OS Project1 Report

資工二 b07902124 鄭世朋

設計

說明：

- 我的程式的start time是程式被創造的時間，然後我的程式的stdout是按照程式的結束時間印出。

Two CPUs

- 我使用了兩個CPU，一個作為scheduler，而另一個則是被創造出來的child process所使用。

程式流程

- 首先讀入所有input並確定此次的scheduling policy，並把scheduler的CPU設為0號，再來，將所有的process依照ready time排序，接者就進入while迴圈scheduling的階段。
- while迴圈的每一個週期都是一個unit time，而while迴圈一開始會先檢查有哪一個程式已經到了ready time，並fork()創造一個對應的程式，然後將其assign到CPU 1號上，接著scheduler會去檢查上一個unit time在跑的程式是否已經結束，若已經結束則output程式的名字和pid到stdout並printk，接下來，scheduler會根據policy去找這一個unit time輪到誰去跑，並實際跑一個unit_time，最後更新已經跑了多久並減少這一個unit time跑的process的execution time然後開啟下一個迴圈。
- Policy 為 RR 在決定這一個unit time是誰該跑時，若到了time slice的話，會是從當前要被停止的程式的下一個開始判斷，下一個是依一一開始按照ready time排序的順序
- fork 出來的process會被scheduler用 sched_setscheduler控制priority去達到schedule的效果。

核心版本

- linux 4.14.25
- ubuntu 16.04

實際結果與理論結果差異

計算理論時間

- 計算方式為使用TIME_MESUREMENT.txt 並得到平均run 500 unit time需要0.9s，如附圖(同output/TIME_MESUREMENT_dmesg.txt)

```
zhengshipeng@ubuntu:~/project/output$ cat TIME_MEASUREMENT_dmesg.txt
[ 3112.752796] [project1] 6605 1587971190.831669415 1587971191.732816905
[ 3114.441669] [project1] 6606 1587971192.527496408 1587971193.421744356
[ 3116.101679] [project1] 6607 1587971194.195677813 1587971195.081809995
[ 3117.782649] [project1] 6608 1587971195.855186685 1587971196.762833734
[ 3119.445413] [project1] 6609 1587971197.537529082 1587971198.425655114
[ 3121.111498] [project1] 6610 1587971199.196937047 1587971200.091794853
[ 3122.774299] [project1] 6611 1587971200.863970630 1587971201.754649912
[ 3124.482596] [project1] 6612 1587971202.526940353 1587971203.463004364
[ 3126.169057] [project1] 6613 1587971204.260746537 1587971205.149520233
[ 3127.830359] [project1] 6614 1587971205.926910486 1587971206.810877168
```

- 我使用同demo的那幾筆測資作為實驗的對象

差異

- 可以觀察到以下兩個事實
  - 實際結果皆會略高於理論結果
- 造成此結果的原因可能有以下幾種可能
  - context switch 帶來的overhead
  - PSJF 跟 SJF在尋找有最短execution time的process時帶來的overhead
  - 實際上while迴圈的每一圈其實run超過1 unit_time
  - system loading 也會影響(有一個例子提供在SJF_4.txt下面)

實際結果

- FIFO_1.txt
  平均為每500unit_time需0.945s左右，最接近理論值

```
zhengshipeng@ubuntu:~/project/output$ cat FIFO_1_dmesg.txt
[ 1668.104215] [project1] 5881 1587969746.182954997 1587969747.102638025
[ 1669.034924] [project1] 5882 1587969746.182700580 1587969748.033424745
[ 1670.009317] [project1] 5883 1587969746.182746339 1587969749.007899037
[ 1670.974662] [project1] 5884 1587969746.182798752 1587969749.973327003
[ 1671.914643] [project1] 5885 1587969746.199005411 1587969750.913386522
```

- PSJF_2.txt
  平均為每500unit_time需1.03s左右，誤差最大

```
zhengshipeng@ubuntu:~/project/output$ cat PSJF_2_dmesg.txt
[ 1994.096819] [project1] 6010 1587970071.034998320 1587970073.053449558
[ 1998.663851] [project1] 6009 1587970069.009138261 1587970077.619688761
[ 2004.755403] [project1] 6013 1587970079.839017846 1587970083.710177374
[ 2006.694254] [project1] 6014 1587970083.730972401 1587970085.648688591
[ 2012.680679] [project1] 6011 1587970074.727018087 1587970091.634061194
```

- RR_3.txt
  平均為每500unit_time需0.949s左右，略高於理論值

```
zhengshipeng@ubuntu:~/project/output$ cat RR_3_dmesg.txt
[ 2139.346919] [project1] 6111 1587970190.346986926 1587970218.292789344
[ 2140.328963] [project1] 6109 1587970184.398129201 1587970219.275156763
[ 2141.225581] [project1] 6110 1587970187.443023741 1587970220.172066835
[ 2159.158706] [project1] 6114 1587970195.127006654 1587970238.111034446
[ 2162.933329] [project1] 6113 1587970193.362988700 1587970241.886889410
[ 2164.696633] [project1] 6112 1587970193.354983961 1587970243.650766526
```

- SJF_4.txt
  平均為每500unit_time需0.993s左右，高於理論值

```
zhengshipeng@ubuntu:~/project$ cat output/SJF_4_dmesg.txt
[ 8577.775217] [project1] 4276 1588149400.348431285 1588149405.764829232
[ 8579.582537] [project1] 4277 1588149404.225196659 1588149407.571937361
[ 8586.819951] [project1] 4278 1588149405.765031407 1588149414.808509981
[ 8588.621994] [project1] 4280 1588149414.808711474 1588149416.610342663
[ 8592.216865] [project1] 4279 1588149411.465177117 1588149420.204794510
```

  在CPU loading較重的時候，平均run每500unit_time 可以來到1.08s左右，有顯著的差異

```
zhengshipeng@ubuntu:~/project/output$ cat SJF_4_dmesg.txt
[ 2387.075139] [project1] 6235 1587970460.419199866 1587970466.049752553
[ 2389.010450] [project1] 6236 1587970464.422992026 1587970467.985164190
[ 2397.184370] [project1] 6242 1587970466.049926102 1587970476.159507605
[ 2399.120567] [project1] 6244 1587970476.159726900 1587970478.095806130
[ 2403.117334] [project1] 6243 1587970472.023087852 1587970482.092779758
```