OS-Assignment 2

姓名：張哲榕 學號：314551128

1. Describe how you implemented the program in detail.

   A: Main thread 的部分主要分成 Parse arguments, Create thread, Thread function。Parse arguments 的部分主要是使用 unistd.h 來完成提取 arguments，-s 和-p 則需要使用 strtok()分割 flag 後的 string，依序存進 priorities array 和 policies array。Create thread 之前先將每個 thread attribute 設定好，如：policy(從 policies array 取得), real-time thread priority(從 priroities array 取得), CPU affinity(鎖定 CPU 0)，再完成 create thread。Thread function 中使用 pthread_barrier 讓所有的 thread 同時開始，在實作 print 三次的迴圈中，使用 CLOCK_THREAD_CPUTIME_ID 做為紀錄開始與結束的 clock，該 clock 是紀錄佔用 CPU 的時間，因此不會將 context switch 的時間也算進去。

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

   A:

```
/bin # ./sched_demo -n 3 -t 1.0 -s NORMAL,FIFO,FIFO -p -1,10,30
Thread 2 is running
Thread 2 is running
Thread 2 is running
Thread 1 is running
Thread 1 is running
Thread 1 is running
Thread 0 is running
Thread 0 is running
Thread 0 is running
```

   所有的 thread 都被鎖在 CPU 0 執行。由於 thread 1 和 thread 2 的 policy 都被設為 FIFO，且 thread 2 的 priority 也比 thread 1 大，因此會由 thread 2 跑完全部的 work，接著 thread 1 跑完全部的 work，而 NORMAL 的 thread 只有一個，因此最後由 thread 0 跑完 work。

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

   A:

```
/bin # ./sched_demo -n 4 -t 0.5 -s NORMAL,FIFO,NORMAL,FIFO -p -1,10,-1,30
Thread 3 is running
Thread 3 is running
Thread 3 is running
Thread 1 is running
Thread 1 is running
Thread 1 is running
Thread 2 is running
Thread 0 is running
Thread 2 is running
Thread 0 is running
Thread 0 is running
Thread 2 is running
```

   所有的 thread 都被鎖在 CPU 0 執行。Thread 1 和 thread 3 的 policy 都被設為 FIFO，且 thread 3 的 priority 大於 thread 1，因此一開始是由 thread 3 跑完 3 次 loop，接著是 thread 跑完 3 次 loop；剩下 policy 為 NORMAL 的 thread 0 和 thread 2，他們的執行順序是由 CFS 決定的，每次跑的順序可能都不太一樣，但不會像 real-time thread 一樣佔據 CPU 直到做完。

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

   A: 利用”struct timespec”的結構來記錄 start 和 now 的時間，且時間是由 CLOCK_THREAD_CPUTIME_ID 提供，該時鐘紀錄 thread 佔用 CPU 的時間，不會將 context switch 的時間算入，因此可以真正的模仿 thread 佔據 CPU 的時間。3 次的 loop 中 print 完”Thread i is running”後，先記錄 start 的時間，接著一個 while 迴圈中紀錄 now 的時間，若是 now-start > -t 之值才跳出迴圈，確保 thread 每次回圈能夠佔據-t 的時間。

5. What does the kernel.sched_rt_runtime_us effect? If this setting is changed (eg. 500000, 950000, 1000000), what will happen?

   A: kernel.sched_rt_runtime_us 是一個 kernel 參數，主要功用是控制 real-time thread 在每次執行的週期(sched_rt_period_us)中能夠執行的總時間。週期(sched_rt_period_us)預設為 1000000μs，如果將 kernel.sched_rt_runtime_us 設成 500000，則有

500000μs(1000000-500000)可以讓 NORMAL thread 執行，以此類推，在 assignment 中將 kernel.sched_rt_runtime_us 設為-1 代表將 kernel.sched_rt_runtime_us 設成 1000000。