

組別：14

學生姓名：XXX、YYY

學生學號：Bxxxxxxx、Byyyyyyyy

學生系級：資工四、資工二

Implementation I：

一、注意事項：在執行 Program 時在 terminal 的 Command 前面都要加上“sudo”。

二、實作步驟

Step 1：將多核的 CPU 限定只能用單核。

程式碼：(檔案 sched_test.c 中第 41 行到第 44 行。)「

```
cpu_set_t set;
CPU_ZERO(&set);
CPU_SET(0, &set);
if (sched_setaffinity(0, sizeof(cpu_set_t), &set) != 0) err_sys("could not set affinity");
」
```

Step 2：取得 sched_param 參數

程式碼：(檔案 sched_test.c 中第 47 行到第 48 行。)「

```
struct sched_param param;
sched_getparam(0, &param);
」
```

Step 3：將 sched_param 參數設定在 sched_get_priority_max 與 sched_get_priority_min 之間。

程式碼：檔案 sched_test.c 中第 49 行。

```
” param.sched_priority = sched_get_priority_max(SCHED_FIFO)-1;”
```

Step 4：將 CPU 調整成使用 First in First out 的 Policy。

程式碼：檔案 sched_test.c 中第 57 行。

```
” if (sched_setscheduler(0, SCHED_FIFO, &param) != 0) err_sys("could not change scheduler. Try using sudo.");”
```

Step 5：創造兩個 threads。

程式碼：檔案 sched_test.c 中第 64 行到第 68 行。

```
「
pthread_t pid[2];
for (int i = 0; i < 2; ++i) {
    pthread_create(&pid[i], NULL, run_thread, (void*)i);
    printf("Thread %d was created\n", i+1);
}
」
```

」

Step 6：每個在 running 狀態中的 thread 跑三次，每當 thread 跑完一次就讓 while 迴圈跑距離現在時間 0.25 秒，再執行下一次。

注意：不能用 sleep()原因：因為使用 sleep()會讓 CPU 將該 Thread 從 Running 踢到 Sleeping 的狀態。而 CPU 會再揀選另一個 Thread 進入 Running 的狀態，不符合 First in First out 精神，先開始執行的 Thread 要先執行完的目的。

程式碼：檔案 sched_test.c 中函式* run_thread()。

Step 7：消滅執行完的 thread。

程式碼：檔案 sched_test.c 中第 69 行到第 71 行。

「

```
for (int i = 0; i < 2; ++i) {  
    pthread_join(pid[i], NULL);  
}
```

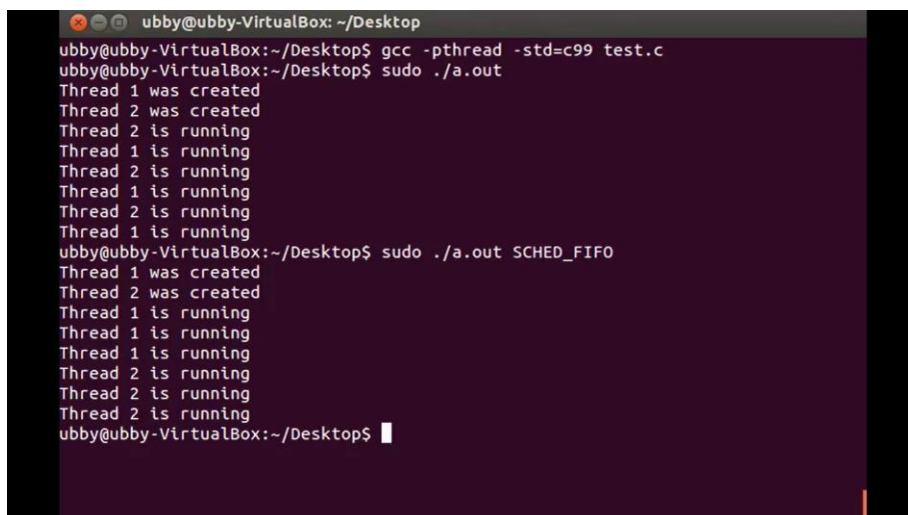
」

三、遇到問題

1. 一開始忘記在 Terminal 輸入 comand 的前面加上"sudo"，使得 if (sched_setscheduler(0, SCHED_FIFO, ¶m) != 0)錯誤。
2. 我們使用 SCHED_FIFO，一開始 terminal 在跑出 Output 前會卡住，最後才把所有 Output 一次跑出來。

四、實作成功相片

1.



```
ubby@ubby-VirtualBox: ~/Desktop  
ubby@ubby-VirtualBox:~/Desktop$ gcc -pthread -std=c99 test.c  
ubby@ubby-VirtualBox:~/Desktop$ sudo ./a.out  
Thread 1 was created  
Thread 2 was created  
Thread 2 is running  
Thread 1 is running  
Thread 2 is running  
Thread 1 is running  
Thread 2 is running  
Thread 1 is running  
Thread 2 is running  
Thread 1 is running  
ubby@ubby-VirtualBox:~/Desktop$ sudo ./a.out SCHED_FIFO  
Thread 1 was created  
Thread 2 was created  
Thread 1 is running  
Thread 1 is running  
Thread 1 is running  
Thread 2 is running  
Thread 2 is running  
Thread 2 is running  
ubby@ubby-VirtualBox:~/Desktop$
```

2.

```
ubby@ubby-VirtualBox: ~/Desktop
Thread 1 is running
Thread 2 is running
Thread 1 is running
Thread 2 is running
Thread 1 is running
ubby@ubby-VirtualBox:~/Desktop$ sudo ./a.out SCHED_FIFO
Thread 1 was created
Thread 2 was created
Thread 1 is running
Thread 1 is running
Thread 1 is running
Thread 2 is running
Thread 2 is running
Thread 2 is running
ubby@ubby-VirtualBox:~/Desktop$ sudo ./a.out SCHED_FIFO
Thread 1 was created
Thread 2 was created
Thread 1 is running
Thread 1 is running
Thread 1 is running
Thread 2 is running
Thread 2 is running
Thread 2 is running
ubby@ubby-VirtualBox:~/Desktop$
```

Implementation II :

一、實作細節

(一) enqueue_task_weighted_rr 函式實作：

函式功能：將新的 task 加入 running queue。

Step 1：將新的 task 加到 running queue 的最末端

程式碼：list_add_tail();

Step 2：將 running queue 中 task 數目加一。

程式碼：rq->weighted_rr.nr_running++;

(二) dequeue_task_weighted_rr() 函式實作：

函式功能：將 running queue 最前面的 task 踢出 running queue。

Step 1：更新 running queue 的資訊。

程式碼：update_curr_weighted_rr(rq);

Step 2：將 running queue 的頭去掉。

程式碼：list_del(&p->weighted_rr_list_item);

Step 3：將 running queue 中 tasks 數目減一。

程式碼：rq->weighted_rr.nr_running--;

(三) yield_task_weighted_rr 函式實作：

函式功能：將正在 running 狀態的 task 放到 Ready queue。

Step 1：將正在 running 狀態的 task 放到 Ready queue 的最後面。

程式碼：list_move_tail(&rq->curr->weighted_rr_list_item, &rq->weighted_rr.queue);

(四) *pick_next_task_weighted_rr 函式實作：

函式功能：將下一個可被執行的 task 從 runnig queue 中 waiting 模式轉換成 running 模式。

Step 1：確定 running queue 不為空。

程式碼：if (rq->weighted_rr.nr_running == 0) return NULL;

Step 2：利用 `list first entry()` 函式回傳排在 `running queue` 第一個的 task。

(五) task tick weighted rr 函式實作：

Step 1 : 更新 `update_curr_weighted_rr(rq);`

Step 2 : task time slice 值減一

Step 3：檢查 task time slice 值是否等於 0

Step 3.1 : task time slice 設為 weighted time slice 原先設定的值。

Step 3.2 : 呼叫 `set_tsk_need_resched()` 函式。

Step 3.3 : 呼叫 `request task weighted rr()` 函式將 task 放回到 running queue 最後面。

二、遇到問題

三、實作成功圖片