


2020/4/29

作業系統

Project 1

李建德(R08922180)

臺灣大學 資訊工程研究所 一年級



零、目標 Goal

學習撰寫程式運行時需要之系統輔助 system calls、並能夠編譯 Linux kernel，最終再比較理論值與實驗值差異、並分析其可能原因。

一、設計

使用兩核心，分別執行排程與指令『空等』，並以不同測資，運行不同排程方式(FIFO, PSFJ, RR, SJF)：

```
#define __NR_gettime 333 : 新增的 system call 之編號  
#define __NR_printtime 334 : 新增的 system call 之編號
```

```
void gettimeofday : 使用新增的 system call 取得系統時間  
void printtime : 使用新增的 system call 印出時間在 dmesg  
void time_msg : 以要求之格式，輸出 pid, 開始時間、結束時間等資訊  
void Run* : 對應不同排程方式，執行其排程(*表：FIFO, PSJF, RR, SJF)
```

二、核心版本

使用 HW1 的環境：

Oracle VM VirtualBox: Version 6.1.4 Edition.

Ubuntu (64-bit): 16.04.4 LTS.

Kernel: Linux 4.14.25.

新增內容：

1. arch/x86/entry/syscalls/syscall_64.tbl 新增 system call 名稱與編號 (333 common gettimeofday sys_gettime 與 334 common printtime sys_printtime)。
2. include/linux/syscalls.h 新增 asmlinkage 資料(asmlink long sys_gettime(struct timespec *ts); 與 asmlinkage long sys_printtime(char *str, int pid, struct timespec *start, struct timespec *end);)。
3. 新增 gettimeofday, printtime 資料夾，內放 gettimeofday 與 printtime 的 Makefile, .c, .h 檔，執行時會自動生成 .o 檔。
4. 重新編譯 kernel 並重開機後，即可使用 make, make run, make run1~5, make allfile...等指令，來編譯 main.c, run.c, run.h 等檔案，最終完成目標。

三、實驗結果

以下將以要求之五組測試資料作說明 (TIME_MEASURE.txt, FIFO_1.txt, PSJF_2.txt, RR_3.txt, SJF_4.txt) · 若有不清楚 · 可直接參考 excel 檔案 · 裡頭有更詳細的內容、包含圖片、公式是如何計算的：

原始 dmesg	編號	開始時間	結束時間	差值(結束-開始)	比例	派差	P	Ready	Exec	Turn
TEST_MEASUREMENT										
[33231.865341] [Project] 30857 1587863629.498247773 1587863630.167628020	30857	1587863629.498247773	1587863630.167628020	0.66937995	500	0	P0	0	500	
[33233.237546] [Project] 30858 1587863630.851550300 1587863631.539895393	30858	1587863630.851550300	1587863631.539895393	0.688339949	514.1623596	0.028324719	P1	1000	500	
[33234.692982] [Project] 30859 1587863632.278196530 1587863632.994995909	30859	1587863632.278196530	1587863632.994995909	0.716800213	335.4210365	0.070842073	P2	2000	500	
[33236.115172] [Project] 30860 1587863633.718165905 1587863634.417648817	30860	1587863633.718165905	1587863634.417648817	0.699480057	572.4835739	0.044967148	P3	3000	500	
[33237.568267] [Project] 30861 1587863635.176332976 1587863635.870807150	30861	1587863635.176332976	1587863635.870807150	0.694469929	518.7412091	0.037482418	P4	4000	500	
[33238.986545] [Project] 30862 1587863636.586094931 1587863637.289147004	30862	1587863636.586094931	1587863637.289147004	0.703049898	575.1501024	0.050300205	P5	5000	500	
[33240.347539] [Project] 30863 1587863637.959587378 1587863638.650200383	30863	1587863637.959587378	1587863638.650200383	0.690619946	515.8654259	0.031730852	P6	6000	500	
[33241.692544] [Project] 30864 1587863639.298072642 1587863639.995265630	30864	1587863639.298072642	1587863639.995265630	0.697190046	520.7730279	0.041546056	P7	7000	500	
[33243.033124] [Project] 30865 1587863640.659375734 1587863641.335905807	30865	1587863640.659375734	1587863641.335905807	0.676530123	505.3408929	0.010681786	P8	8000	500	
[33244.441001] [Project] 30866 1587863642.061301270 1587863642.743843456	30866	1587863642.061301270	1587863642.743843456	0.68253994	509.8299854	0.019659971	P9	9000	500	
FIFO_1										
[33151.689774] [Project] 30784 1587863549.275710743 1587863549.988498677	30784	1587863549.275710743	1587863549.988498677	0.712779999	500	0	P1	0	500	
[33152.408181] [Project] 30785 1587863549.275712060 1587863550.706938477	30785	1587863549.275712060	1587863550.706938477	1.431219816	1003.970242	0.003970242	P2	0	500	
[33151.112699] [Project] 30786 1587863549.275713091 1587863551.411478324	30786	1587863549.275713091	1587863551.411478324	2.13575983	1498.190069	0.001206621	P3	0	500	
[33153.803771] [Project] 30787 1587863549.275714115 1587863557.107590635	30787	1587863549.275714115	1587863557.107590635	2.826879978	1987.996144	0.008501978	P4	0	500	
[33154.495039] [Project] 30788 1587863549.275715193 1587863552.793888742	30788	1587863549.275715193	1587863552.793888742	3.51816988	2467.921298	0.012831481	P5	0	500	
PSJF_2										
[44333.169932] [Project] 937 1587882732.894369874 1587882734.318229216	937	1587882732.894369874	1587882734.318229216	1.423859835	1000	0	P2	1000	1000	
[44336.350904] [Project] 936 1587882731.454460528 1587882737.152771530	936	1587882731.454460528	1587882737.152771530	5.697810173	4001.665076	0.000416269	P1	0	3000	
[44341.225704] [Project] 939 1587882739.013537257 1587882742.027212290	939	1587882739.013537257	1587882742.027212290	3.013679981	2116.556636	0.058278318	P4	5000	2000	
[44342.681653] [Project] 940 1587882742.058047981 1587882743.483200616	940	1587882742.058047981	1587882743.483200616	1.424260139	1000.281141	0.000781141	P5	7000	1000	
[44346.681234] [Project] 938 1587882734.457654262 1587882747.482889866	938	1587882734.457654262	1587882747.482889866	13.02523017	9147.831726	0.016425747	P3	2000	4000	

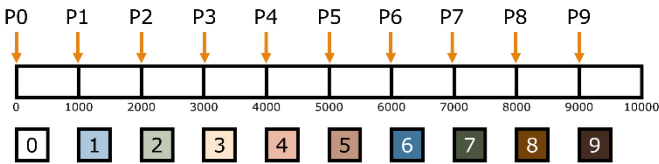
原始 dmesg	編號	開始時間	結束時間	差值(結束-開始)	比例	派差	P	Ready	Exec	Turn
RR_3										
[43469.943736] [Project] 686 1587881849.579904115 1587881870.723347882	686	1587881849.579904115	1587881870.723347882	21.14344001	15400	0	P3	3600	3000	
[43471.327052] [Project] 684 1587881845.968407311 1587881872.106672155	684	1587881845.968407311	1587881872.106672155	26.11826992	19023.45865	0.024438018	P1	1200	5000	
[43472.754309] [Project] 685 1587881847.798372242 1587881873.534033553	685	1587881847.798372242	1587881873.534033553	25.73566008	18744.7816	0.065044409	P2	2400	4000	
[43484.048935] [Project] 689 1587881853.035039168 1587881884.828676577	689	1587881853.035039168	1587881884.828676577	31.79364014	23157.16165	0.033801859	P6	5800	5000	
[43486.918504] [Project] 688 1587881852.049471213 1587881887.698273857	688	1587881852.049471213	1587881887.698273857	35.64880013	25965.09943	0.038603977	P5	5200	6000	
[43488.266931] [Project] 687 1587881851.420016516 1587881889.046713754	687	1587881851.420016516	1587881889.046713754	37.62669992	27405.71916	0.038095423	P4	4800	7000	
SJF_4										
[44613.035959] [Project] 1041 1587883009.489717418 1587883013.843552908	1041	1587883009.489717418	1587883013.843552908	4.353839874	3000	0	P1	0	3000	
[44614.348082] [Project] 1042 1587883010.922388140 1587883015.155700096	1042	1587883010.922388140	1587883015.155700096	4.233319998	2916.956149	0.027681284	P2	1000	1000	
[44620.294689] [Project] 1043 1587883012.318187516 1587883021.102415635	1043	1587883012.318187516	1587883021.102415635	8.784229994	6052.746712	0.008791119	P3	2000	4000	
[44621.628043] [Project] 1045 1587883019.542196819 1587883022.435793594	1045	1587883019.542196819	1587883022.435793594	2.893599987	1993.82168	0.003086906	P5	7000	1000	
[44624.384028] [Project] 1044 1587883016.644912366 1587883025.191828194	1044	1587883016.644912366	1587883025.191828194	8.546090809	5889.221967	0.018463006	P4	5000	2000	

1. 單個 Process 之執行時間

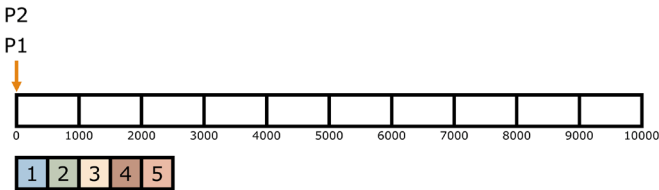
i. 計算理論值：

分配先將各排程方法與測資之 Turnaround 值計算出來：

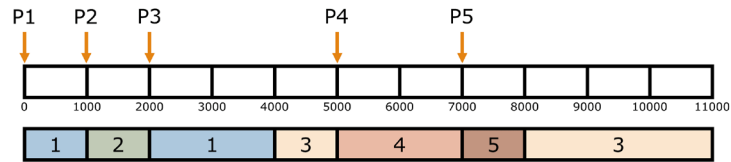
FIFO		Turnaround Time :	
10			
P0 0 500	P5 5000 500	P0 : 500	P5 : 500
P1 1000 500	P6 6000 500	P1 : 500	P6 : 500
P2 2000 500	P7 7000 500	P2 : 500	P7 : 500
P3 3000 500	P8 8000 500	P3 : 500	P8 : 500
P4 4000 500	P9 9000 500	P4 : 500	P9 : 500



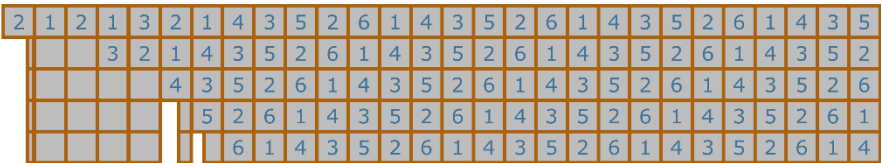
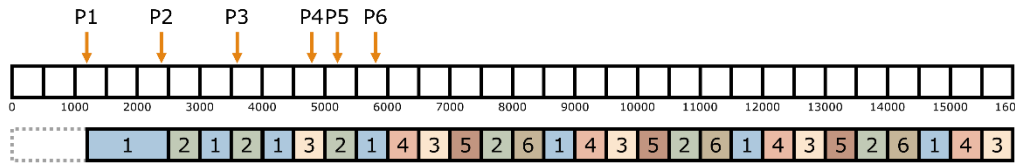
FIFO		Turnaround Time：
5		
P1 0 500	P1：1000	
P2 0 500	P2：2000	
P3 0 500	P3：3000	
P4 0 500	P4：4000	
P5 0 500	P5：5000	



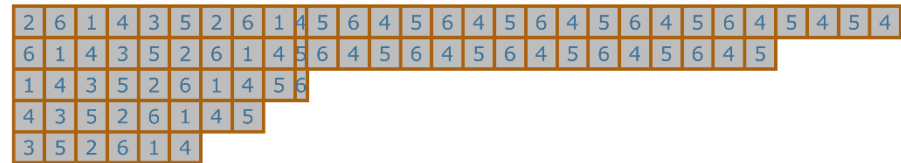
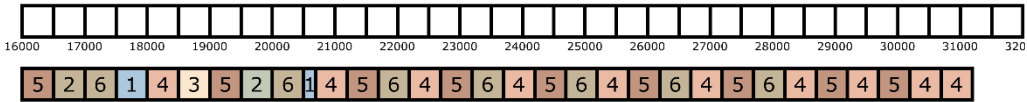
PSJF	
5	Turnaround Time :
P1 0 3000	P1 : 4000
P2 1000 1000	P2 : 1000
P3 2000 4000	P3 : 9000
P4 5000 2000	P4 : 2000
P5 7000 1000	P5 : 1000



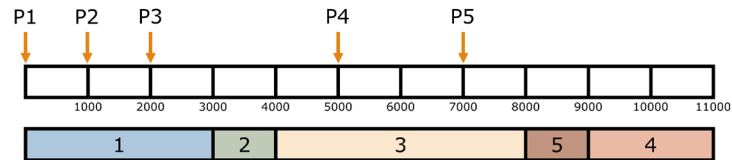
RR	
6	Turnaround Time :
P1 1200 5000	P1 : 20700-1200=19500
P2 2400 4000	P2 : 20000-2400=17600
P3 3600 3000	P3 : 19000-3600=15400
P4 4800 7000	P4 : 31200-4800=26400
P5 5200 6000	P5 : 30200-5200=25000
P6 5800 5000	P6 : 28200-5800=22400



RR	
6	Turnaround Time :
P1 1200 5000	P1 : 20700-1200=19500
P2 2400 4000	P2 : 20000-2400=17600
P3 3600 3000	P3 : 19000-3600=15400
P4 4800 7000	P4 : 31200-4800=26400
P5 5200 6000	P5 : 30200-5200=25000
P6 5800 5000	P6 : 28200-5800=22400



SJF	
5	Turnaround Time :
P1 0 3000	P1 : 3000
P2 1000 1000	P2 : 3000
P3 2000 4000	P3 : 6000
P4 5000 2000	P4 : 6000
P5 7000 1000	P5 : 2000



ii. 分析 dmesg 紀錄：

將輸出之 dmesg 紀錄存至 excel 並拆解，利用 mid(位置, 起啟位, 截取位數)，分別抓出 process 的編號、開始與結束時間，相減後可得『實際執行時間』（差值=結束時間-開始時間）。

iii. 比較理論與實際值：

將每次實驗的第一個 Process 作類似正規化，轉換成 Turnaround Time，再對餘下 Processes 也進行相同運算(excel 圖表內深藍色數字組)，最後再將利用公式： $\text{abs}(\text{實際轉換值}-\text{理論值})/\text{理論值}$ ，即可求得誤差(excel 圖表內深綠色數字組)，可發現實際值與理論值非常相近。

不過必須特別說明：以上狀況在進行實驗時，依然還是有數據不合的可能出現，主要是某幾個 process 可能會執行超乎預期的久，推測應是因為實驗是以虛擬機來執行，故可能會受到 host 同時執行其他程式的影響，但通常只要再重新運行幾次就可以得到吻合結果。

2. 整體 Program 之執行時間

i. 計算理論值：

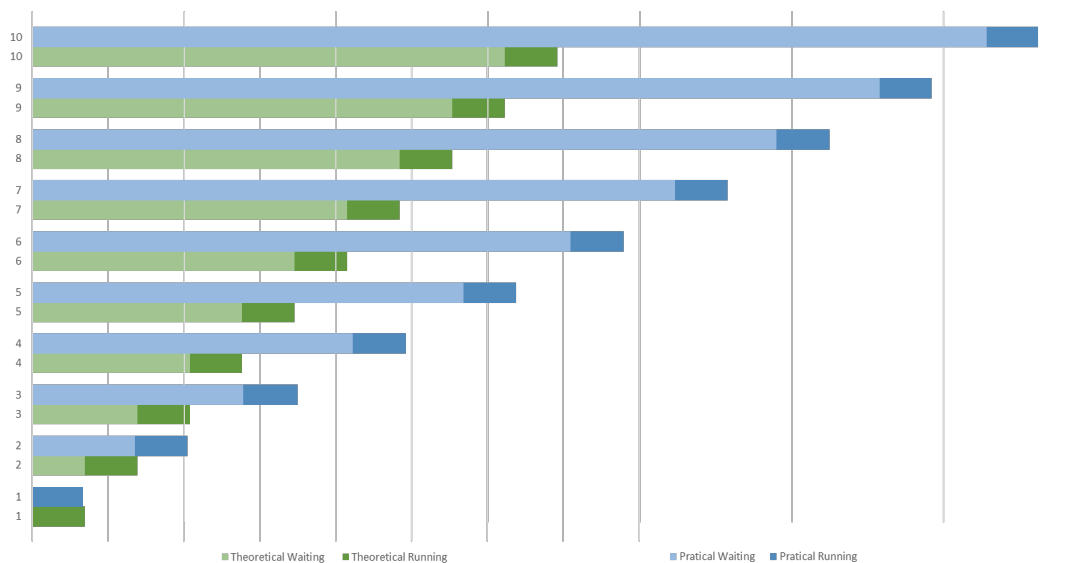
利用 TIME_MEASUREMENT.txt 之資料，將單個 Process 執行時間加總後平均，即可求出理想中的執行時間。

ii. 繪出圖形：

再利用 i. 之理論時間，疊上實際執行時間，可得到下圖：

(綠色表理論值、藍色表實際值，淺色為 Processes 仍在等待時間，深色表 Processes 開始執行)。

(圖片為 excel 完成，並利用 powerpoint 疊圖產生)



iii. 分析：

可能原因不外乎排程時間的延遲、資源獲取延遲，且誤差會隨著時間增加而累積，進而產生愈來愈慢、延遲愈來愈嚴重的結果。

四、實驗過程筆記

1. VM 中若要更改 cores 數量，需先完全關閉電源(非儲存)，才可於 VM 介面上操作、新增 cores，且愈多 cores，編譯 kernel 速度愈快。
2. `sudo dmesg -c/C = sudo dmesg clean/clear`。
3. “<”，“>” 可以分別表示將已打好資料輸入到程式，及將 stdout 內容輸出至 txt(因為一開始並不知道有這樣的指令，故都用 `fscanf/fprintf` 來寫...)。
4. 雖然 HW1 有提到：kernel/Makefile 要加上 `obj-y += ...`，但意外發現沒加好像也還是可以執行，不過其他部分則就必須加(且若有額外資料夾，也要到最外層 Makefile 裡去通知可以使用)。
5. `./main` 可能無法直接執行，需要使用 `sudo ./main`。

五、參考資料

1. Makefile : https://www.youtube.com/watch?v=E1_uuFWibuM
2. Windows 上 Makefile(MinGW) :
<https://www.itread01.com/content/1546865588.html>
3. CPU Affinity :
https://www.gnu.org/software/libc/manual/html_node/CPU-Affinity.html
4. waitpid : https://blog.csdn.net/Roland_Sun/article/details/32084825