B07902115 陳致元

Project 1 Report

Reference: OS group 27,

https://github.com/andy920262/OS2016/tree/master/project1

1. 設計：
    a. 環境：虛擬機、單核系統
    b. 主函式(main)

       主要負責處理輸入資料及啟動scheduler()。
    c. 程序控制(process control)
        i. 每個程序各有一個對應的結構procData, 儲存該程序的名稱(name)、準備完成時間(tReady)、剩餘執行時間(tExec)、取得的程序ID(pid)。這些資料由main()儲存，並由scheduler()使用及管理。
        ii. execProc()負責產生相應的程序，同時記錄開始(fork())和結束(unitsOfTime()結束)的時間並寫入dmesg。紀錄完成後該程序自行結束。
        iii. wakeProc(), blockProc()分別使用sched_setscheduler()將指定程序的執行優先度調為最高/最低，以模擬執行中或睡眠中的效果。
    d. 排程(scheduling)：主要由scheduler()及selectNext()完成。
        i. 初始化：將各程序的pid設為-1，代表所有程序皆尚未被啟動。同時將主程序的執行優先度調為最高，使排程功能能正常執行。
        ii. 主要變數：使用變數tUnits(自開始以來經過的單位時間)、runningProc(執行中的程序，-1表空閒)、nFinished(已執行完畢的程序數)、lastSwitch(上次執行環境切換的時刻(tUnits))紀錄目前執行的情況。

            若使用的排程政策為RR，則會再使用一佇列紀錄等待中的程序。

iii. 檢查執行中程序的狀態：若判斷執行中的程序已執行完畢，則將此單位時間設為空閒中(runningProc=-1)，且將nFinished加一。

iv. 啟動準備完成的程序：若有程序恰好已準備完畢(tUnits==tReady)，則使用execProc()將該程序啟動，並馬上使其休眠(blockProc())。

v. 選擇將執行的程序：使用selectNext()根據指定的排程政策選擇將在此單位時間執行的程序。
若使用的排程政策為RR且選擇非執行中的程序，則該程序將會從佇列中移除。

vi. 環境切換(Context Switch)：藉由改變執行優先度來模擬環境切換的過程。若執行中的程序與將執行的程序不同，則會使前者休眠(blockProc())，且將後者喚醒(wakeProc())。
若使用的排程政策為RR且有執行中的程序進入休眠，則將此程序插入於佇列中。

vii. 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減一(tExec-1)。

viii. 中止：當所有程序皆執行完畢(nFinished==nProc)，本程序結束。

2. 核心版本：4.14.25
3. 比較與原因解釋：
    a. 「單位時間」的實際大小不一：觀察測試資料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，理論上時間差應該都要相同。然而，觀察TIME_MEASUREMENT_dmesg.txt的結果，發現每個時間差在1~3秒不等，與理論的預期差距甚大。我認為這是某些時候單位時間被放大所導致。這可能跟我所使用的單核環境有關，因為執行中的程

序和排程函式的程序會互相爭奪CPU資源，使得欲執行的程序實際上需要更長的時間才能完成。

b. 執行總時間與預估時間不合：
    i. 估計單位時間大小：根據TIME_MEASUREMENT_dmesg.txt的結果，取第一個時間和最後一個時間相減，除以9500得到單位時間大小的估計值u。
    ii. 估計執行時間：取一筆測試資料計算總共需要的單位時間數t，再乘以u即得估計時間(tu)。
    iii. 實際執行時間計算：根據對應的[測資]_dmesg.txt，取第一個時間和最後一個時間相減即得。
    iv. 結果：預估時間和實際時間相差數秒，有高估也有低估。
    v. 可能原因：單位時間的估計不準確(上述a.)、排程程序和欲執行程序搶CPU(使欲執行程序較慢完成)、環境資源不穩定(筆電當時的使用狀況等)