資工三 B06902075 林詩敏

核心版本：linux-4.14.25

設計：

1 kernel_files/my_get_time.c

1.1 實作 systemcall 以藉由 getnstimeofday 取得 seconds.nanoseconds。

2 process.c

2.1 process_cpu 指定 process 所使用的 cpu。

2.2 process_create 利用 fork 建立 child process。child process 呼叫 sys_my_get_time 的 systemcall 取得開始與結束時間（開始時間是 process 建立的時間），並在執行完 t_exec 個 UNIT_TIME (empty for-loop of one million iterations) 之後，藉由 echo "…" > /dev/kmsg 使 dmesg 顯示相關資訊。parent process 則將 child process 放在另一個專門跑 child process 的 cpu。

2.3 process_idle 以 sched_setscheduler(pid, SCHED_IDLE, &param) 賦予 process 較低的 priority。

2.4 process_run 以 sched_setscheduler(pid, SCHED_OTHER, &param) 賦予 process 較高的 priority。

3 scheduler.c

3.1 cmp 用於 qsort。

3.2 next_process 決定下一個要執行的 process。在 non-preemptive 的 FIFO 跟 SJF 中，若存在正在執行的 process (running_P != -1)，則讓該 process 繼續執行。其他狀況下，FIFO 選擇 t_ready 最早的 process，RR 每 500 個 UNIT_TIME 選擇下一個 process (照 t_ready 排序)，PSJF 及 SJF 則選擇 t_exec 最短的 process。

3.3 scheduling 負責排程，跑在不同於 child process 的 cpu 避免干擾。每一次 iteration 大約對應一 UNIT_TIME，其中包含以下工作：

3.3.1 檢查逐次遞減的 t_exec 有沒有等於零以判斷 process 是否結束。

3.3.2 在時間等於 t_ready 時，呼叫 process_create 以建立 child process（在此之前該 process 的 pid = -1）。

3.3.3 呼叫 next_process 決定下一次要被執行的 process (next_P)，若 running_P != next_P，則利用 process_idle 及 process_run 進行 context switch。

3.3.4 增加 t_now 並減少 running process 的 t_exec。

結果：

1 TIME_MEASUREMENT

1.1 P0~P9 的執行時間 (t_diff) 如下表所示，可以看見理論上應該相同的 t_diff 在實作上並不完全一樣，可能是 cpu 在執行時的狀態不一致所造成，也可能會受到 child process 中呼叫 systemcall 或輸出訊息至 /dev/kmsg 的快慢影響。

P#	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9
t_diff	0.81	0.75	0.76	0.77	0.75	0.78	0.75	0.75	0.73	0.75

2 FIFO_1

2.1 由 TIME_MEASUREMENT 我們得到 unit_time=0.00151969，進一步得到 FIFO_1 中各個 process 以 unit_time 為單位的 turnover time，如下表所示。

P#	P1	P2	P3	P4	P5
t_turnover	538	1006	1516	2052	2567

2.2 可以看見 turnover time 比理論上的 500, 1000,1500, 2000, 2500 來的久一些，推測可能是 scheduling 的 overhead 較長所造成（相對於 TIME_MEASUREMENT 測試）。

3 PSJF_2

3.1 理論上各個 process 的執行順序與 turnover time 如下兩張表所示。

t	1000	2000	4000	5000	7000	8000	11000
P#	P1	P2	P1	P3	P4	P5	P3

P#	P2	P1	P4	P5	P3
t_turnover	1000	4000	2000	1000	8000

3.2 實際上各個 process 的 turnover time 如下表所示。

P#	P2	P1	P4	P5	P3
t_turnover	972	3931	1939	1000	7412

3.3 可以看見 turnover time 比理論上的短一些，推測可能是 scheduling 的 overhead 較短所造成（相對於 TIME_MEASUREMENT 測試）。