作業系統 2020 Project1

R08922158 黃聲凱

核心版本

Kernel 版本：Linux4.14.25
Platform：16.04

設計概念

使用兩顆 cpu，將行程分配給特定的 cpu，以此避免 scheduler 和 forked processes 之間的搶佔。實現四個排程策略，透過 function 來調整優先級，使行程在一顆 cpu 上依排程策略執行，紀錄行程的 start time 和 end time，再與理論時間做對比。

在 while 迴圈中，1.檢查是否有正在執行的行程已經完成執行。如果所有行程都已經執行完，則迴圈中斷。2.檢查是否有行程準備就緒。3.確定在下一個時間單元中要執行的行程。4.檢查是否需要結束正在執行的行程並且啟動下一個行程。

main.c

進行讀檔：scheduling policy, process name, ready time 和 execution time；
根據排程策略開始進行排程。

priority.c

透過兩個 function 來提高/降低(raise_priority(), reduce_priority())排程的優先權。

process.c

提供 function 來控制 process：
unit_time()用來定義一個 unit 的執行時間。
assign_proc_to_cpu()用來限制 process 在特定 cpu 上執行。
exec_process()用來 fork 行程：

首先 fork 子行程，使用 run unit_time()根據行程的執行時間紀錄 start time；紀錄 end time，使用系統呼叫將 start time 和 end time 打印到 dmesg 中。最後打印行程 name 和 pid。

其次，在父行程中，透過 reduce_priority()模擬子行程就緒狀態，以此來降低子行程的優先級。

schedule.c：

被主程式呼叫的 function，從這裡再去執行選擇的排程策略：
FIFO 排程實現、RR 排程實現、SJF 排程實現、PSJF 排程實現

理論與實際結果差異總結：

1.由於在虛擬機上運行代碼,因此存在不穩定性。

2.我們選擇下一個行程的方式是反覆檢查每個行程並從就緒佇列中選擇一個行程,因此行程數可能會影響 scheduling time。

3.machine 可能還有其他行程,因此我們創建的行程可能會受到 context switch 影響。

4.即使運行相同數量的迴圈,不同的行程可能有不同的執行時間。