Operating System Project 1

b05502087 資工系 王竑睿

1 設計

- 本次實作採用兩個 cpu core，一個給 scheduler，另外一個給 create 出來的 child processes。
  - 實作上以 sched_setaffinity 來綁定的指定的 CPU 上
- 對於每一個 process，只要進入 ready 狀態，就會 fork 出一個 child process 來運行助教提供的 for-loop

```
#define UNIT_TIME() { volatile unsigned long i;for (i = 0; i < 1000000UL; i++);}
```

- 但是會隨時以 sched_setscheduler 來調整各 processes 的 priority
  - 利用 SCHED_IDLE 模擬 process 脫離 CPU
  - 利用 SCHED_OTHER 模擬 process 進入 CPU

1.1 SJF

- 對 child process 的 execution time 以 stdlib 內建的 qsort 進行排序
- execution time 愈小者愈先執行

1.2 FIFO

- 對 child process 的 ready time 以 stdlib 內建的 qsort 進行排序
- 愈早 ready 的 process 愈先開始執行

1.3 Round Robin

- maintain 一個 queue，當有 process 到了 ready 的時間時，scheduler 即會將其加入 queue 中
- 每過 500 個 time_unit 就會進行一次 context switch 以讓其他 process 有機會得到 CPU。實作上是將當下在運行的 process 從 queue 的最前端移至最末端
- 當 process 執行時間結束時，即從 queue 中 pop 掉

2 核心版本

- 本次作業使用的 kernel 版本為 linux-4.14.25

3 比較實際結果與理論結果，並解釋造成差異的原因

3.1 計算 time unit

- 首先先計算 time_unit 的時長
- 將 TIME_MEASUREMENT.txt 內的十個 processes 跑過一遍
  - 每次可以從 dmesg 中得到一個 start time 和 end time
  - TIME_MEASUREMENT.txt 內每一個 process 的 execution time 都是 500 個 units。
  - 因此可以估算一個 unit 約過了多少時間

```
[  331.903190] [Project1] 1656 1588169021.571703675 1588169022.557661504
[  333.837042] [Project1] 1657 1588169023.510971885 1588169024.491334029
[  335.786042] [Project1] 1658 1588169025.453200865 1588169026.440157680
[  337.780972] [Project1] 1659 1588169027.419319312 1588169028.434891259
[  339.748523] [Project1] 1661 1588169029.392851626 1588169030.402261406
[  341.749225] [Project1] 1663 1588169031.392520100 1588169032.402759686
[  343.751109] [Project1] 1664 1588169033.416806384 1588169034.404447389
[  345.797731] [Project1] 1665 1588169035.401699438 1588169036.450863606
[  347.810082] [Project1] 1667 1588169037.418152477 1588169038.463000834
[  349.750570] [Project1] 1668 1588169039.419829531 1588169040.403284999
```

- $Time\_Unit = \frac{mean(endTime - startTime)}{500} = \frac{sum(endTime - startTime)}{5000}$
- 本次實作我在 virtual box 上測得的 time unit 為 0.002011

3.2 觀察 FIFO 執行時間

```
OverallstSecond 1588172788.398846900
create 0.5922668
create 3.172446361
create 3.179945415
end 4.258547845
create 4.269936019
end 5.323017899
end 6.18446744073436810455
end 7.18446744073453220367
```

```
FIFO
4
P1 0 2000
P2 500 500
P3 500 200
P4 1500 500
```

- 由圖可以看出 P2，P3 理論上必須同時 ready，同時被 scheduler create。實際上也確實如此
- P2 理論上應該於 $2500 * TIME\_UNIT \approx 2500 * 0.002 = 5$ 結束。實際上也於程式開始後 5.32 秒結束
- P3 理論上應該於 $2700 * TIME\_UNIT \approx 2700 * 0.002 = 5.4$ 結束。實際上則在程式開始後 6.18 秒結束
- P4 理論上應該於 $3200 * TIME\_UNIT \approx 3200 * 0.002 = 6.4$ 結束。實際上則在程式開始後 7.18 秒結束
- 這些 delay 可能是在程式之間，如 I/O、函式之間的傳遞資料產生的 overhead。

- 從程式起始到 process create 出來的誤差特別大，可能是這之間經過的 operations 相對於到 process end 更多且耗時。