1. 設計

(1) main process 利用 sched_setscheduler 將其設定為 priority=99 的 real-time process，目的是在整個過程中 main process 有最優先權在指定時間點 fork child process

(2) 計算一個 time unit 的平均長度，以 microsec 為單位

(3) 計算並儲存相鄰兩個 child process 的 start time 間隔，開始跑時，main process 會在 fork child process 後根據儲存的時間間隔 usleep()，將 CPU 讓給 child process 跑，直到 usleep() 醒來後 fork 下一個 child process

(4) child process 的 priority 設定，依據不同模式而有不同設計，但都是在 SCHED_FIFO 的 real-time priority 功能之下做調整，以下分別詳述：

(A) FIFO:

所有的 child process 在被 fork 之後，皆由 main process 將其 priority 設定為一定值，如50。但須注意因 child process 被 fork 的當下會繼承 priority=99，而 linux 中 real-time process 若被調低 priority，會被置放至該 priority 的最前端，如此會造成晚 fork 的 child process 會先跑，因此每次 fork 時須先將 child process 的 priority 設定為1，再設定成50，如此晚 fork 的 child process 即可被正確的置放在 priority 的後端。

(B) RR:

如同 FIFO 的做法，只是每個 child process 跑的時候，規定每跑 500 time unit 即執行一次 sched_yield()，也就是讓該 child process 離開 CPU，到 priority queue 的最尾端從頭排隊，也就是 RR 的做法。

(C) SJF:

main process 讀取 input 後，根據 child process 的 execution time 由少到多進行 priority 排序。但因 SJF 在child process 開始跑後便不會被 preempted，因此在 child process 中需自行進行 sched_setscheduler，將自身的 priority 調到 98 (比 main process 少)。

(D) PSJF:

PSJF 需考量 SJF 的方法跑到一半可能進來了 execution time 更短的 process 便會進行 preemption，因此我們利用所有 child process 的 exection time 長度來劃分 priority 的級距，execution time 越短代表 priority 級距越高。每個 child process 在 CPU 跑時，每跑完一個 time unit，便檢查自身剩餘的 execution time 落在哪個 priority 級距內，並視情況更新 (隨著 child process 跑，其 priority 會逐漸上升)；而main process 在 fork 出 child process 時，也會根據被 fork 的 child process 的 execution time 來指定 priority。如此一來若新的 child process 的 execution 較短 (priority 較高)，便會進行preemption，反之若原本的 child process 只剩下少量 execution time，那麼其 priority 便會被提高至相對應的層級，也就能保證在 CPU 上跑的永遠會是剩餘 execution time 最短的 child process。

2. 核心版本

Linux 4.14.25

3. 比較實際結果與理論結果，並解釋造成差異的原因

實際結果跑起來各個 child process 開始與結束的"順序"與理論結果均一致，有達到 scheduling 預期之成效。唯 child process 實際跑的 time unit 與理論值會

有些許誤差(但仍在可接受範圍)，個人認為主要原因一是 context switch 仍會佔用 CPU 時間，且在虛擬機上並未開啟多核心，因此 main process 監管時也會影響到 child process 跑的時間，可視此結果為 task scheduling 必須付出之成本。其次因為是開虛擬機跑，雖然在虛擬機上只有 Project1 在跑，但本機上仍有其他程式在運行，也會影響到虛擬機上的觀測結果。
總體而言，此次 project 的 task scheduling 實作結果還算成功，也有體會到各種 scheduling mode 的箇中滋味，對 task scheduling 也有了更深入的了解。