2020 OS Project 1

B05902040 資工四 宋易軒

設計

- 由一個負責scheduling的process，負責決定該跑哪個process
- 使用兩顆CPU，利用**sched_setaffinity**做設定，一個給負責scheduling的parent，一個給fork出來的child process，避免block住parent的排程工作
- 在排程之前先對child process依據ready time排序，有助FIFO與RR方便找尋下一個要跑的process
- 當一支process第一次獲得執行的priority時，透過**fork**來建立process，並在該process計時，若execution time到了便結束
- 一支process能否真正的在CPU執行的priority，透過**sched_setscheduler**，利用**SCHED_IDLE**以及**SCHED_OTHER**兩個參數，這兩個參數分別對應到要被block以及要被執行的process
- Schedule policy
  - PSJF: 每個時間點都檢查各個process剩餘的執行時間，並執行最短的
  - SJF: 每當CPU閒置或有process結束時，會去檢查各個process剩餘的執行時間，並執行最短的
  - FIFO: 每當CPU閒置或有process結束時，會從一開始照ready time排序的processes中，挑下一個執行
  - RR:每當CPU閒置或有process結束時，會從一開始照ready time排序的processes中，挑下一個執行。在每個time quantum時間到時，會循環式的尋找下一個process來執行
- syscall: 基於printk跟getnstimeofday，建立自己的system function
  - my_get_time(): 負責call getnstimeofday，並把當時的秒數紀錄起來
  - my_print_msg(): call printk，把函式輸入經由printk印出至dmesg
- child process在被生成以及結束時會記錄當前時間，最後印至dmesg

核心版本

linux-4.14.25

比較

從理論上來看，一個process結束換成另一個process執行時，應該會快速的交由下一個process來執行，但是從dmesg印出來的資訊中可看到，兩process之間經過了一小段時間，除context switch的PCB的載入之外，因為我們的scheduler由user space的另一支process負責，當一支process結束時，負責schedule的process要去wait該process，並接著依照policy選擇並產生或提高下支process的priority，也就造成了比理論上更久一些的轉換時間。