# 嵌入式系統軟體設計 Embedded System Software Design

PA1

指導教授: 陳雅淑 教授

課程學生: M10907305 陳俊億

* **Part 1**

**[Global Scheduling. 10%]**

* Describe how to implement Global scheduling by using pthread. 5%

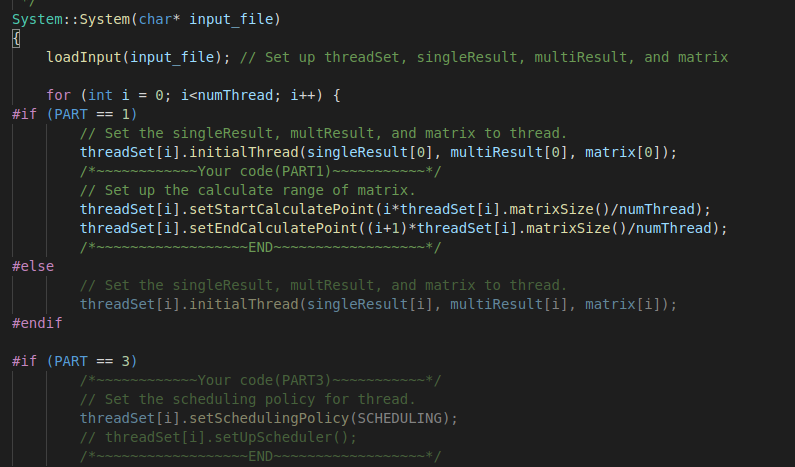


Fig1. Global Scheduling code-1

一開始會先使用threadset先定義每個，thread的計算的範圍在哪。

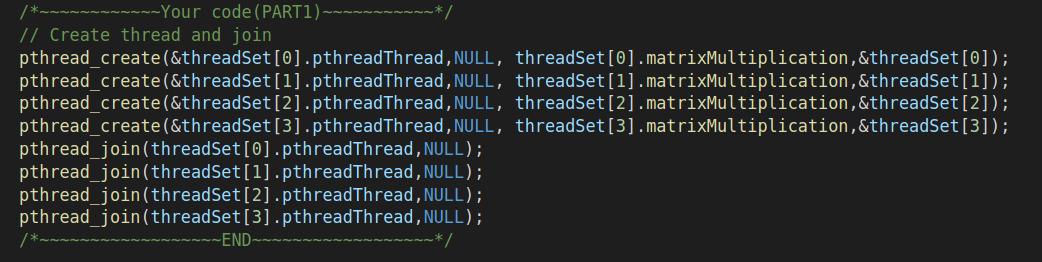


Fig2. Global Scheduling code-1.

然後使用pthread\_create將每個thread與指定的function連接去執行，執行完之後再透過join結束此thread。

* Describe how to observe task migration. 5%

因為global沒有綁住core，所以電腦會有自定義的排程將thread跳到電腦的排程規則裡，一開始先定義每一個thread一開始使用的core，並把數值存入每個thread的core裡，再有不同core執行thread時就會顯示，原本的core跳到另一個core去執行。

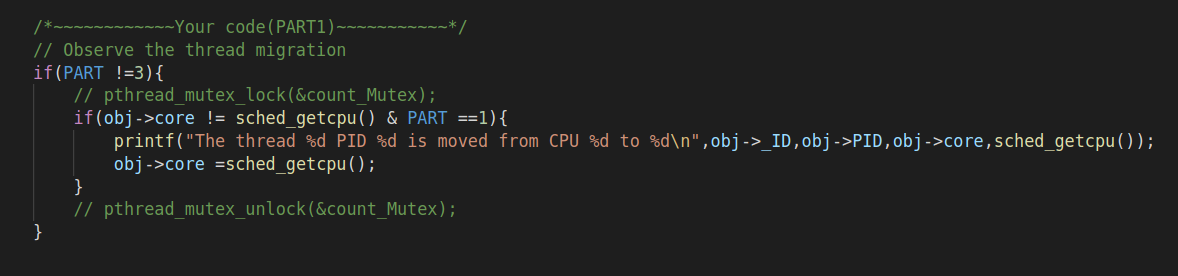


Fig3. Observe task migration code.

**[Partition Scheduling. 5%]**

* Describe how to implement partition scheduling by using pthread.

這邊要先自定義每個thread的core並需固定，然後再pthread\_create和join。



Fig4. Partition scheduling code-1.

上面是只有定義core的參數，thread的執行core並未固定，在這邊透過判別是將定義的core透過setUpCPUAffinityMask function，透過cpu\_set將thread固定在指定的core上，最後printf所要的數據。

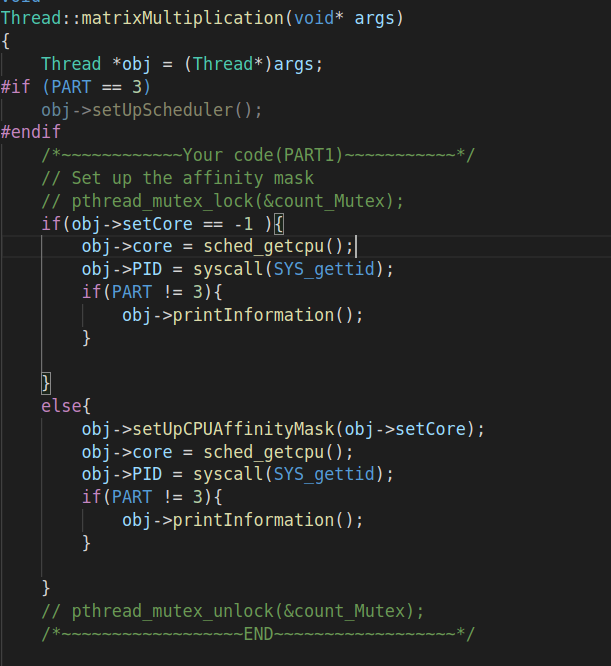


Fig5. Partition scheduling code-2.

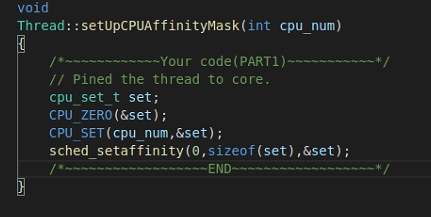


Fig6. Partition scheduling code-3.

**[Result. 10%]**

* Show the scheduling states of tasks. (You have to show the screenshot result of using the input part1\_Input.txt)

1. Screenshot result of using the input part1\_Input.txt

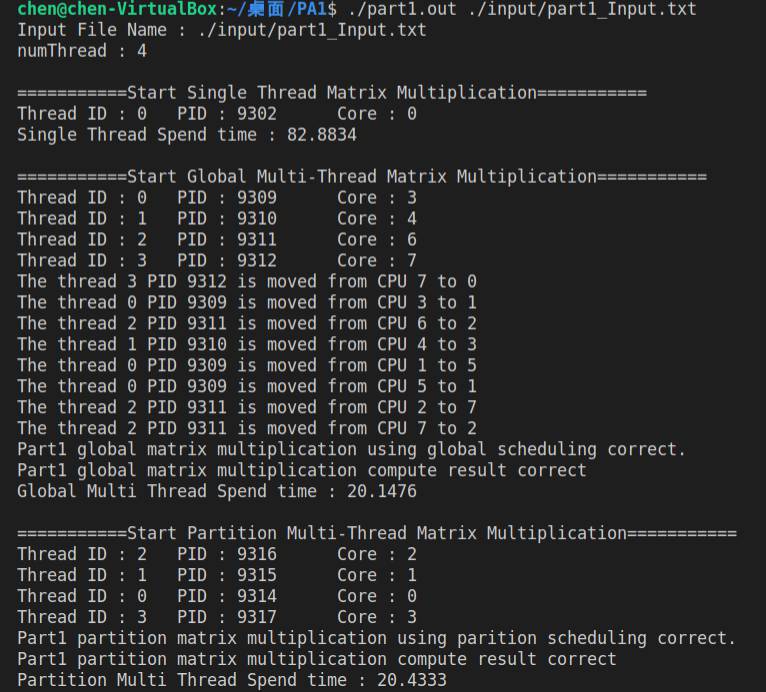


Fig7. Result of Global and Partition.

* **Part 2**

**[Partition method Implementation. 10%]**

* Describe how to implement the three different partition methods (First-Fit, Best-Fit, Worst-Fit) in partition scheduling.

First-Fit，採取將thread優先給低index的core執行，在程式上先直白的，先將第一顆core的使用量塞滿，如果塞不下就給下一顆，直到四顆core都滿了，如果還有沒被排進去的thread，就會被顯示出來以及每顆core被安排的thread是誰。

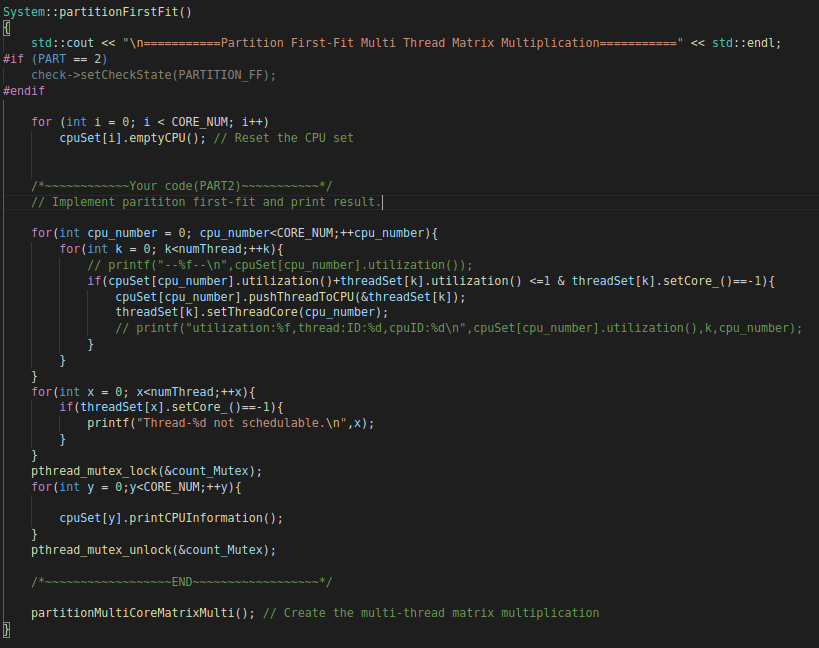


Fig7. First-Fit code

在Best-Fit中，優先把thread給使用量較高的core，會每次去比對core的使用量來判斷。

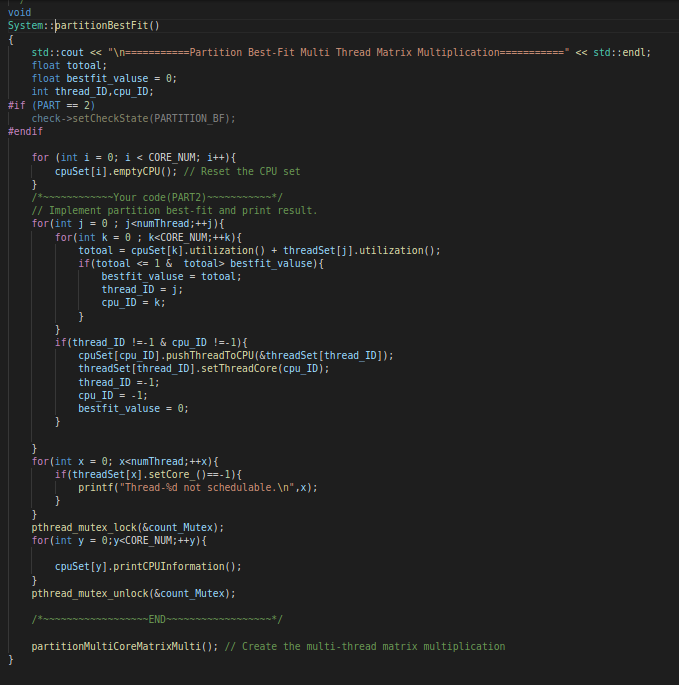


Fig8. Best-Fit code

在Worsr-Fit中，優先把thread給使用量較低的core，會每次去比對core的使用量來判斷。

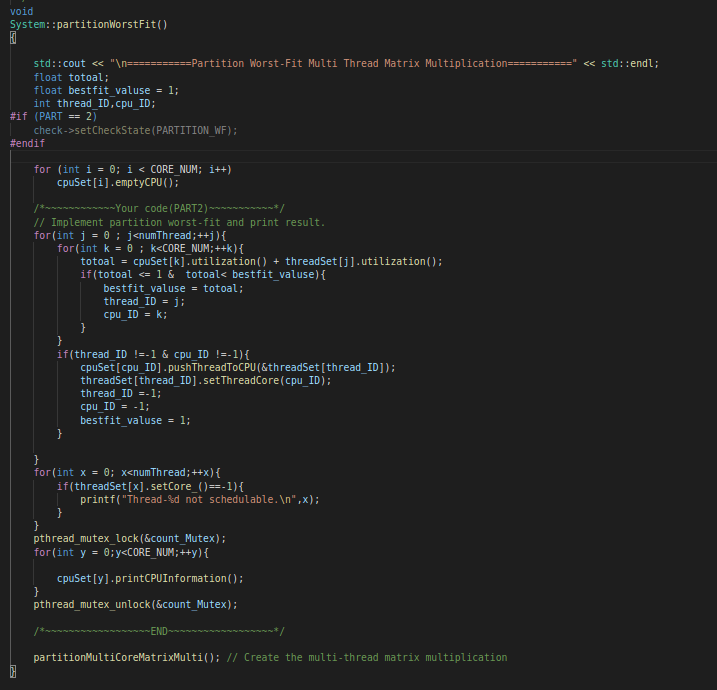


Fig9. Worst-Fit code

**[Result. 30%]**

* Show the scheduling states of tasks. (You have to show the screenshot result of using input part2\_Input\_10.txt and part2\_Input\_20.txt)

1. Result of using input part2\_Input\_10.txt.

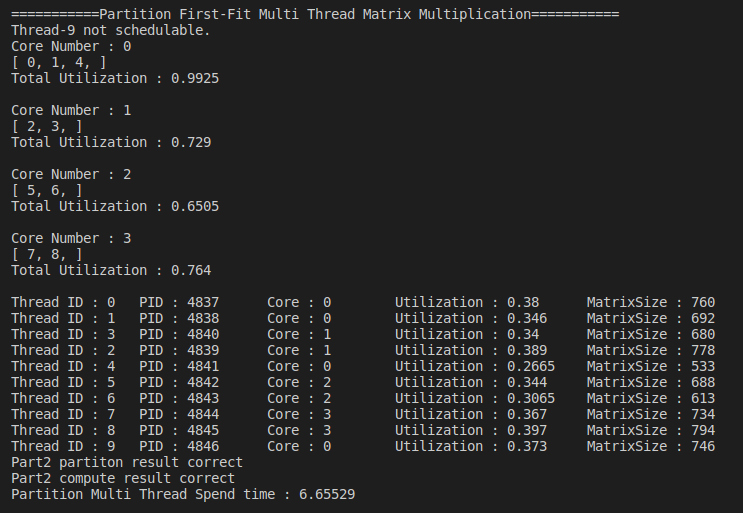


Fig10. Result of using input part2\_Input\_10.txt by FF.

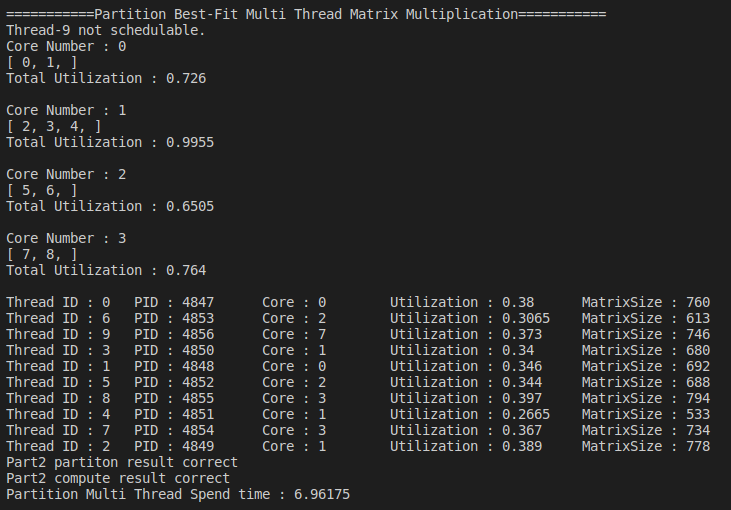


Fig11. Result of using input part2\_Input\_10.txt by BF.

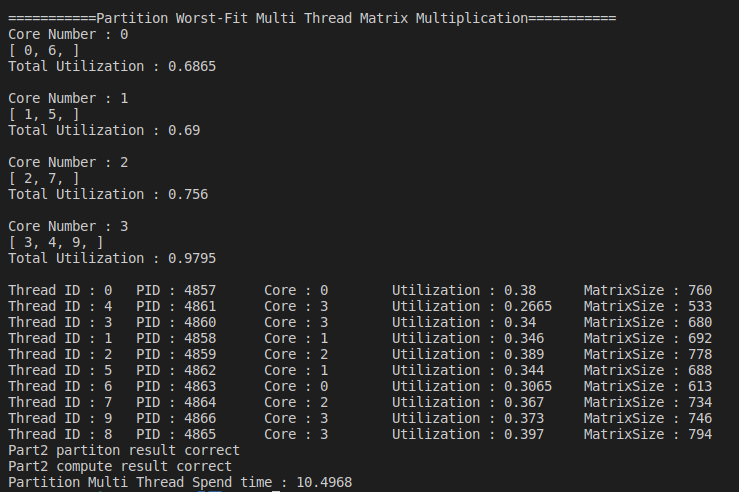


Fig12. Result of using input part2\_Input\_10.txt by WF.

1. Result of using input part2\_Input\_20.txt.

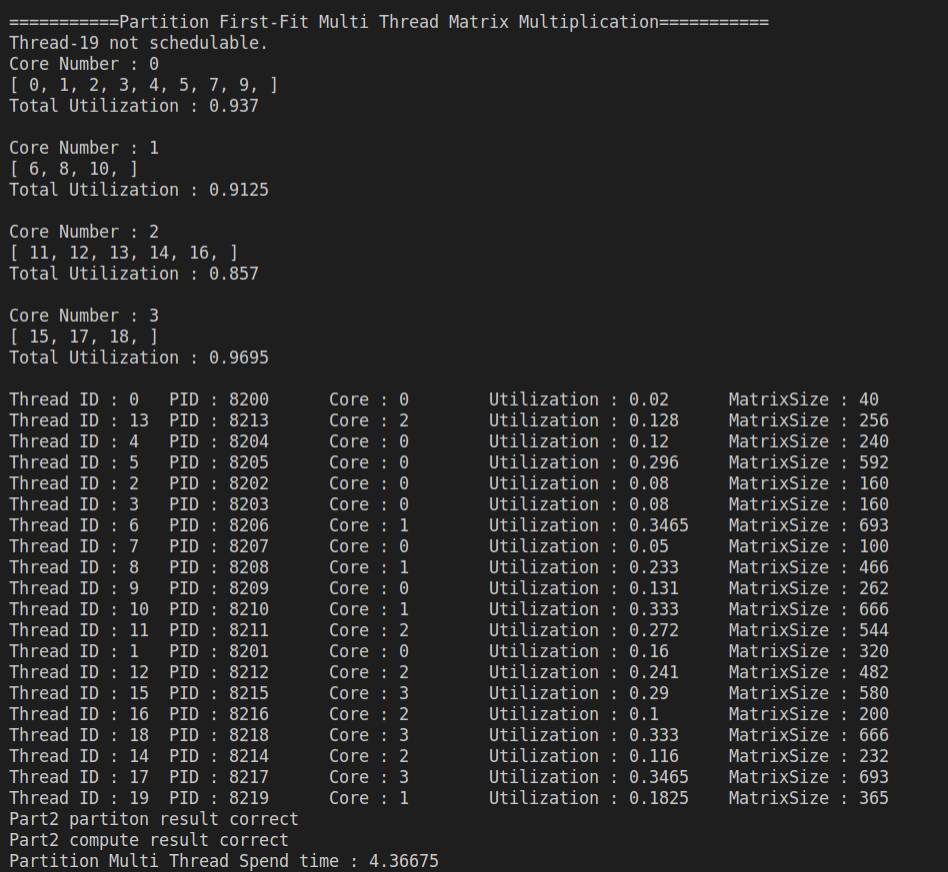


Fig13. Result of using input part2\_Input\_20.txt by FF.

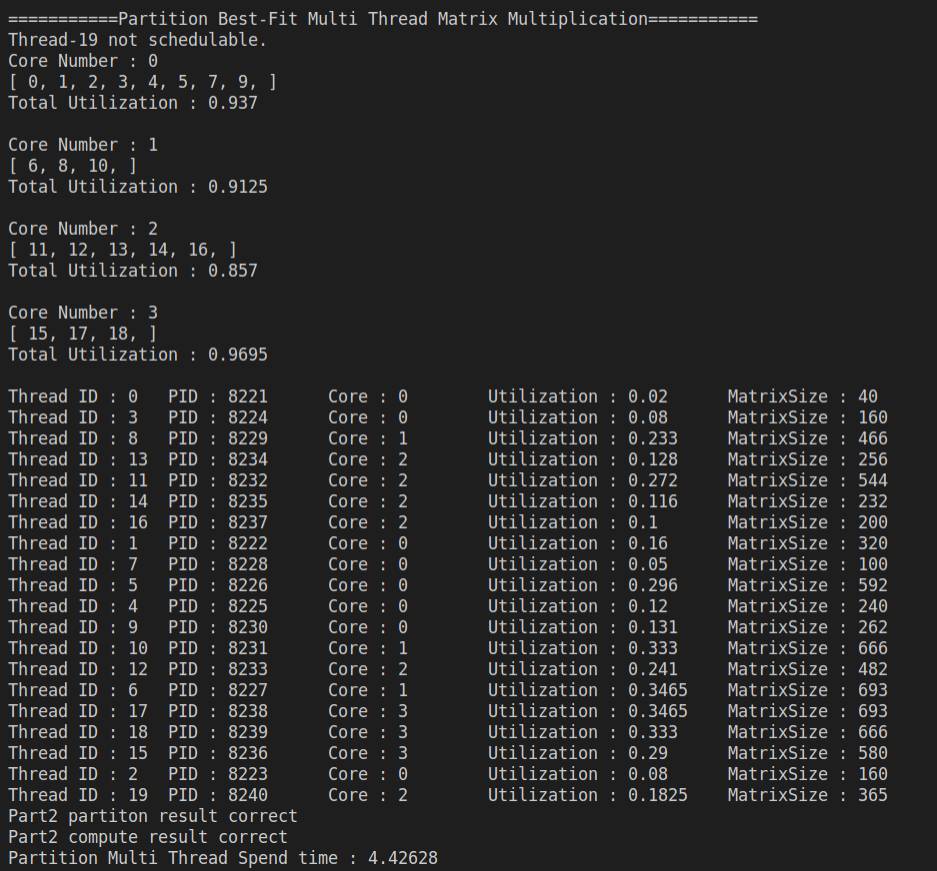


Fig14. Result of using input part2\_Input\_20.txt by BF.

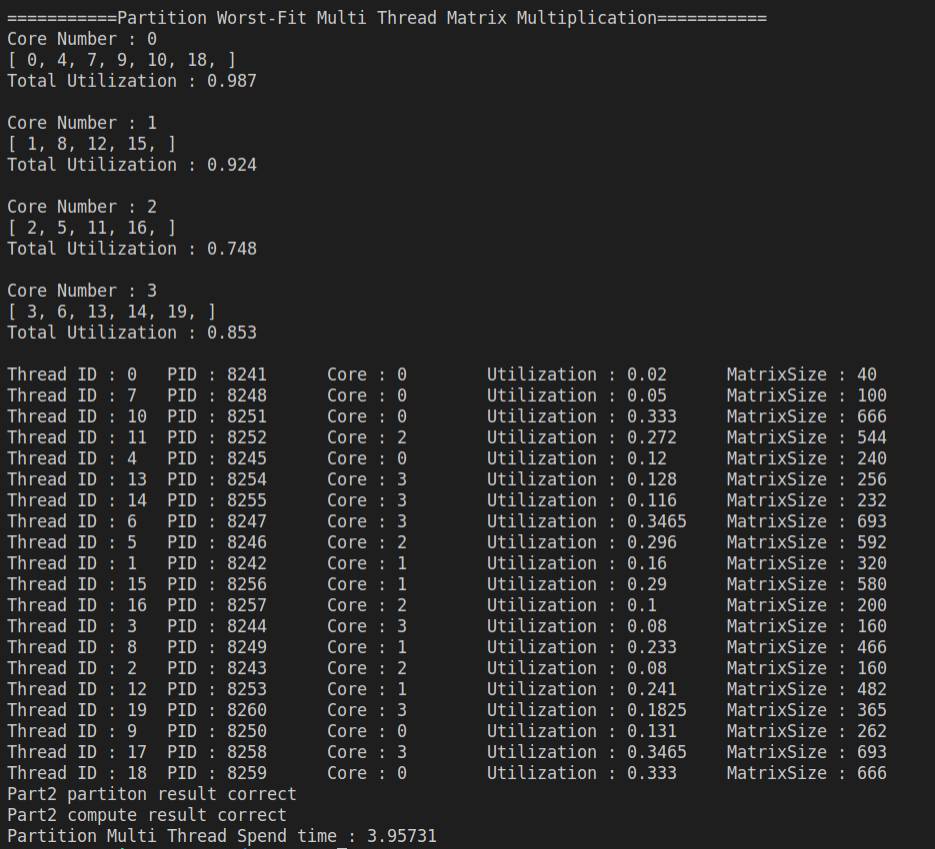


Fig15. Result of using input part2\_Input\_20.txt by WF.

* **Part 3**

**[Scheduler Implementation. 10%]**

* Describe how to implement the scheduler setting in partition scheduling. (FIFO with FF, RR with FF)

一開始除了初始化thread，在part3，一開始會定義出要使用哪個排程(FIFO or RR)，但還沒有成功設定排程規則。

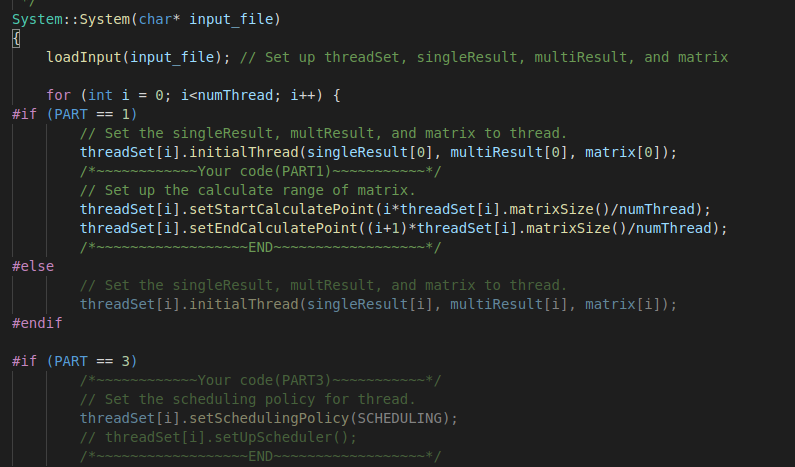


Fig16. implement the scheduler setting code-1.

透過matrixMultiplication呼叫每個thread的setUpScheduler()。

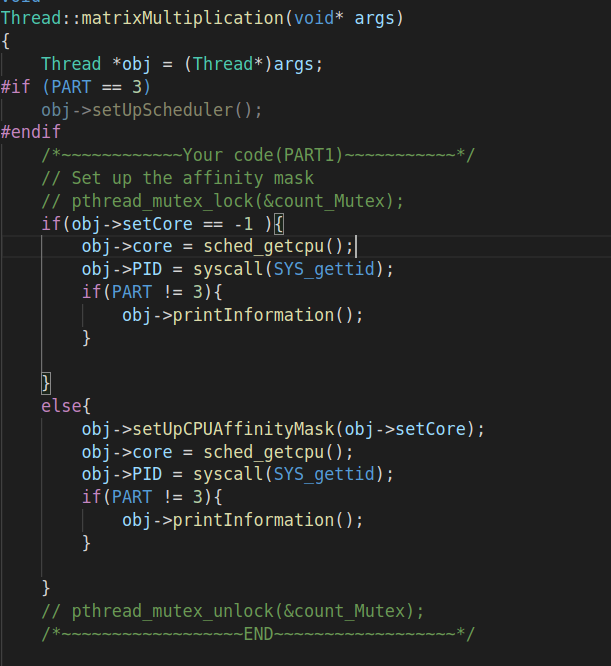


Fig17. implement the scheduler setting code-2.

透過此function可以直接對每一個thread進行排程規則調。

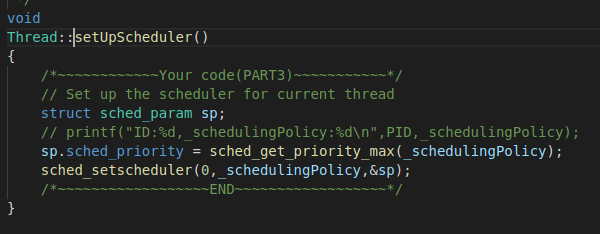


Fig18. implement the scheduler setting code-3.

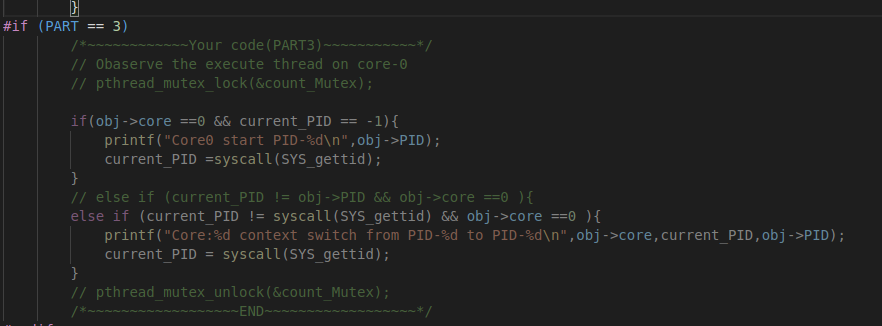


Fig19. implement the scheduler setting code-4.

**[Result. 10%]**

* Show the process execution states of tasks. (You have to show the screenshot result of using input part3\_Input.txt)

1. Result of FIFO

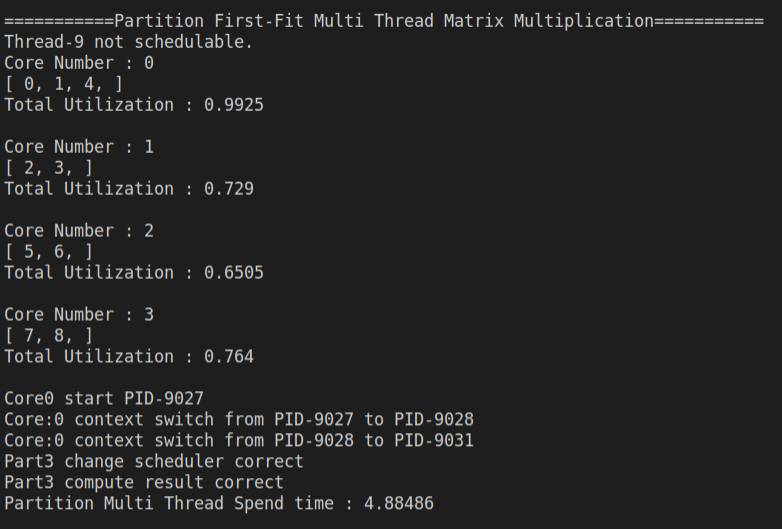


Fig20. Result of FIFO

1. Result of RR

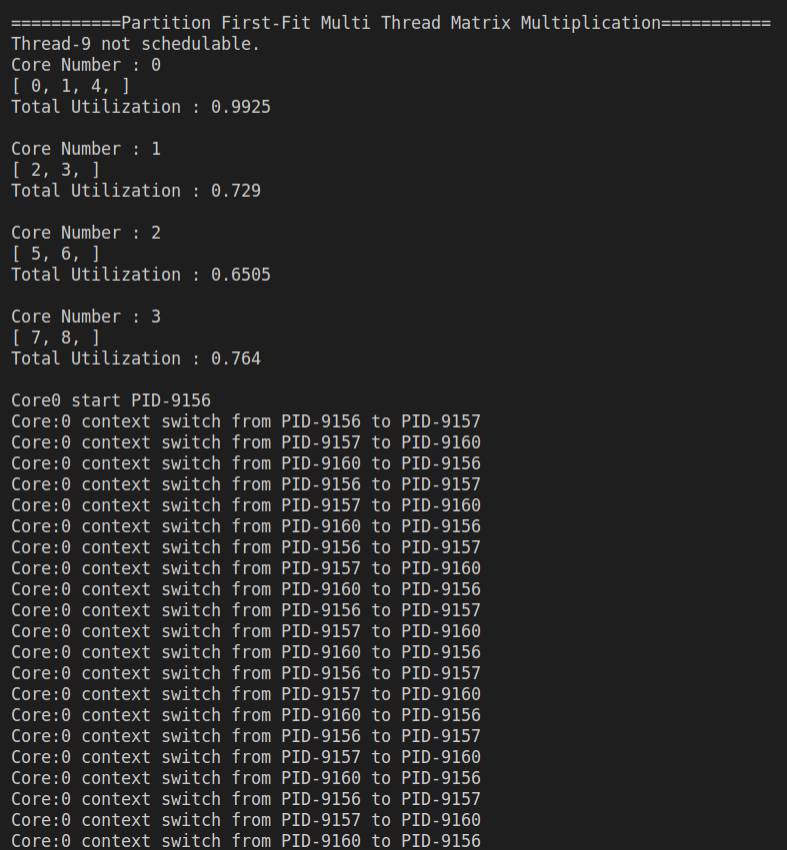


Fig21. Result of RR-1

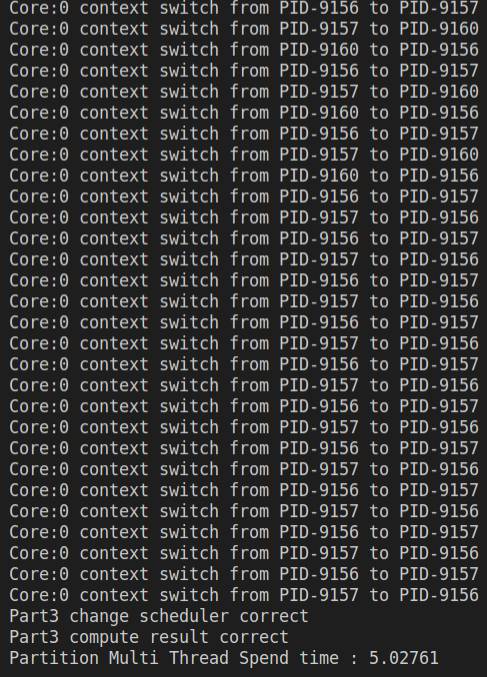


Fig22. Result of RR-2

* **Discussion**
* Analyze and compare the response time of the program, with single thread and multi-thread using in part1 and part2. (Including Single, Global, First-Fit, Best-Fit, Worst-Fit) 10%

Part 1: Result of using input part1\_Input.txt.

|  |  |
| --- | --- |
| Single | Global |
| **93.4943** | **25.1637** |
| **93.5339** | **23.7803** |
| **93.6785** | **21.425** |
| **93.6103** | **24.6045** |
| **93.3217** | **24.9995** |

Part 2: Result of using input part2\_Input\_10.txt.

|  |  |  |  |
| --- | --- | --- | --- |
| Single | First-Fit | Best-Fit | Worst-Fit |
| **18.6205** | **6.91561** | **6.58652** | **9.40535** |
| **18.4112** | **6.0931** | **6.73887** | **9.28765** |
| **18.1321** | **6.14959** | **6.54708** | **9.35252** |
| **18.7423** | **6.108** | **6.48719** | **9.30566** |
| **18.6457** | **6.41064** | **7.1208** | **9.28416** |

Part 2: Result of using input part2\_Input\_20.txt.

|  |  |  |  |
| --- | --- | --- | --- |
| Single | First-Fit | Best-Fit | Worst-Fit |
| **10.5764** | **4.44263** | **4.37661** | **3.9445** |
| **10.559** | **4.39945** | **4.22268** | **3.9663** |
| **10.5462** | **4.47182** | **4.41158** | **3.93752** |
| **10.5662** | **4.41889** | **4.27109** | **3.90958** |
| **10.5418** | **4.24276** | **4.20603** | **3.89103** |

從上述個數據表得知，在part 1時，單一core跑出來的時間很明顯與global差異蠻大的，代表在切範圍分散給各core去平行運算，是有達到加速的功用。而在part 2，很明顯可以看到First-Fit與Best-Fit其實差距不大，甚至在每個core的排程上快幾乎一模一樣，所以無法上述數據中比較出他們之間的誰比較快，可能需要更多的測試數據去觀察，在Worst-Fit，輸入10筆與輸入20筆差距蠻大的，甚至在速度上比其他兩個演算法來的優秀，

但為啥差距之大，有可能是在輸入10筆的排程中，再等一個loading比較大的拖累時間，而在輸入20筆中，剛好分配的很平均，才導致比其他兩個演算法還要好，但還是可能要測試不同種數據才能在更深入的比對。

* Analyze and compare the response time of the program, with two different schedulers. (FIFO with FF, RR with FF) 5%

|  |  |  |
| --- | --- | --- |
| FIFO with FF | RR with FF | |
| 4.8846 | 5.0271 | |
| 4.9 | 5.10912 | |
| 4.82911 | | 5.10921 |

觀察part3的結果後，我們得知FIFO與RR的排程方法不一樣，在FIFO中採取先進先出，當core 0 第一個thread進去執行，他會把core 0 鎖住不讓其他想要用core 0 的thread得進去執行，等到第一個進去thread的做完之後，才會讓下一個thread使用，以此類推。而在RR中採取循環的方式，每一輪每個thread都會進去做一點事，然後馬上切換到下一個thread執行，且每次進去的排程都是固定的，如1、2、4接下來也是1、2、4，除非有其中幾項thread做完，不然會一直輪流下去。Response time目前實行過幾次發現，FIFO速度有優於RR，可能是在context swtich的延遲，才會有一點點差距。