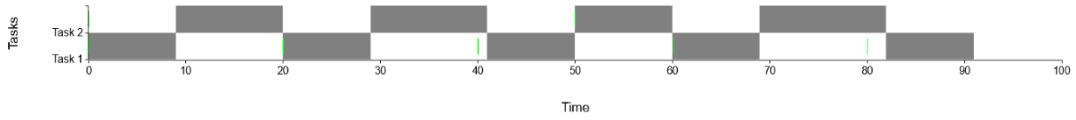
Test2



Test3

無法排程

Test4



Test5

無法排程

Test6

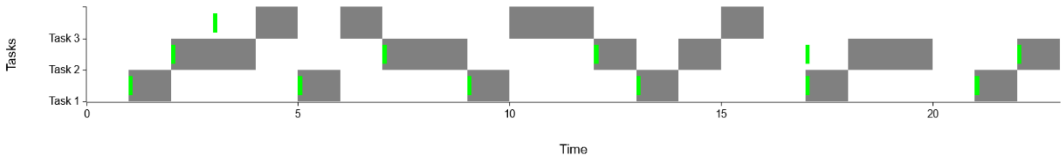
無法排程

圖例:

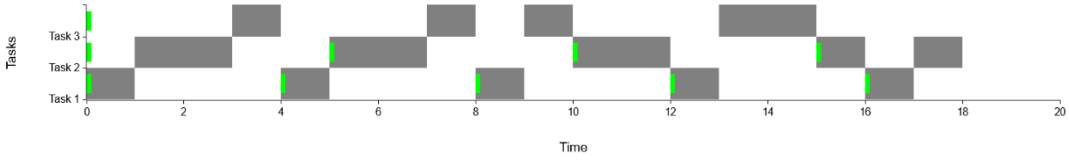
- 綠線為該 Job release time
- 紅線為該 Job miss deadline

Strict LST

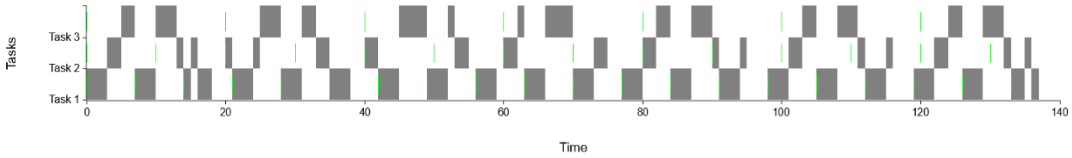
Test1



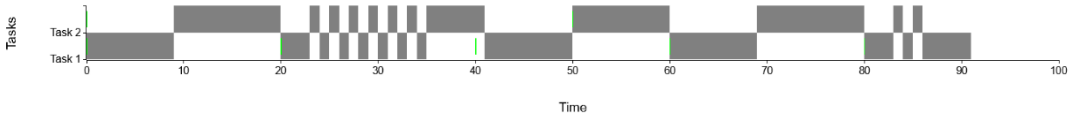
Test2



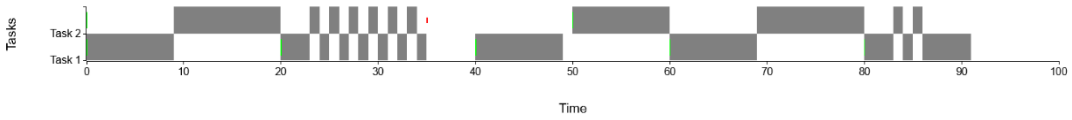
Test3



Test4



Test5



Test6