Project1 報告

蔡昭信B07902017

參考資料:我們小組的幫忙(OS group 27)和
https://github.com/andy920262/OS2016.git

特別感謝陳致元在我電腦送修的時候幫我跑程式碼

一、 設計：

1. 環境：在VM上執行，為單核系統
2. 主函式(main.c):

   會處理輸入進來的資料並呼叫schedule()
3. 程序控制(process.h和process.c):
   a. 定義了struct process，上面存有程序的名稱(name)、準備完成時間(t_ready)、剩餘執行時間(t_exec)、取得的pid(pid)。這些資料從main()讀入並交給schedule ()來做使用。
   b. exec ()負責產生相應的程序，同時記錄開始(fork())和結束的時間並寫入dmesg。自己寫的gettime和printf會在這裡被使用，紀錄完成後程序會自行結束。
   c. wake(), block ()會使用sched_setscheduler()將指定程序的執行優先度調為最高或是最低，優先度最高表示是在模擬執行中，而優先度最低時則是在模擬sleep的狀態。
4. 排程(scheduler.h和scheduler.c)：

   主要由schedule()及next_process()完成。
   a. 初始化：將每個process的pid設為-1，表示尚未啟動。同時將主程序的執行優先度調為最高，使排程功能正常執行。
   b. 主要變數：使用ntime(自開始以來經過的單位時間)、running(執行中的程序，-1時表示cpu是空閒的)、finished(執行完畢的程序數量)、t_last(上次執行context switch的時刻)來記錄情況。
   c. 檢查執行中程序的狀態：如果執行中的程序已執行完畢，則將running設成-1，且將finished+1。

d. 啟動準備完成的程序：若有程序恰好已準備完畢(ntime==t_ready)，則使用exec()啟動該程序，並讓他休眠block()，避免他提前被執行。

e. 選擇將執行的程序：使用next_process ()根據指定的排程政策選擇將在此單位時間執行的程序。

f. Context Switch：如果執行中的程序與將執行的程序不同，則會使前者休眠block()，且將後者喚醒wake()。

g. 執行程序：使本程序執行1單位時間，並將執行中程序的剩餘執行池間-1(t_exec-1)。

h. 中止：當所有程序皆執行完畢(finished==amount)，程式就完成。

二、 核心版本：4.14.25，和hw1所使用的一樣。

三、 比較與原因解釋：

1. 「單位時間」的實際大小不一：觀察TIME_MEASUREMENT.txt，每個時刻(開始到結束，結束到下一個程序開始)之間的差都是500個單位時間，因此理論上時間差應該都要相同。但是結果發現每個時間差皆不相同，與理論的預期差距甚大。可能是因為exec()所產生的程序和schedule()的程序會互相爭奪CPU資源，使得實際執行的程序需要更長的時間才能完成。

2. 因為使用的環境是VM而不是實際的系統，所以在呼叫systemcall時會比正常狀況慢。