OS project1 report

B06902009 柯建宇

1. 設計
   (a) 排程方式

   在每一個 JOB（如：P1）因為抵達 Ready Time 而讀入後，利用 fork() 製造一個新的 process 並且記錄 Process ID，並將其 ID, progression time 記錄到 waiting_list[] 裡面。

   若排程規則為 FIFO/SJF，則是執行 waiting_list[]中依照規則而被排在第一個的 process，直到該 process 執行結束，再決定下一個該執行的 process。

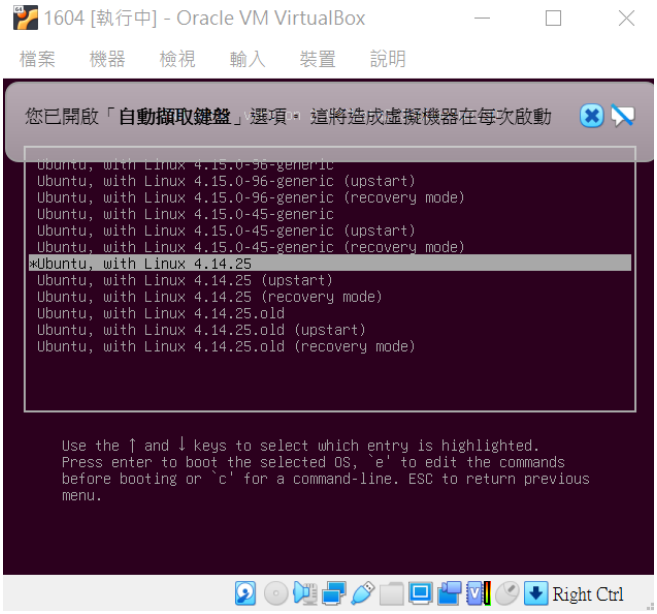
   若排程規則為 PSJF/RR，執行 waiting_list[]中依照規則而被排在第一個的 process 後，需要在特定時間檢查（PSJF 要在每一個新 JOB 進來後，或者當前 process 結束時檢查；RR 需要在每一週期結束時，或者當前 process 結束時檢查）：檢查當前 priority 最高的 process 為何者，再利用 sched_setscheduler()改變當前排程。

   (b) Kernel 變更

   在 linux 的 kernel 裡寫入了兩個程式：my_time、my_dmesg。主要記錄每一 process 產生、結束的時間，並且將此紀錄以 printk() 放置於 dmesg 上。

2. 核心版本

   VirtualBox Ubuntu 16.04 with Linux 4.14.25，2 CPU



3. 實際結果與理論結果，並解釋造成差異的原因
   (a) 執行時間的差異

在一般桌電/筆電的環境下執行，執行時間在各次都會有不只30%以上的變異，經過檢查，判斷應為在一般桌電/筆電，都會有其他應用程式/背景程式等等需要執行，而佔用了些許 CPU 的效能，造成時間的誤差。

後來，運用 VirtualBox 製造較少程式干擾的獨立環境，解決了問題。

(b) CPU 的數量導致排程方式差異

在執行 PSJF/RR 的排程方法時，發現 sched_setscheduler()能夠成功改變 process 的 priority，但是沒有如實將當前的 process 改變成另一個應該執行的 process。

後來，發現原先在 3.-(a) 提及之 VirtualBox 環境，只有配置一顆 CPU，在重新設定新環境，並且設定為 2 顆 CPU 之後，解決了該問題，可以正常執行 PSJF/RR 排程。