上海交通大學

SHANGHAI JIAOTONG UNIVERSITY

操作系统课程设计报告 - Project4

Scheduling Algorithms

姓名：薛春宇

学号：518021910698

完成时间：2020/6/2

一、实验目的

1. 在课程学习的基础上，充分掌握五种调度算法的原理和使用；

2. 利用所学知识，成功实现上述五种调度算法；

二、实验内容

实现 fcfs (first come first service)、sjf (short job first)、priority、rr (round robin)、priority_rr 五种调度算法。

三、内容

1. 实验原理：

基于课程中所学习的关于上述五种调度算法的基础知识，我们可以对这些算法的思路有一个基本的认识：

（1）First-come, first-served (FCFS): 按照 CPU 的要求顺序顺次执行；

（2）Shortest-job-first (SJF): 按照任务的 CPU burst 长度从小到大执行；

（3）Priority scheduling: 根据优先级从大到小顺次执行；

（4）Round-robin (RR) scheduling: 每个任务都只执行一个固定的时间片；

（5）Priority with round-robin: 在根据优先级从大到小顺次执行的基础上，对于优先级相同的任务，采用 RR 进行轮转，所有任务每次都只能执行一个固定的时间片；

在了解这些调度算法的基本原理之后，我们需要做的就是利用学过的 coding 知识将其实现。代码模版中给出的数据结构是链表，因此本次实验在实现过程中同样需要对链表较为熟练的运用。

2. 实验代码：见附件 **/code/schedule_fcfs.c** 、 **schedule_sjf.c** 、**schedule_priority.c**、**schedule_rr.c**、**schedule_priority_rr.c**

3. 实验结果截图：

（1）FCFS：

```
dicardo@ubuntu:~/桌面/OSProject4$ make fcfs
gcc -Wall -g -c driver.c
gcc -Wall -g -c list.c
gcc -Wall -g -c CPU.c
gcc -Wall -g -c schedule_fcfs.c
gcc -Wall -g -g -o fcfs driver.o schedule_fcfs.o list.o CPU.o
dicardo@ubuntu:~/桌面/OSProject4$ ./fcfs test.txt
Running task = [T1] [4] [20] for 20 units.
The unique task identifier of task [T1] is: [1]
-----------------------------------------------
Running task = [T2] [3] [25] for 25 units.
The unique task identifier of task [T2] is: [2]
-----------------------------------------------
Running task = [T3] [3] [25] for 25 units.
The unique task identifier of task [T3] is: [3]
-----------------------------------------------
Running task = [T4] [5] [15] for 15 units.
The unique task identifier of task [T4] is: [4]
-----------------------------------------------
The average turnaround time is: 55.000000 units
The average wait time is: 33.750000 units
The average response time is: 33.750000 units
dicardo@ubuntu:~/桌面/OSProject4$
```

（2）SJF：

```
dicardo@ubuntu:~/桌面/OSProject4$ make sjf
gcc -Wall -g -c schedule_sjf.c
gcc -Wall -g -o sjf driver.o schedule_sjf.o list.o CPU.o
dicardo@ubuntu:~/桌面/OSProject4$ ./sjf test.txt
Running task = [T4] [5] [15] for 15 units.
The unique task identifier of task [T4] is: [4]
-----------------------------------------------
Running task = [T1] [4] [20] for 20 units.
The unique task identifier of task [T1] is: [1]
-----------------------------------------------
Running task = [T3] [3] [25] for 25 units.
The unique task identifier of task [T3] is: [3]
-----------------------------------------------
Running task = [T2] [3] [25] for 25 units.
The unique task identifier of task [T2] is: [2]
-----------------------------------------------
The average turnaround time is: 48.750000 units
The average wait time is: 27.500000 units
The average response time is: 27.500000 units
dicardo@ubuntu:~/桌面/OSProject4$
```

（3）Priority：

```
dicardo@ubuntu:~/桌面/OSProject4$ make priority
gcc -Wall -g -c schedule_priority.c
gcc -Wall -g -o priority driver.o schedule_priority.o list.o CPU.o
dicardo@ubuntu:~/桌面/OSProject4$ ./priority test.txt
Running task = [T4] [5] [15] for 15 units.
The unique task identifier of task [T4] is: [4]
-----------------------------------------------
Running task = [T1] [4] [20] for 20 units.
The unique task identifier of task [T1] is: [1]
-----------------------------------------------
Running task = [T3] [3] [25] for 25 units.
The unique task identifier of task [T3] is: [3]
-----------------------------------------------
Running task = [T2] [3] [25] for 25 units.
The unique task identifier of task [T2] is: [2]
-----------------------------------------------
The average turnaround time is: 48.750000 units
The average wait time is: 27.500000 units
The average response time is: 27.500000 units
dicardo@ubuntu:~/桌面/OSProject4$
```

(4) RR:

```
dicardo@ubuntu:~/桌面/OSProject4$ make rr
gcc -Wall -g -c driver.c
gcc -Wall -g -c list.c
gcc -Wall -g -c CPU.c
gcc -Wall -g -c schedule_rr.c
gcc -Wall -g -o rr driver.o schedule_rr.o list.o CPU.o
dicardo@ubuntu:~/桌面/OSProject4$ ./rr test.txt
Running task = [T1] [4] [20] for 10 units.
The unique task identifier of task [T1] is: [1]
-----------------------------------------------
Running task = [T2] [3] [25] for 10 units.
The unique task identifier of task [T2] is: [2]
-----------------------------------------------
Running task = [T3] [3] [25] for 10 units.
The unique task identifier of task [T3] is: [3]
-----------------------------------------------
Running task = [T4] [5] [15] for 10 units.
The unique task identifier of task [T4] is: [4]
-----------------------------------------------
Running task = [T1] [4] [10] for 10 units.
The unique task identifier of task [T1] is: [1]
-----------------------------------------------
Running task = [T2] [3] [15] for 10 units.
The unique task identifier of task [T2] is: [2]
-----------------------------------------------
Running task = [T3] [3] [15] for 10 units.
The unique task identifier of task [T3] is: [3]
-----------------------------------------------
Running task = [T4] [5] [5] for 5 units.
The unique task identifier of task [T4] is: [4]
-----------------------------------------------
Running task = [T2] [3] [5] for 5 units.
The unique task identifier of task [T2] is: [2]
-----------------------------------------------
Running task = [T3] [3] [5] for 5 units.
The unique task identifier of task [T3] is: [3]
-----------------------------------------------
The average turnaround time is: 72.500000 units
The average wait time is: 51.250000 units
The average response time is: 7.500000 units
dicardo@ubuntu:~/桌面/OSProject4$
```

（5）Priority_RR：

```
dicardo@ubuntu:~/桌面/OSProject4$ make priority_rr
gcc -Wall -g    -c -o schedule_priority_rr.o schedule_priority_rr.c
gcc -Wall -g -o priority_rr driver.o schedule_priority_rr.o list.o CPU.o
dicardo@ubuntu:~/桌面/OSProject4$ ./priority_rr test.txt
Running task = [T4] [5] [15] for 10 units.
The unique task identifier of task [T4] is: [4]
-----------------------------------------------
Running task = [T4] [5] [5] for 5 units.
The unique task identifier of task [T4] is: [4]
-----------------------------------------------
Running task = [T1] [4] [20] for 10 units.
The unique task identifier of task [T1] is: [1]
-----------------------------------------------
Running task = [T1] [4] [10] for 10 units.
The unique task identifier of task [T1] is: [1]
-----------------------------------------------
Running task = [T2] [3] [25] for 10 units.
The unique task identifier of task [T2] is: [2]
-----------------------------------------------
Running task = [T3] [3] [25] for 10 units.
The unique task identifier of task [T3] is: [3]
-----------------------------------------------
Running task = [T2] [3] [15] for 10 units.
The unique task identifier of task [T2] is: [2]
-----------------------------------------------
Running task = [T3] [3] [15] for 10 units.
The unique task identifier of task [T3] is: [3]
-----------------------------------------------
Running task = [T2] [3] [5] for 5 units.
The unique task identifier of task [T2] is: [2]
-----------------------------------------------
Running task = [T3] [3] [5] for 5 units.
The unique task identifier of task [T3] is: [3]
-----------------------------------------------
The average turnaround time is: 53.750000 units
The average wait time is: 32.500000 units
The average response time is: 23.750000 units
dicardo@ubuntu:~/桌面/OSProject4$
```

五、实验结果

在经过一天左右的 coding 之后完成该项目。根据代码的测试及运行结果来看，本次实验取得成功。

六、实验反思

总的来说，本次实验的难度不大，只要在课堂上较为熟练地掌握这五种调度算法的基本原理及过程，再结合相应的实现操作即可完成。

但是，在实现这些代码的过程中，由于使用了链表结构作为数据载体，因此在进行链表操作的时候仍然可能出现一些难以发现的 bug。

在这次实验中，我对几种调度算法的掌握程度得到了一个很大的提升，同时自己在面对 bug 时的处理能力也得到了锻炼。虽然自己在 coding 过程中仍旧存

在一些例如代码逻辑不够简练、初次编写会存在较多低级 bug 等问题，但我相信，随着自己做的项目越来越多，我的代码能力也会越来越得到提升。