OS Project1

1.設計

本專案分為三個程式:main,scheduler,process，其分工設計如下:

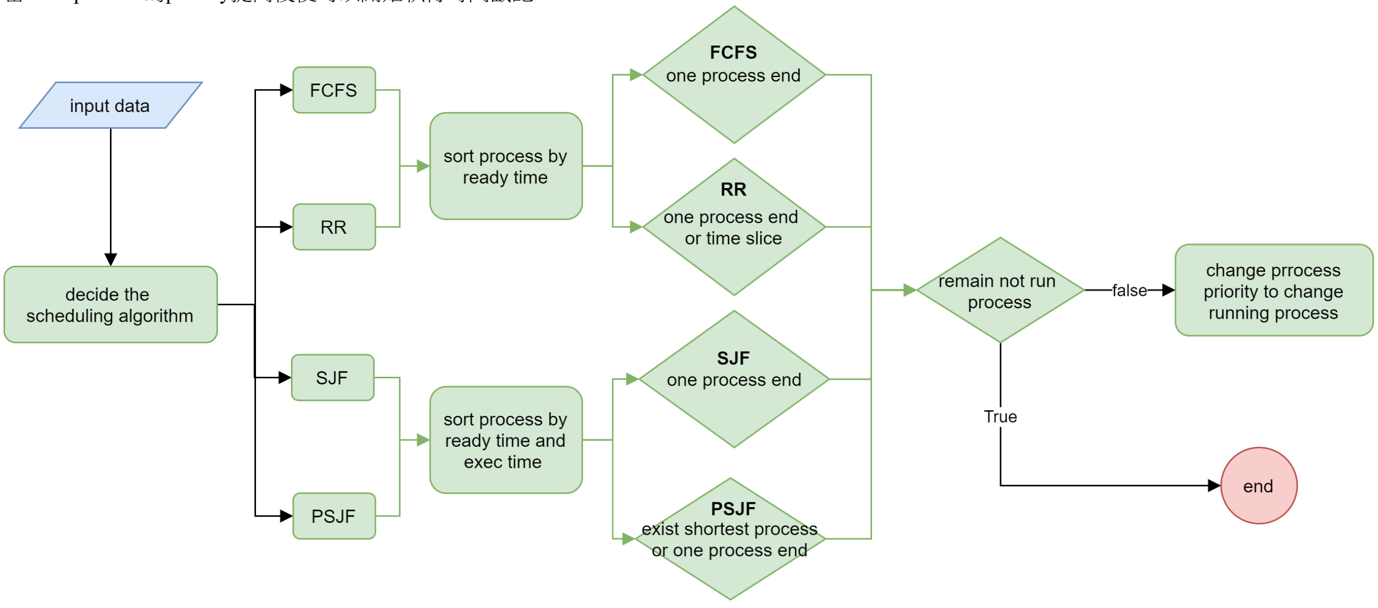
- `main`: 讀資料，選定要求的排程演算法，call 相對應的schedule function
- `scheduler`: 負責決定排程,固定該process的CPU core,有四種排程的函數:
  void schedule_FCFS(Process[], int);
  void schedule_RR(Process[], int);
  void schedule_SJF(Process[], int);
  void schedule_PSJF(Process[], int);
  分別可以決定在什麼時間需要fork或是stop某個process
- `process`: 負責跑child process的內容，及存放可以調整process 優先權的函數

兩個額外的syscall:

- `my_gettime`: 利用gettimeofday()來獲取系統時間
- `my_print`: 用來執行printk()

邏輯簡述:

- read input:讀檔及辨識要用哪種演算法模式
- sort process set:根據排程模式的不同，會有不同的sorting comparator 例如SJF要根據start time 做排序，再根據exec time 排第二次
- schedule_*():parent process 會決定甚麼時候會有event發生，要開始或終止process就做相對應的priority設定，若process start time到來時，會去call process.c裡面的fork()來建立prrocess 並且將優先權降到最低
- 當child process 的priority提高後便可以開始執行時間戳記

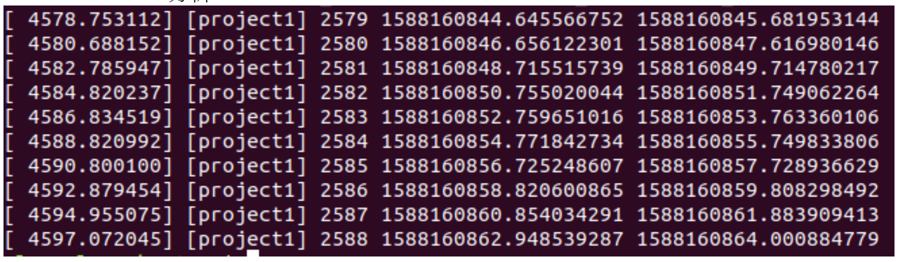

input data
decide the scheduling algorithm
FCFS
RR
SJF
PSJF
sort process by ready time
sort process by ready time and exec time
FCFS one process end
RR one process end or time slice
SJF one process end
PSJF exist shortest process or one process end
remain not run process
false
change prrocess priority to change running process
True
end

2.核心版本

- kernel 4.15.8(HW1的版本)

3.結果分析

- time measurement 分析:



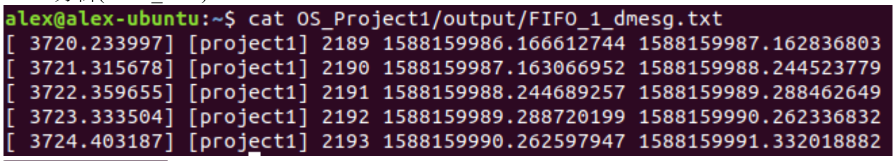
```
[ 4578.753112] [project1] 2579 1588160844.645566752 1588160845.681953144
[ 4580.688152] [project1] 2580 1588160846.656122301 1588160847.616980146
[ 4582.785947] [project1] 2581 1588160848.715515739 1588160849.714780217
[ 4584.820237] [project1] 2582 1588160850.755020044 1588160851.749062264
[ 4586.834519] [project1] 2583 1588160852.759651016 1588160853.763360106
[ 4588.820992] [project1] 2584 1588160854.771842734 1588160855.749833806
[ 4590.800100] [project1] 2585 1588160856.725248607 1588160857.728936629
[ 4592.879454] [project1] 2586 1588160858.820600865 1588160859.808298492
[ 4594.955075] [project1] 2587 1588160860.854034291 1588160861.883909413
[ 4597.072045] [project1] 2588 1588160862.948539287 1588160864.000884779
```

  end - start:
  1.036386489868164
  0.9608578681945801
  0.9992644786834717
  0.994042158126831
  1.003709077835083
  0.9779911041259766
  1.003688097000122
  0.9876978397369385
  1.0298752784729004

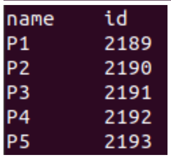
1.0523455142974854
avg:
1.0045857906341553
平均每500單位時間約一秒鐘，但是也有長有短，估計是因為context switch跟CPU沒有辦法達到穩定100%使用率有關，誤差大越0.4%

以下的表格分析 都為該process停止計時的時間扣除第一個process ready的時間(代表dmesg有紀錄的最小值)

- FCFS分析(FIFO_1.txt):



```
alex@alex-ubuntu:~$ cat OS_Project1/output/FIFO_1_dmesg.txt
[ 3720.233997] [project1] 2189 1588159986.166612744 1588159987.162836803
[ 3721.315678] [project1] 2190 1588159987.163066952 1588159988.244523779
[ 3722.359655] [project1] 2191 1588159988.244689257 1588159989.288462649
[ 3723.333504] [project1] 2192 1588159989.288720199 1588159990.262336832
[ 3724.403187] [project1] 2193 1588159990.262597947 1588159991.332018882
```



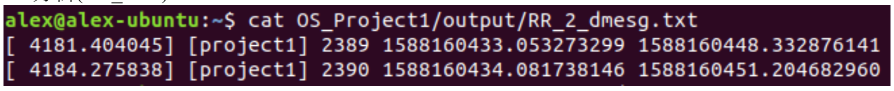
```
name    id
P1      2189
P2      2190
P3      2191
P4      2192
P5      2193
```

process end time - start time:

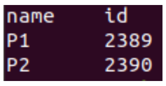
p name	理論	實際	誤差
p1	1	0.9962241649627686	-0.37%
p2	2	2.077911138534546	3.89%
p3	3	3.12185001373291	4.06%
p4	4	4.095724105834961	2.39%
p5	5	5.165406227111816	3.31%

誤差來源預估為context switch 跟set priority的time delay有關

- RR分析(RR_2.txt):

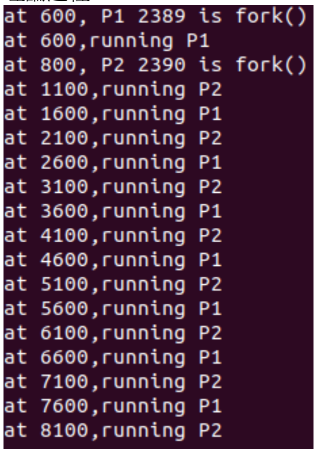


```
alex@alex-ubuntu:~$ cat OS_Project1/output/RR_2_dmesg.txt
[ 4181.404045] [project1] 2389 1588160433.053273299 1588160448.332876141
[ 4184.275838] [project1] 2390 1588160434.081738146 1588160451.204682960
```



```
name    id
P1      2389
P2      2390
```

理論過程:



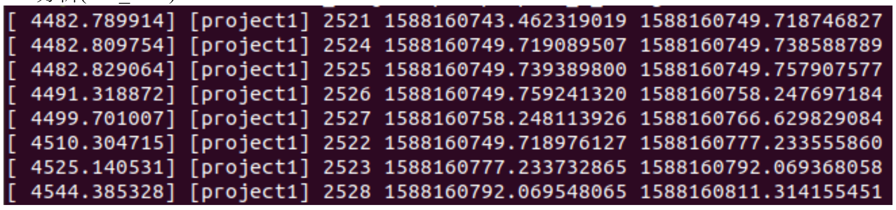
```
at 600, P1 2389 is fork()
at 600,running P1
at 800, P2 2390 is fork()
at 1100,running P2
at 1600,running P1
at 2100,running P2
at 2600,running P1
at 3100,running P2
at 3600,running P1
at 4100,running P2
at 4600,running P1
at 5100,running P2
at 5600,running P1
at 6100,running P2
at 6600,running P1
at 7100,running P2
at 7600,running P1
at 8100,running P2
```

process end time - start time:

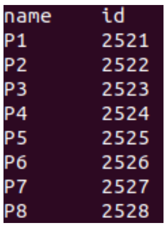
p name	理論	實際	誤差
p1	16.5	15.279603004455566	-7.39%
p2	19.2	18.15140986442566	-5.46%

這裡的實際比理論快一些 猜測是因為set priority會有delay造成每次的slice都會多跑一些些

- SJF分析(SJF_3.txt):

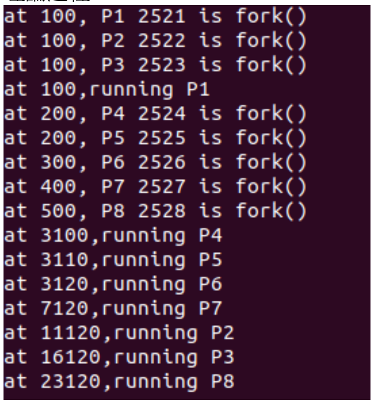


```
[ 4482.789914] [project1] 2521 1588160743.462319019 1588160749.718746827
[ 4482.809754] [project1] 2524 1588160749.719089507 1588160749.738588789
[ 4482.829064] [project1] 2525 1588160749.739389800 1588160749.757907577
[ 4491.318872] [project1] 2526 1588160749.759241320 1588160758.247697184
[ 4499.701007] [project1] 2527 1588160758.248113926 1588160766.629829084
[ 4510.304715] [project1] 2522 1588160749.718976127 1588160777.233555860
[ 4525.140531] [project1] 2523 1588160777.233732865 1588160792.069368058
[ 4544.385328] [project1] 2528 1588160792.069548065 1588160811.314155451
```



```
name    id
P1      2521
P2      2522
P3      2523
P4      2524
P5      2525
P6      2526
P7      2527
P8      2528
```

理論過程:



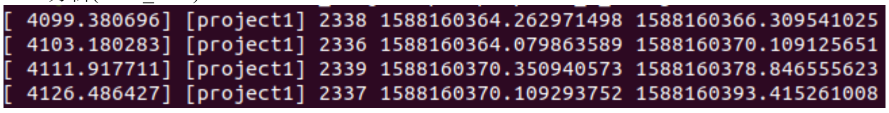
```
at 100, P1 2521 is fork()
at 100, P2 2522 is fork()
at 100, P3 2523 is fork()
at 100,running P1
at 200, P4 2524 is fork()
at 200, P5 2525 is fork()
at 300, P6 2526 is fork()
at 400, P7 2527 is fork()
at 500, P8 2528 is fork()
at 3100,running P4
at 3110,running P5
at 3120,running P6
at 7120,running P7
at 11120,running P2
at 16120,running P3
at 23120,running P8
```

process end time - start time:

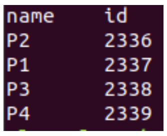
p name	理論	實際	誤差
p1	6	6.256427764892578	4.27%
p4	6.02	6.2762696743011475	4.25%
p5	6.04	6.295588493347168	4.23%
p6	14.04	14.785377979278564	5.30%
p7	22.04	23.16751003265381	5.11%
p2	32.04	33.77123665809631	5.40%
p3	46.04	48.607048988342285	5.47%
p8	64.04	33.77123665809631	5.95%

一樣預估是context switch 及for 迴圈造成的誤差

- PSJF分析(PSJF_4.txt):

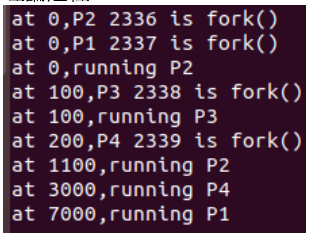


```
[ 4099.380696] [project1] 2338 1588160364.262971498 1588160366.309541025
[ 4103.180283] [project1] 2336 1588160364.079863589 1588160370.109125651
[ 4111.917711] [project1] 2339 1588160370.350940573 1588160378.846555623
[ 4126.486427] [project1] 2337 1588160370.109293752 1588160393.415261008
```



```
name    id
P2      2336
P1      2337
P3      2338
P4      2339
```

理論過程:



```
at 0,P2 2336 is fork()
at 0,P1 2337 is fork()
at 0,running P2
at 100,P3 2338 is fork()
at 100,running P3
at 200,P4 2339 is fork()
at 1100,running P2
at 3000,running P4
at 7000,running P1
```

process end time - start time:

p name	理論	實際	誤差
p3	2.2	2.046569585800171	-6.97%
p2	6.0	5.84615421295166	-2.56%
p4	14	14.583584308624268	4.16%
p6	28	29.152289628982544	4.11%

總結:誤差會根據時間越久累積越多但是因為同時分母也較大，所以誤差百分比相近，誤差因素有很多例如CPU被背景程式佔據，螢幕錄影時跟parent process搶資源，priority改變到生效的時間delay，parent process的event判斷，context switch等等皆有可能影響實際情況，尤其在虛擬機跑更會被本來的OS影響CPU的功效。