OS project 1: CPU Scheduling

B05902062 王子銘　B05902066 蔡翔陞

1. 設計

   此程式會以以下步驟進行：

   a. 使用 sched_setscheduler()設定排程方式為 FIFO，此設定會延續給 fork 出來的 child，並運用 schd_setparam()來設定 priority 以模擬實際排程。
   b. 讀入輸入並排序：先將所有 task 依照 start_time 進行排序，越先開始則越前，若是 type 為 SJF 與 PSJF；則將 start_time 相同的 task 依照 exec_time 的長短排序，越短的越前。
   c. 利用 sched_setaffinity()設定母程序使用 cpu#0，避免與之後 fork 出來的子程序干擾。
   d. 在等待相對應的時間後，運用 fork()生成子程序並依照排程方式設置 priority，對應如下：
      I. FIFO：依照排序順序設置 priority，次序越先 priority 越高。
      II. RR：將所有 task 的 priority 設為相同。
      III. SJF：依照排序順序設置 priority，次序越先 priority 越高。
      IV. PSJF：事先利用 find_pro()來模擬實際排程狀況，並依照結束的先後順序設定 priority，越先結束的 task priority 越高。
   e. 母程序利用 sched_setaffinity()設定子程序使用 cpu#1，避免與母程序干擾並確保子程序的排程由單一 cpu 執行。
   f. 子程序也利用 sched_setaffinity()設定使用 cpu#1，確保只使用固定 cpu。之後便依照 exec_time 執行相對次數的迴圈，結束後便結束程序。唯獨當排程方式為 RR 時，會每隔 500 time units 將自身 priority 減一，達到切換程序進行的要求。
   g. 在 fork 完所有 task 後，母程序便利用 wait()等待所有子程序結束後結束程序，避免產生殭屍程序。

2. 時間

   dmesg 印出的時間由 gettimeofday()產生，對於每個 task，被 fork()之前紀錄一次，呼叫 exit()之前也記錄一次。每個 task 結束之前用 printk()將兩個結果傳到紀錄檔。

3. 範測結果

FIFO

stdout

```
task 1
P1 pid=3239
P2 pid=3240
P3 pid=3241
P4 pid=3242
P5 pid=3243
```

dmesg

```
[ 2768.387734] [os_project1] pid=3239, start = 1526307979.723511934, end = 1526307980.495654106
[ 2769.163994] [os_project1] pid=3240, start = 1526307979.723586082, end = 1526307981.270903110
[ 2769.970957] [os_project1] pid=3241, start = 1526307979.723630905, end = 1526307982.76811075
[ 2770.820073] [os_project1] pid=3242, start = 1526307979.723669052, end = 1526307982.924818992
[ 2771.668164] [os_project1] pid=3243, start = 1526307979.723709106, end = 1526307983.771805047
```

stdout

```
task 2
P1 pid=3246
P2 pid=3247
P3 pid=3248
P4 pid=3249
```

dmesg

```
[ 2900.898823] [os_project1] pid=3246, start = 1526307983.781410932, end = 1526308112.910938024
[ 2909.607214] [os_project1] pid=3247, start = 1526307983.978293895, end = 1526308121.616682052
[ 2911.635095] [os_project1] pid=3248, start = 1526307984.192583084, end = 1526308123.643944025
[ 2913.315316] [os_project1] pid=3249, start = 1526307984.366058111, end = 1526308125.323654890
```

stdout

```
task 3
P1 pid=3663
P2 pid=3678
P3 pid=3682
P4 pid=3685
P5 pid=3695
P6 pid=3696
P7 pid=3700
```

dmesg

```
[ 2926.261903] [os_project1] pid=3663, start = 1526308125.339586973, end = 1526308138.266307115
[ 2934.251229] [os_project1] pid=3678, start = 1526308125.759445905, end = 1526308146.253200054
[ 2939.133230] [os_project1] pid=3682, start = 1526308125.983306884, end = 1526308151.133717060
[ 2940.748104] [os_project1] pid=3685, start = 1526308126.196090936, end = 1526308152.748100042
[ 2942.398079] [os_project1] pid=3695, start = 1526308126.372786045, end = 1526308154.397569894
[ 2944.010485] [os_project1] pid=3696, start = 1526308126.372999906, end = 1526308156.9484052
[ 2950.203885] [os_project1] pid=3700, start = 1526308126.541793107, end = 1526308162.201001882
```

RR

stdout

```
task 1
P5 pid=3735
P4 pid=3734
P3 pid=3733
P2 pid=3732
P1 pid=3731
```

dmesg

```
[ 2951.042052] [os_project1] pid=3735, start = 1526308162.214848995, end = 1526308163.39073944
[ 2951.865444] [os_project1] pid=3734, start = 1526308162.214782953, end = 1526308163.862381935
[ 2952.625847] [os_project1] pid=3733, start = 1526308162.214720010, end = 1526308164.622704982
[ 2953.394393] [os_project1] pid=3732, start = 1526308162.214653015, end = 1526308165.391170978
[ 2954.151309] [os_project1] pid=3731, start = 1526308162.214564085, end = 1526308166.148008108
```

stdout

```
task 2
P1 pid=3769
P2 pid=3770
```

dmesg

```
[ 2967.489760] [os_project1] pid=3769, start = 1526308167.81875085, end = 1526308179.485070943
[ 2969.832255] [os_project1] pid=3770, start = 1526308167.384825944, end = 1526308181.827321052
```

stdout

```
task 3
P3 pid=3776
P2 pid=3775
P1 pid=3774
P6 pid=3779
P5 pid=3778
P4 pid=3777
```

dmesg

```
[ 2997.608766] [os_project1] pid=3776, start = 1526308187.580722093, end = 1526308209.600934028
[ 3005.078934] [os_project1] pid=3775, start = 1526308185.649342060, end = 1526308217.70322990
[ 3011.030909] [os_project1] pid=3774, start = 1526308183.664361953, end = 1526308223.21758079
[ 3012.515252] [os_project1] pid=3779, start = 1526308191.80750942, end = 1526308224.506094932
[ 3016.289096] [os_project1] pid=3778, start = 1526308190.90620994, end = 1526308228.279921054
[ 3017.824412] [os_project1] pid=3777, start = 1526308189.457547903, end = 1526308229.815231084
```

SJF

stdout

```
task 1
P2 pid=3811
P3 pid=3813
P4 pid=3814
P1 pid=3812
```

dmesg

```
[ 3020.876678] [os_project1] pid=3811, start = 1526308229.820694923, end = 1526308232.867482900
[ 3022.396601] [os_project1] pid=3813, start = 1526308229.972565889, end = 1526308234.387402057
[ 3028.891502] [os_project1] pid=3814, start = 1526308230.122601032, end = 1526308240.882272958
[ 3040.557667] [os_project1] pid=3812, start = 1526308229.820745944, end = 1526308252.548388004
```

stdout

dmesg

task 2 P2 pid=3818 P5 pid=3820 P4 pid=3819 P3 pid=3821 P1 pid=3822	[3040.899063] [os_project1] pid=3818, start = 1526308252.718581914, end = 1526308252.889775037 [3041.198658] [os_project1] pid=3820, start = 1526308252.890000104, end = 1526308253.189373016 [3047.447478] [os_project1] pid=3819, start = 1526308252.718791961, end = 1526308259.438165903 [3053.598486] [os_project1] pid=3821, start = 1526308252.890089988, end = 1526308265.589147090 [3064.393957] [os_project1] pid=3822, start = 1526308252.890140056, end = 1526308276.384569883
stdout	dmesg
task 3 P1 pid=3827 P4 pid=3830 P5 pid=3831 P6 pid=3832 P7 pid=3833 P2 pid=3828 P3 pid=3829 P8 pid=3834	[3069.117583] [os_project1] pid=3827, start = 1526308276.543947935, end = 1526308281.108176946 [3069.132552] [os_project1] pid=3830, start = 1526308276.712543010, end = 1526308281.123152017 [3069.147202] [os_project1] pid=3831, start = 1526308276.712641000, end = 1526308281.137801885 [3075.163748] [os_project1] pid=3832, start = 1526308276.864043951, end = 1526308287.154315948 [3081.313380] [os_project1] pid=3833, start = 1526308277.10598897, end = 1526308293.303921937 [3089.015764] [os_project1] pid=3828, start = 1526308276.544055938, end = 1526308301.6268024 [3099.708588] [os_project1] pid=3829, start = 1526308276.544100999, end = 1526308311.699048995 [3113.367190] [os_project1] pid=3834, start = 1526308277.161422967, end = 1526308325.357592105

PSJF	
stdout	dmesg
task 1 P4 pid=3842 P3 pid=3841 P2 pid=3840 P1 pid=3839	[3122.884987] [os_project1] pid=3842, start = 1526308330.207109928, end = 1526308334.875348091 [3129.060891] [os_project1] pid=3841, start = 1526308328.662414073, end = 1526308341.51228046 [3138.391678] [os_project1] pid=3840, start = 1526308327.13663053, end = 1526308350.381800889 [3152.172867] [os_project1] pid=3839, start = 1526308325.363086938, end = 1526308364.163100957
stdout	dmesg
task 2 P2 pid=3846 P1 pid=3845 P4 pid=3848 P5 pid=3849 P3 pid=3847	[3155.287806] [os_project1] pid=3846, start = 1526308365.747116088, end = 1526308367.278031110 [3158.273698] [os_project1] pid=3845, start = 1526308364.168705940, end = 1526308370.263911962 [3163.144641] [os_project1] pid=3848, start = 1526308372.6047964, end = 1526308375.134830951 [3164.675090] [os_project1] pid=3849, start = 1526308375.148303031, end = 1526308376.665275096 [3168.988841] [os_project1] pid=3847, start = 1526308367.311810016, end = 1526308380.979008913
stdout	dmesg
task 3 P2 pid=3855 P3 pid=3856 P4 pid=3857 P1 pid=3854	[3170.570207] [os_project1] pid=3855, start = 1526308381.772943973, end = 1526308382.560365915 [3171.319337] [os_project1] pid=3856, start = 1526308382.549339056, end = 1526308383.309495925 [3172.072019] [os_project1] pid=3857, start = 1526308383.303654909, end = 1526308384.62170982 [3174.328288] [os_project1] pid=3854, start = 1526308380.984054088, end = 1526308386.318429946

4. 比較結果與解釋：

   以下表格為理想的執行時間(start_time, exit_time)與實際執行時間(*start_time, *end_time)的比較，其中實際執行時間已經用 FIFO_4.txt 的結果進行平均，並推算 1 (time units) = 0.00135 (s)

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

FIFO_4.txt

(實際結果儲存於 result)

		task	Start_time	End_time	*start_time	*end_time
FIFO	FIFO_1	P1	0	500	0	571
		P2	0	1000	0	1146
		P3	0	1500	0	2255
		P4	0	2000	0	2371
		P5	0	2500	0	2998
	FIFO_2	P1	0	80000	0	95651
		P2	100	85000	46	102000
		P3	200	86000	300	103600
		P4	300	87000	437	104850
	FIFO_3	P1	0	8000	0	9575
		P2	200	1300	311	15500
		P3	300	1600	470	19100
		P4	400	1700	635	20600
		P5	500	1800	760	21500
		P6	500	1900	761	23400
		P7	600	2300	830	27244

		task	Start_time	End_time	*start_time	*end_time
RR	RR_1	P1	0	500	0	2913
		P2	0	1000	0	2352
		P3	0	1500	0	1783
		P4	0	2000	0	1220
		P5	0	2500	0	871

	RR_2	P1	600	8100	600	9241
		P2	800	9600	-300	10398
	RR_3	P1	1200	20200	1200	30500
		P2	2400	20500	2700	26443
		P3	3600	18200	4200	20511
		P4	4800	19400	4300	34300
		P5	5200	30200	5400	33000
		P6	5800	28200	6000	30221

		task	Start_time	End_time	*start_time	*end_time
SJF	SJF_1	P1	0	14000	0	16835
		P2	0	2000	0	2256
		P3	100	3000	110	3380
		P4	200	7000	223	8200
	SJF_2	P1	200	15400	230	17633
		P2	100	200	100	226
		P3	200	4400	230	9636
		P4	100	8400	100	5077
		P5	200	400	230	451
	SJF_3	P1	100	3100	100	3480
		P2	100	16120	100	18679
		P3	100	23120	100	26140
		P4	200	3110	130	3397
		P5	200	3120	130	3307
		P6	300	7120	240	7960
		P7	400	11120	420	12420
		P8	500	32120	460	36160

		task	Start_time	End_time	*start_time	*end_time
PSJF	PSJF_1	P1	0	25000	0	28740
		P2	1000	16000	1300	18518
		P3	2000	10000	2500	12000
		P4	3000	6000	3600	7000
	PSJF_2	P1	0	3000	0	4514
		P2	1000	4000	1200	2334
		P3	2000	11000	2400	12500
		P4	5000	7000	6300	7174
		P5	7000	8000	8200	9323
	PSJF_3	P1	0	3500	0	3951
		P2	500	1000	600	1183
		P3	1000	1500	1160	1720
		P4	1500	2000	1720	2720

若是單就執行的開始與結束順序，我們的方法在執行的結果上都十分符合預期，但是在 RR 的實作上會發生實作結果與預設情形相反的狀況，可能的原因是 FIFO 在 linux 的規則是新加入的 task 會在 ready list (相同 priority)的最前面，而我們在預測時的想法則是在加在最尾。

另外我們的執行結果並沒有十分精準，可能的原因會是以 FIFO_4.txt 的執行結果本身就有誤差，或是因為最後做了平均使精準度下降，也有可能是當成是在執行時受到背景活動的干擾，由於電腦的程序執行牽涉到許多機制，誤差的來源可能非常廣泛。

5. 工作分配

程式碼撰寫：王子銘（system call）、蔡翔陞（排程主程式）
測資執行整理：王子銘
report 撰寫整理：蔡翔陞