

# OS Assignment 2

110550108 施柏江

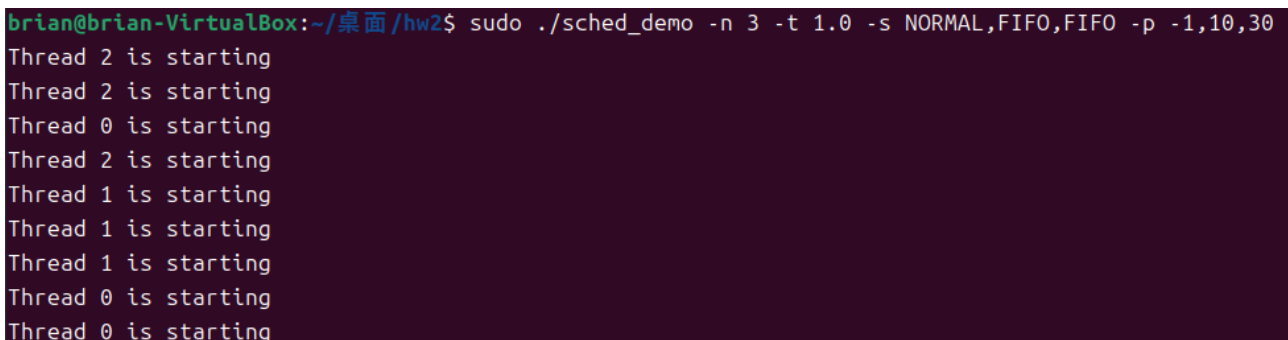
## 1. Describe how you implemented the program in detail.

我用 getopt 來處理 command line 的參數，像是 n 代表 thread 的數量，轉成整數存進 num\_threads；t 是 waiting time，轉成浮點數存到 time\_wait；s 和 p 分別是 policy 和 priority，用逗號分隔後存進 policies 和 priorities 的向量裡。

每個 thread 都有一個 pthread\_t 來存放 id，一個 pthread\_attr\_t 來存放屬性。為了讓所有 thread 能同時開始，我用 pthread\_barrier\_t，並在初始化的時候讓它知道要等待幾個 thread。我用 sched\_setaffinity 把 main thread 綁定到 CPU 0，再用一個 for loop 來設定每個 thread 的屬性，最後用 pthread\_create 建立 thread。

Thread 建立後會先在 barrier 那邊等著，等所有 thread 都到齊了才開始執行。我用 time 取得時間，讓 thread 忙指定的秒數後才結束，然後呼叫 pthread\_exit 來結束 thread。最後用 pthread\_join 確保所有 thread 都執行完後，銷毀每個 thread。

## 2. Describe the results of `sudo ./sched_demo -n 3 -t 1.0 -s NORMAL,FIFO,FIFO -p -1,10,30` and what causes that.



```
brian@brian-VirtualBox: ~/桌面/hw2$ sudo ./sched_demo -n 3 -t 1.0 -s NORMAL,FIFO,FIFO -p -1,10,30
Thread 2 is starting
Thread 2 is starting
Thread 0 is starting
Thread 2 is starting
Thread 1 is starting
Thread 1 is starting
Thread 1 is starting
Thread 0 is starting
Thread 0 is starting
```

FIFO 的 priority 高於 NORMAL，因此 thread 1 和 thread 2 的 priority 高於 thread 0。

Priority 為 30 的 thread 2 高於 priority 為 10 的 thread 1，因此 thread 2 比 thread 1 更優先執行。

由於 sched\_rt\_runtime\_us 預設是小於 sched\_rt\_period\_us 的，保留了一部分時間給 non real-time thread (thread 0)。當 thread 1 和 thread 2 使用了 real-time thread 的 CPU time 上限時，系統會把剩餘的 CPU time 分配給 thread 0，導致 thread 0 有可能在 thread 1 或 thread 2 開始之前就開始了。因此結果就是 thread 2 一定比所有 thread 1 還優先，但 thread 0 無法保證何時執行。

### 3. Describe the results of `sudo ./sched_demo -n 4 -t 0.5 -s NORMAL, FIFO, NORMAL, FIFO -p -1,10,-1,30` and what causes that.

```
brian@brian-VirtualBox:~/桌面/hw2$ sudo ./sched_demo -n 4 -t 0.5 -s NORMAL,FIFO,NORMAL,FIFO -p -1,10,-1,30
Thread 3 is starting
Thread 3 is starting
Thread 0 is starting
Thread 3 is starting
Thread 1 is starting
Thread 1 is starting
Thread 1 is starting
Thread 2 is starting
Thread 2 is starting
Thread 0 is starting
Thread 2 is starting
Thread 0 is starting
```

FIFO 的 priority 高於 NORMAL，因此 thread 1 和 thread 3 的 priority 高於 thread 0 和 thread 2。Priority 為 30 的 thread 3 高於 priority 為 10 的 thread 1，因此 thread 3 比 thread 1 更優先執行。

由於 sched\_rt\_runtime\_us 預設是小於 sched\_rt\_period\_us 的，保留了一部分時間給 non real-time thread (thread 0 和 thread 2)。當 thread 1 和 thread 3 使用了 real-time thread 的 CPU time 上限時，系統會把剩餘的 CPU time 平均分配給 thread 0 和 thread 2，導致 thread 0 和 thread 2 有可能在 thread 1 或 thread 3 開始之前就開始了。因此結果就是 thread 3 一定比所有 thread 1 還優先，但 thread 0 和 thread 2 無法保證何時執行，且會輪流執行。

#### 4. Describe how did you implement n-second-busy-waiting?

我先透過 `time(nullptr)` 取得 thread 進入 waiting state 時的起始時間，並將這個時間存入變數 `start`。在一個 while 迴圈中，不斷使用 `time(nullptr)` 取得當前時間，並與起始時間進行比較，檢查經過的時間是否超過指定的等待時間 `time_wait`。當  $(\text{time}(\text{nullptr}) - \text{start}) > \text{time\_wait}$  時，代表已經經過了指定的等待秒數，這時就會跳出迴圈，結束 waiting。

#### 5. What does the `kernel.sched_rt_runtime_us` effect? If this setting is changed, what will happen?

它會影響 real-time thread 的 CPU 使用量，限制 thread 在每個排程週期中能使用的 CPU time。如果超過了這個時間限制，會將 CPU 讓給 non real-time thread。這樣可以確保 non real-time thread 仍然有機會執行，防止 real-time thread 完全佔用 CPU。

如果將數值調低，會降低 real-time thread 的執行效率，但提高 non real-time thread 的效率。

如果將數值調高，可能會讓 real-time thread 完全佔有 CPU，導致 non real-time thread 有 starvation 的現象。

以下有一個例子：我將值設為 1000000，等同於 `sched_rt_period_us` 的值，因此 real-time thread 會完全佔據 CPU，non real-time thread 會等待 real-time thread 全部執行完才會開始。

```
brian@brian-VirtualBox:~/桌面/hw2$ sudo sysctl -w kernel.sched_rt_runtime_us=1000000
kernel.sched_rt_runtime_us = 1000000
brian@brian-VirtualBox:~/桌面/hw2$ sudo ./sched_demo -n 3 -t 1.0 -s NORMAL,FIFO,FIFO -p -1,10,30
Thread 2 is starting
Thread 2 is starting
Thread 2 is starting
Thread 1 is starting
Thread 1 is starting
Thread 1 is starting
Thread 0 is starting
Thread 0 is starting
Thread 0 is starting
brian@brian-VirtualBox:~/桌面/hw2$ sudo ./sched_demo -n 4 -t 0.5 -s NORMAL,FIFO,NORMAL,FIFO -p -1,10,-1,30
Thread 3 is starting
Thread 3 is starting
Thread 3 is starting
Thread 1 is starting
Thread 1 is starting
Thread 1 is starting
Thread 2 is starting
Thread 0 is starting
Thread 2 is starting
Thread 0 is starting
Thread 2 is starting
Thread 0 is starting
```