OS2020_Project 1_ Report

b07902012 江宗翰

核心版本：Linux 4.14.25

設計：主要參考 https://github.com/andy920262/OS2016 之設計。將排程和子程式分給兩個不同的 CPU 來避免其交互影響；print pid 的位置則改成在 child process 被執行時就印出而不用等其跑完；每個單位時間內都先檢查是否有完成的程式，再 fork 已經 ready 的程式，最決定下一個要跑的程式。

比較差異：

由 TIME_MEASUREMENT_dmesg.txt 來計算一單位時間是多少：

每 500 單位時間是 0.784798 秒，每一單位時間是 0.001570 秒

比較項目-FIFO_1.txt

processes	finish_time−ready_time(sec)	finish_time−ready_time(unit)
P1	0.822216	523.704632
P2	1.605824	1022.817679
P3	2.377949	1514.617045
P4	3.155010	2009.560652
P5	3.897081	2482.216981

此例中 P1-4 都是高於理論時間的，而 P5 低於理論時間。

processes	理論值(unit)	實際值/理論值
P1	500	1.047409
P2	1000	1.022818
P3	1500	1.009745
P4	2000	1.00478
P5	2500	0.992887

比較項目-PSJF_2.txt

processes	finish_time−ready_time(sec)	finish_time−ready_time(unit)
P1	6.674092	4251.013896
P2	1.645339	1047.986936
P3	15.686905	9991.659328
P4	3.670829	2338.107376
P5	1.953636	1244.354248

此例中 P1-5 都是高於理論時間的。

processes	理論值(unit)	實際值/理論值
P1	4000	1.062753
P2	1000	1.047987
P3	9000	1.110184
P4	2000	1.169054
P5	1000	1.244354

比較項目-SJF_4.txt

processes	finish_time−ready_time(sec)	finish_time−ready_time(unit)
P1	4.989172	3177.816701
P2	5.143529	3276.132930
P3	10.210381	6503.427408
P4	10.121703	6446.944832
P5	3.389163	2158.702862

此例中 P1-5 都是高於理論時間的。

processes	理論值(unit)	實際值/理論值
P1	3000	1.059272
P2	3000	1.092044
P3	6000	1.083905
P4	6000	1.074491
P5	2000	1.079351

小結：

以上 FIFO 或 SJF 誤差都在 10%甚至 5%內，而 PSJF 則是有超過 10%的誤差。另一個觀察為大部分實際值均高於理論值。

推論：

PSJF 的誤差高於 FIFO 或 SJF 是因為其為 preemptive，受到 context switch 的時間影響；除了 context switch，其 syscalls 的呼叫以及紀錄時間的 instructions 也可能是讓實際值普遍大於理論值的因素。

比較項目-RR_3.txt

processes	finish_time−ready_time(sec)	finish_time−ready_time(unit)
P1	30.743878	19582.087827
P2	29.805970	18984.694390
P3	25.872615	16479.372523
P4	45.850373	29204.059255
P5	43.350083	27611.517754

P6	39.510947	25166.207970

因理論值過於難計算，故沒有列出和理論值的差異；但推論一定高於理論值，且比例應高於 PSJF，因為其更多的 context switch。