

# Real-Time System hw2

廖怡誠

這次的作業是使用 CUS、TBS 排程機制，模擬實際的工作排程，以下將分成架構設計、遇到問題與心得三個部分。

## 1.1. 架構設計

在架構的設計上，我主要分成 main scheduler 與 server 兩個部分，前者負責處理週期性的工作，後者則處理非週期性的工作，main scheduler 需要先從週期性工作的 queue 決定最高優先度的週期性工作，再與 server 決定出的非週期性工作進行比較，若週期性工作優先度較高則該工作執行時間減一；反之，若非週期性工作優先度較高則該工作執行時間減一。兩者最大的不同在於，週期性工作是動態優先權，每次都需要重新決定優先度較高的工作，而非週期性工作則是固定優先權，採用 FIFO 規則。

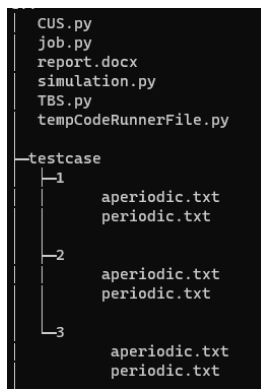
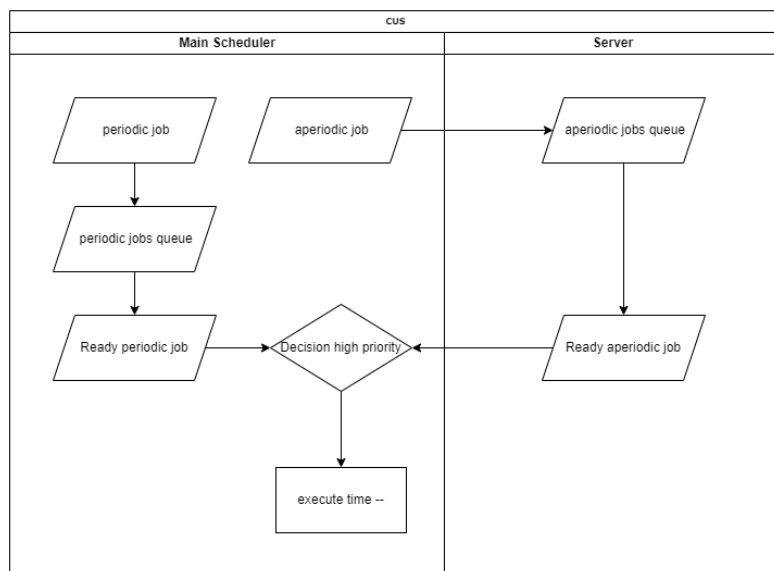


圖 1. 樹狀架構圖



圖二. 程式流程圖

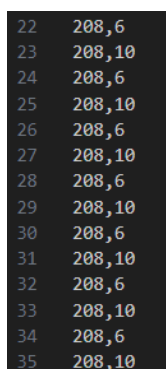
## 1.2. 程式流程圖

圖 2.是這次的程式流程圖，在每一次的時間點都需要先檢查週期性的工作是否有 miss deadline，在依據 Absolute deadline 來排序找出最高優先度的工作，接著 Server 有根據 RULE 找出目前的 Ready job，兩者以 Absolute deadline 決定優先權，並遞減該工作的剩餘執行時間，在每一次的循環中，週期性工作需要從 queue 中重新決定一個工作出來，而 server 則是會將目前的 ready job 消耗完畢才會再從非週期性工作中在找下一個使用。

## 2. 遇到問題

### 2.1. 非週期性工作同時間抵達的執行優先度

在本次專題的複雜 testcase 中有出現非週期性工作同時抵達的情況(圖三)，此時需要 FIFO 規則判定屬於同時進入，教授的说法為按照提供的順序執行，而實際情況是優先執行較短執行時間的工作，我比較支持後者的作法，假設非週期性工作能夠執行的時間是固定的，那必然是較短執行時間的工作能夠完成的越多，能夠提高的效能也越多，但我認為事務上還是要考量長短執行時間的工作 miss deadline 個別所帶來的影響來判定。



22	208,6
23	208,10
24	208,6
25	208,10
26	208,6
27	208,10
28	208,6
29	208,10
30	208,6
31	208,10
32	208,6
33	208,10
34	208,6
35	208,10

圖三. 非週期性工作同時抵達

### 2.2. 非週期性工作 miss deadline 問題

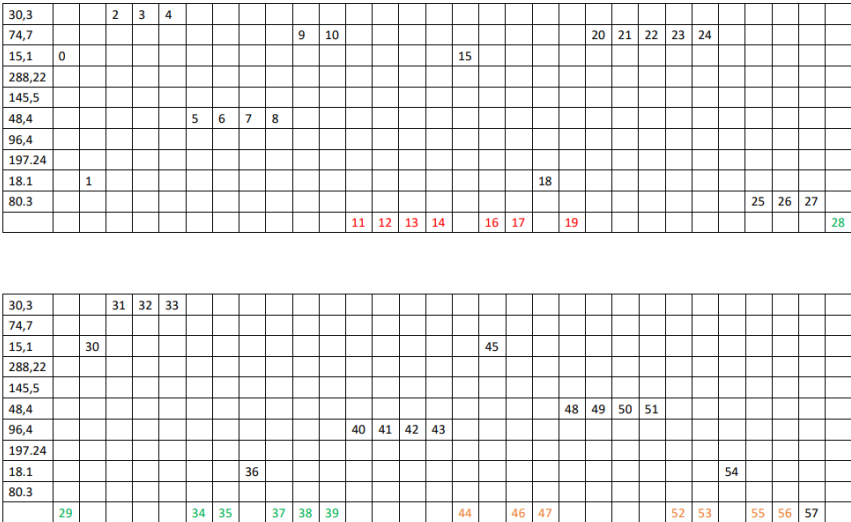
本次專題沒有探討到非週期性工作 miss deadline 的問題，雖說非週期性工作對整體效能影響不大，但也不是說不會影響，以 CUS 來舉例，投影片並沒有提及 clock 到 deadline 時，若 ready job 還沒執行完，該放棄此 job 還是延長執行時間，實際情況應為放棄此 job，但也有人針對這個問題使用動態電壓調整改善系統的能源效率和反應時間或是 Work-demand-based Slack-Stealing scheme，當非週期性工作大於 server 能夠附載的量時，會利用空閒時間提早完成非週期性任務，來降低反應時間。不過，經過檢查之後，測資並沒有出現非週期性工作 miss deadline 的問題，且倘若真的出現 miss deadline 的工作，server 的 deadline 應該先進行補充還是正常等到 Rule 達成在補充也是一個問題點。

## 3. 心得

這次的專題實作起來相較於第一份專題簡單，沒有複雜的邏輯，只需要將程式分成週期性工作與非週期性工作兩部分的程式，週期性工作的概念就是 EDF，而非週期性工作則只要按照規則一步一步實作出來即可，在我處理非週期性工作

的 server 程式碼中，我仿照投影片教的 consumption 與 replenishment 實作，consumption 負責遞減 server 的 budget，replenishment 則負責找出目前的 ready job 以及更新 server 的 deadline，只要根據兩者的功能放入到對應的段落中即可，像是 CUS 的 replenishment 要放在有新的非週期工作進入 queue 或是 clock 已經到 server 的 deadline；而 TBS 則是要放在 server 的 budget 消耗完的部分。最初，我的寫法是將規則拆開，先判斷非週期性工作的 queue 是否為空，若為空，則為 R2a 或 R3b，不為空則為 R2b 或 R3a，但這樣的分法會無法區分出 R2b 和 R3a，因為 R2b 也有 clock 等於 deadline 的情況，於是最後我直接以規則分為 R2 與 R3 兩類。

我認為這份專題最麻煩的地方在於檢查錯誤，因為需要跑 1000 個時間點，如果在比對後，發現不同的部分在時間點 500，就不知道如何檢查，總不可能畫圖畫到時間點 500，幸運的是在 TBS 的 0.8 testcase 中，錯誤的地方在時間點 57，於是我從時間點 0 開始畫到時間點 57(圖四)才發現是 server 的 deadline 出錯。



圖四. 畫圖驗證程式結果