OS2020_Project1 _ Report

1. 設計

CPU:
使用雙核，透過 *set_affinity* 函式做設定，一顆CPU（0）用在scheduler, 一顆（1）用在fork 出來的child 身上。

Syscall:

1. 使用 getnstimeofday 來取得現在時間的 sec, nanosec。
2. 使用 printk 將讀入的字串印到 dmesg。

Process:
process 中使用fork來執行生成process的過程。在生出小孩之後便會將小孩assign到另外一顆CPU。這邊我是用parent來記錄小孩生出來時間以及結束的時間，並將時間綁在child 的 structure身上。

Schedule:
程式一開始會將priority設成最高，當現在時間符合process的ready time，就會fork出去，而對於每個新fork出的process都會先給他最低的prioity，這邊我使用sched_setschedule中的SCHED_FIFO（搭配priority 最低為1, 最高99）來執行，由於一開始的priority是最高，因此現在fork出來的小孩不會馬上執行。每一個time unit 開始之後，會根據現在schedule的type（FIFO, SJF, PSJF, RR）去找到下一個要執行的process時，此時就會將要執行的priority設成最高。若目前沒有其他process在ready，schedule就會執行一個time unit，重複此步驟直所有的process都完成。

2. 核心版本

linux 4.14.25

3. 比較實際結果與理論結果

[以影片demo測資為例]
以下將dmesg每個process得到的時間減去最開始的時間之後轉換成像範例測資的time stemp方便對照執行順序是否正確以及可看出中間的誤差。
每一筆測資誤差值的算法透過以下公式計算：

`(end_time-start_time)- unit time*理論上幾個unit`
unit time 則是根據我的TIME_MEASUREMENT得到

```
[ 4670.873086] [Project1] 18427 1588136456.349362955
1588136457.113943449
[ 4670.873087] [Project1] 18428 1588136457.834976041
1588136458.580312451
[ 4670.873088] [Project1] 18429 1588136459.304785746
1588136460.051599736
[ 4670.873089] [Project1] 18430 1588136460.784878098
1588136461.520379629
[ 4670.873090] [Project1] 18431 1588136462.243616051
1588136462.993676343
[ 4670.873091] [Project1] 18432 1588136463.716788076
1588136464.464791301
[ 4670.873091] [Project1] 18433 1588136465.185400116
1588136465.954007042
[ 4670.873092] [Project1] 18434 1588136466.700139077
1588136467.443080943
[ 4670.873093] [Project1] 18435 1588136468.162013930
1588136468.899233823
```

```
[ 4670.873094] [Project1] 18436 1588136469.604922523
1588136470.342852663
```
=> unit time 約為：0.74769949912(其實這邊每一個跑起來都會有點誤差，因此我取用平均數)
理論值：-0.00366

```
FIFO 1:
P1: from 0 to 500
P2: from 500 to 1000
P3: from 1000 to 1500
P4: from 1500 to 2000
P5: from 2000 to 2500
```
實際結果：

```
FIFO 1:
P1: from 0 to 507
P2: from 507 to 1005
P3: from 1005 to 1503
P4: from 1503 to 1999
P5: from 1999 to 2502
[ 2048.782537] [Project1] 17536 1588133843.998875752
1588133844.757699205
[ 2048.782538] [Project1] 17537 1588133844.757706746
1588133845.502052820
[ 2048.782539] [Project1] 17538 1588133845.502060932
1588133846.247739527
[ 2048.782540] [Project1] 17539 1588133846.247747138
1588133846.989367245
[ 2048.782541] [Project1] 17540 1588133846.989371489
1588133847.741033436
```
=>誤差值約為：-0.00366秒

理論值：

```
RR 3:
P1: from 1200 to 20700
P2: from 2400 to 19900
P3: from 4400 to 18900
P4: from 5900 to 31200
P5: from 6900 to 30200
P6: from 7900 to 28200
```

實際結果：

```
RR 3:
P1: from 1200 to 19835
P2: from 2686 to 20321
P3: from 4200 to 17882
P4: from 6184 to 30534
P5: from 6679 to 29557
P6: from 7179 to 27600
[ 3837.134497] [Project1] 18089 1588135597.225447566
1588135617.685300048
[ 3837.134498] [Project1] 18087 1588135592.737801990
1588135620.604908634
[ 3837.134499] [Project1] 18088 1588135594.960694093
1588135621.331454146
[ 3837.134500] [Project1] 18092 1588135601.680045667
1588135632.217408989
[ 3837.134501] [Project1] 18091 1588135600.932367362
1588135635.143009163
[ 3837.134502] [Project1] 18090 1588135600.190951587
1588135636.604256783
```

=> 誤差值約為：-0.995515秒

理論值：

```
SJF 4:
P1: from 0 to 3000
```

```
P2: from 3000 to 4000
P3: from 4000 to 8000
P4: from 9000 to 11000
P5: from 8000 to 9000
```
實際結果：

```
SJF 4 :
P1: from 0 to 3059
P2: from 3059 to 4063
P3: from 4063 to 8081
P4: from 9063 to 11036
P5: from 8081 to 9063
[ 4452.370307] [Project1] 18354 1588136235.336487834
1588136239.912261548
[ 4452.370308] [Project1] 18355 1588136239.912267231
1588136241.412353555
[ 4452.370309] [Project1] 18356 1588136241.412360863
1588136247.421455890
[ 4452.370310] [Project1] 18360 1588136247.421463638
1588136248.889843995
[ 4452.370311] [Project1] 18357 1588136248.889849028
1588136251.840066292
```
=> 誤差約為：+0.05418944381秒

理論值：

```
PSJF 2:
P1: from 0 to 4000
P2: from 1000 to 2000
P3: from 4000 to 11000
P4: from 5000 to 7000
P5: from 7000 to 8000
```
實際結果：

```
PSJF 2:
P1: from 0 to 4011
P2: from 1012 to 2015
P3: from 4011 to 10960
P4: from 5009 to 7038
P5: from 7039 to 8028
[ 2927.134335] [Project1] 17801 1588134711.655906321
1588134713.156083629
[ 2927.134336] [Project1] 17800 1588134710.141379599
1588134716.140299227
[ 2927.134337] [Project1] 17803 1588134717.632964666
1588134720.667457595
[ 2927.134338] [Project1] 17804 1588134720.667524558
1588134722.147153281
[ 2927.134339] [Project1] 17802 1588134716.140305643
1588134726.532007937
```

=>誤差值約為：-0.05876秒

4. 可能原因

1. context switch
2. 使用for while迴圈的時間
3. 電腦每次跑的結果其實之間都有誤差，Time Measurement也只是參考