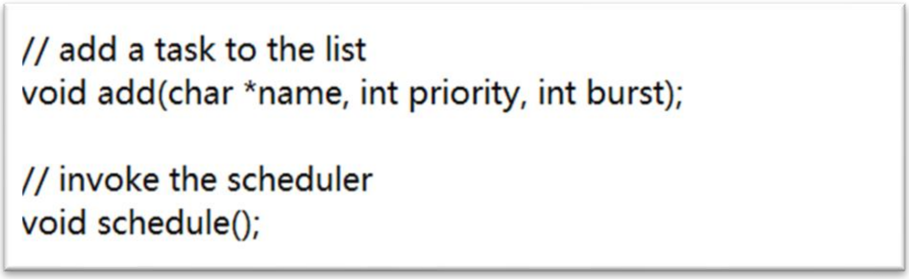
Project 4: Scheduling Algorithms

课本中提供了两种可供选择的语言——C 或者 java。我选择使用 C 语言来完成该项目。

（一）问题分析

这个项目需要实现 5 种调度算法，需要我补充的函数是 schedulers.h 中的 add()和 schedule()。我需要对于每一种调度算法实现这两个函数，用在 driver.c 的 main 函数中。



```
// add a task to the list
void add(char *name, int priority, int burst);

// invoke the scheduler
void schedule();
```

（二）算法实现

一、schedule_fcfs.c

1、实现细节

（1）add()

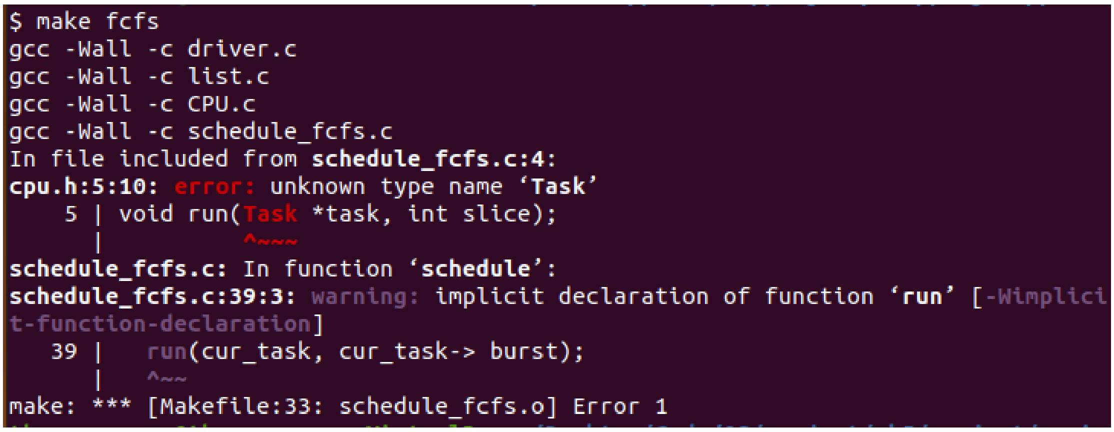
将新建的每个任务结点插入到表头。

（2）schedule()

根据先来先服务调度原则，最先来的任务在表的末尾，所以每次调度前都要遍历到表的末尾，调度表尾的任务，最后删除调度过的任务。

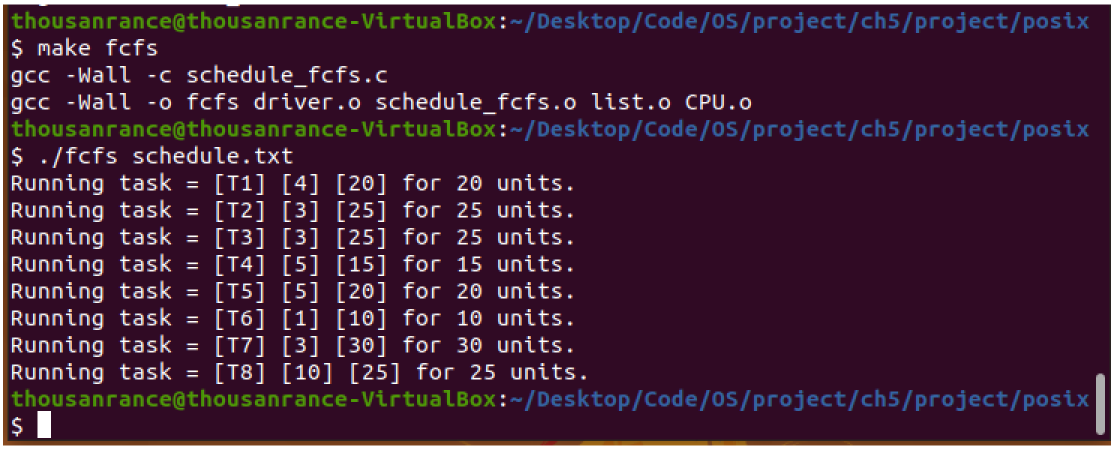
2、运行结果

（1）第一次 make 出现如下错误信息，经检查后发现，cpu.h 文件中没有 include task.h。



```
$ make fcfs
gcc -Wall -c driver.c
gcc -Wall -c list.c
gcc -Wall -c CPU.c
gcc -Wall -c schedule_fcfs.c
In file included from schedule_fcfs.c:4:
cpu.h:5:10: error: unknown type name ‘Task’
    5 | void run(Task *task, int slice);
      |          ^~~~
schedule_fcfs.c: In function ‘schedule’:
schedule_fcfs.c:39:3: warning: implicit declaration of function ‘run’ [-Wimplici
t-function-declaration]
   39 |   run(cur_task, cur_task-> burst);
      |   ^~~
make: *** [Makefile:33: schedule_fcfs.o] Error 1
```

（2）修改后运行正常。



```
thousanrance@thousanrance-VirtualBox:~/Desktop/Code/OS/project/ch5/project/posix
$ make fcfs
gcc -Wall -c schedule_fcfs.c
gcc -Wall -o fcfs driver.o schedule_fcfs.o list.o CPU.o
thousanrance@thousanrance-VirtualBox:~/Desktop/Code/OS/project/ch5/project/posix
$ ./fcfs schedule.txt
Running task = [T1] [4] [20] for 20 units.
Running task = [T2] [3] [25] for 25 units.
Running task = [T3] [3] [25] for 25 units.
Running task = [T4] [5] [15] for 15 units.
Running task = [T5] [5] [20] for 20 units.
Running task = [T6] [1] [10] for 10 units.
Running task = [T7] [3] [30] for 30 units.
Running task = [T8] [10] [25] for 25 units.
thousanrance@thousanrance-VirtualBox:~/Desktop/Code/OS/project/ch5/project/posix
$ 
```

二、schedule_sjf.c

1、实现细节

（1）add()

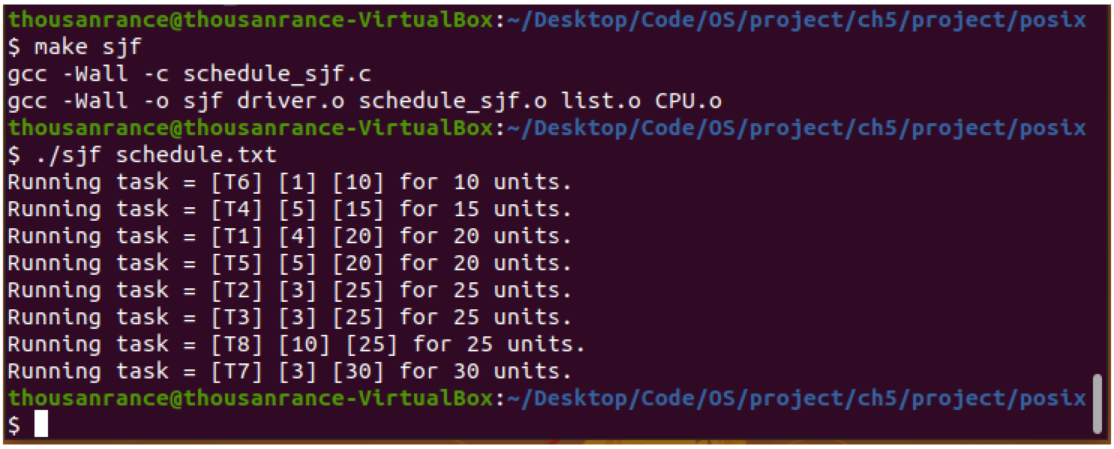
将新建的每个任务结点插入到表头。

（2）schedule()

根据最短服务时间调度原则，每次调度前先都要遍历任务表，找出 burst 最短的任务，调度该任务，最后删除调度过的任务。

在比较 burst 时使用“≤”而不是“＜”的意义是：如果 burst 相等，则按照 fcfs 原则调度。

2、运行结果



```
thousanrance@thousanrance-VirtualBox:~/Desktop/Code/OS/project/ch5/project/posix
$ make sjf
gcc -Wall -c schedule_sjf.c
gcc -Wall -o sjf driver.o schedule_sjf.o list.o CPU.o
thousanrance@thousanrance-VirtualBox:~/Desktop/Code/OS/project/ch5/project/posix
$ ./sjf schedule.txt
Running task = [T6] [1] [10] for 10 units.
Running task = [T4] [5] [15] for 15 units.
Running task = [T1] [4] [20] for 20 units.
Running task = [T5] [5] [20] for 20 units.
Running task = [T2] [3] [25] for 25 units.
Running task = [T3] [3] [25] for 25 units.
Running task = [T8] [10] [25] for 25 units.
Running task = [T7] [3] [30] for 30 units.
thousanrance@thousanrance-VirtualBox:~/Desktop/Code/OS/project/ch5/project/posix
$ 
```

三、schedule_priority.c

1、实现细节

（1）add()

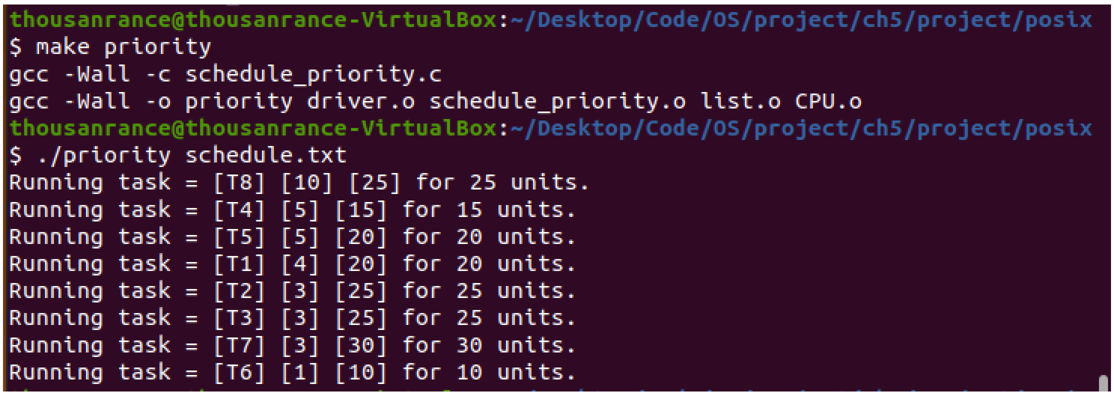
将新建的每个任务结点插入到表头。

（2）schedule()

根据优先级调度原则，每次调度前先都要遍历任务表，找出优先级最高，即 priority 最大的任务，调度该任务，最后删除调度过的任务。

在比较 priority 时使用“≥”而不是“＞”的意义是：如果 priority 相等，则按照 fcfs 原则调度。

2、运行结果



```
thousanrance@thousanrance-VirtualBox:~/Desktop/Code/OS/project/ch5/project/posix
$ make priority
gcc -Wall -c schedule_priority.c
gcc -Wall -o priority driver.o schedule_priority.o list.o CPU.o
thousanrance@thousanrance-VirtualBox:~/Desktop/Code/OS/project/ch5/project/posix
$ ./priority schedule.txt
Running task = [T8] [10] [25] for 25 units.
Running task = [T4] [5] [15] for 15 units.
Running task = [T5] [5] [20] for 20 units.
Running task = [T1] [4] [20] for 20 units.
Running task = [T2] [3] [25] for 25 units.
Running task = [T3] [3] [25] for 25 units.
Running task = [T7] [3] [30] for 30 units.
Running task = [T6] [1] [10] for 10 units.
```

四、schedule_rr.c

1、实现细节

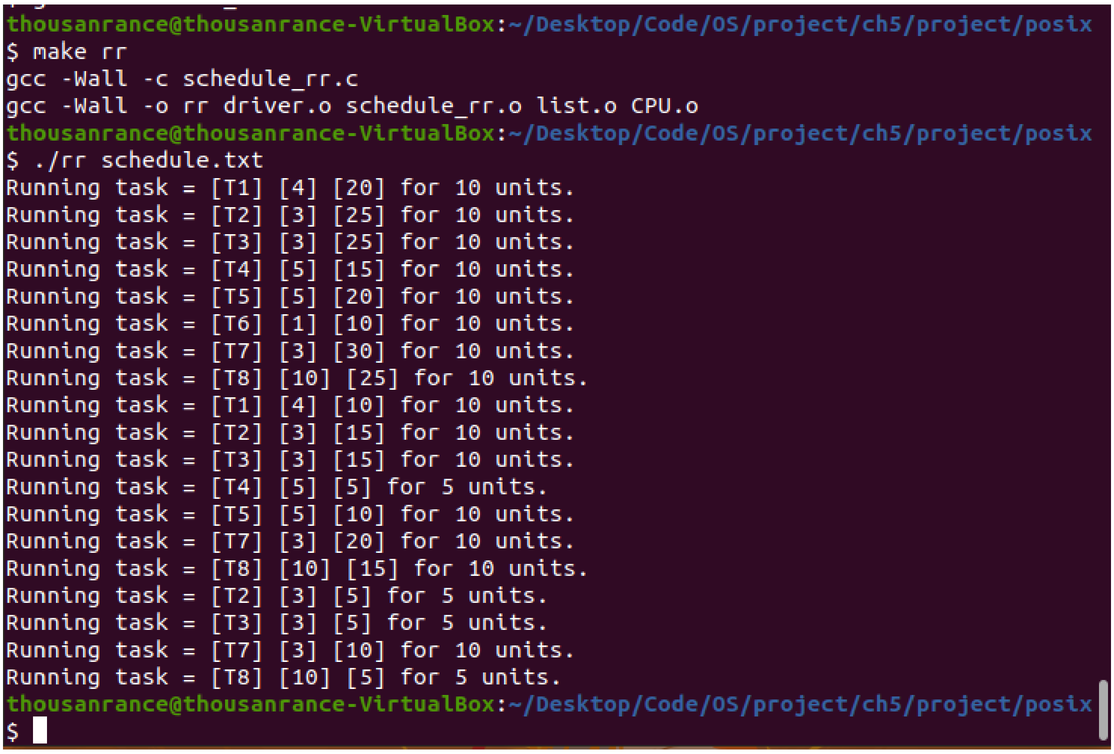
（1）add()

将新建的每个任务结点插入到表头。

（2）schedule()

在一个时间周期内，按照 fcfs 调度原则选择要调度的任务，如果该任务 burst 小于一个时间周期，则执行完该任务后删除该任务，直接进入下一个时间周期，选取下一个要调度的任务；如果该任务 burst 大于一个时间周期，则该任务只执行一个时间周期，一个时间周期结束后将该任务从任务列表尾部删除，将其 burst 减少一个时间周期，在将其加入到任务列表头部，即视为最后一个到达的任务。

2、运行结果



```
thousanrance@thousanrance-VirtualBox:~/Desktop/Code/OS/project/ch5/project/posix
$ make rr
gcc -Wall -c schedule_rr.c
gcc -Wall -o rr driver.o schedule_rr.o list.o CPU.o
thousanrance@thousanrance-VirtualBox:~/Desktop/Code/OS/project/ch5/project/posix
$ ./rr schedule.txt
Running task = [T1] [4] [20] for 10 units.
Running task = [T2] [3] [25] for 10 units.
Running task = [T3] [3] [25] for 10 units.
Running task = [T4] [5] [15] for 10 units.
Running task = [T5] [5] [20] for 10 units.
Running task = [T6] [1] [10] for 10 units.
Running task = [T7] [3] [30] for 10 units.
Running task = [T8] [10] [25] for 10 units.
Running task = [T1] [4] [10] for 10 units.
Running task = [T2] [3] [15] for 10 units.
Running task = [T3] [3] [15] for 10 units.
Running task = [T4] [5] [5] for 5 units.
Running task = [T5] [5] [10] for 10 units.
Running task = [T7] [3] [20] for 10 units.
Running task = [T8] [10] [15] for 10 units.
Running task = [T2] [3] [5] for 5 units.
Running task = [T3] [3] [5] for 5 units.
Running task = [T7] [3] [10] for 10 units.
Running task = [T8] [10] [5] for 5 units.
thousanrance@thousanrance-VirtualBox:~/Desktop/Code/OS/project/ch5/project/posix
$
```

五、schedule_priority_rr.c

1、实现细节

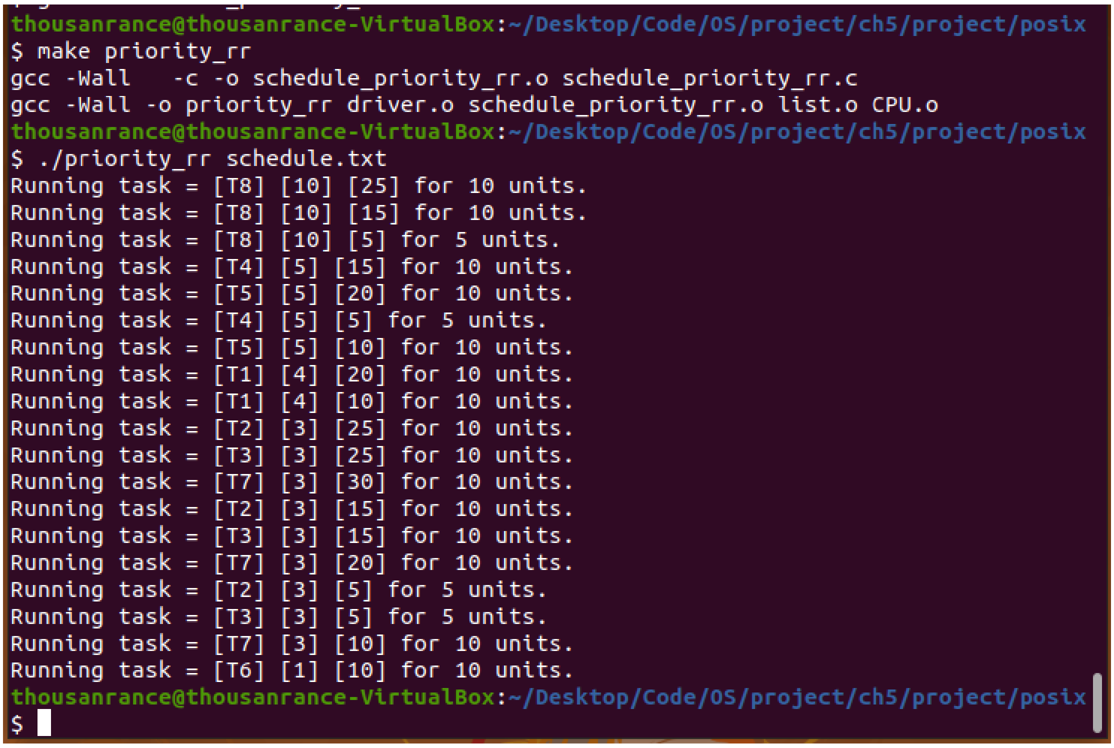
（1）add()

将新建的每个任务结点插入到表头。

（2）schedule()

在一个时间周期内，按照 priority 调度原则选择要调度的任务，如果该任务 burst 小于一个时间周期，则执行完该任务后删除该任务，直接进入下一个时间周期，选取下一个要调度的任务；如果该任务 burst 大于一个时间周期，则该任务只执行一个时间周期，一个时间周期结束后将该任务从任务列表删除，将其 burst 减少一个时间周期，再将其加入到任务列表头部，即视为最后一个到达的任务。

2、运行结果



```
thousanrance@thousanrance-VirtualBox:~/Desktop/Code/OS/project/ch5/project/posix
$ make priority_rr
gcc -Wall   -c -o schedule_priority_rr.o schedule_priority_rr.c
gcc -Wall -o priority_rr driver.o schedule_priority_rr.o list.o CPU.o
thousanrance@thousanrance-VirtualBox:~/Desktop/Code/OS/project/ch5/project/posix
$ ./priority_rr schedule.txt
Running task = [T8] [10] [25] for 10 units.
Running task = [T8] [10] [15] for 10 units.
Running task = [T8] [10] [5] for 5 units.
Running task = [T4] [5] [15] for 10 units.
Running task = [T5] [5] [20] for 10 units.
Running task = [T4] [5] [5] for 5 units.
Running task = [T5] [5] [10] for 10 units.
Running task = [T1] [4] [20] for 10 units.
Running task = [T1] [4] [10] for 10 units.
Running task = [T2] [3] [25] for 10 units.
Running task = [T3] [3] [25] for 10 units.
Running task = [T7] [3] [30] for 10 units.
Running task = [T2] [3] [15] for 10 units.
Running task = [T3] [3] [15] for 10 units.
Running task = [T7] [3] [20] for 10 units.
Running task = [T2] [3] [5] for 5 units.
Running task = [T3] [3] [5] for 5 units.
Running task = [T7] [3] [10] for 10 units.
Running task = [T6] [1] [10] for 10 units.
thousanrance@thousanrance-VirtualBox:~/Desktop/Code/OS/project/ch5/project/posix
$ 
```

（三）Further Challenges

1、Each task provided to the scheduler is assigned a unique task (tid). If a scheduler is running in a SMP environment where each CPU is separately running its own scheduler, there is a possible race condition on the variable that is used to assign task identifiers. Fix this race condition using an atomic integer.

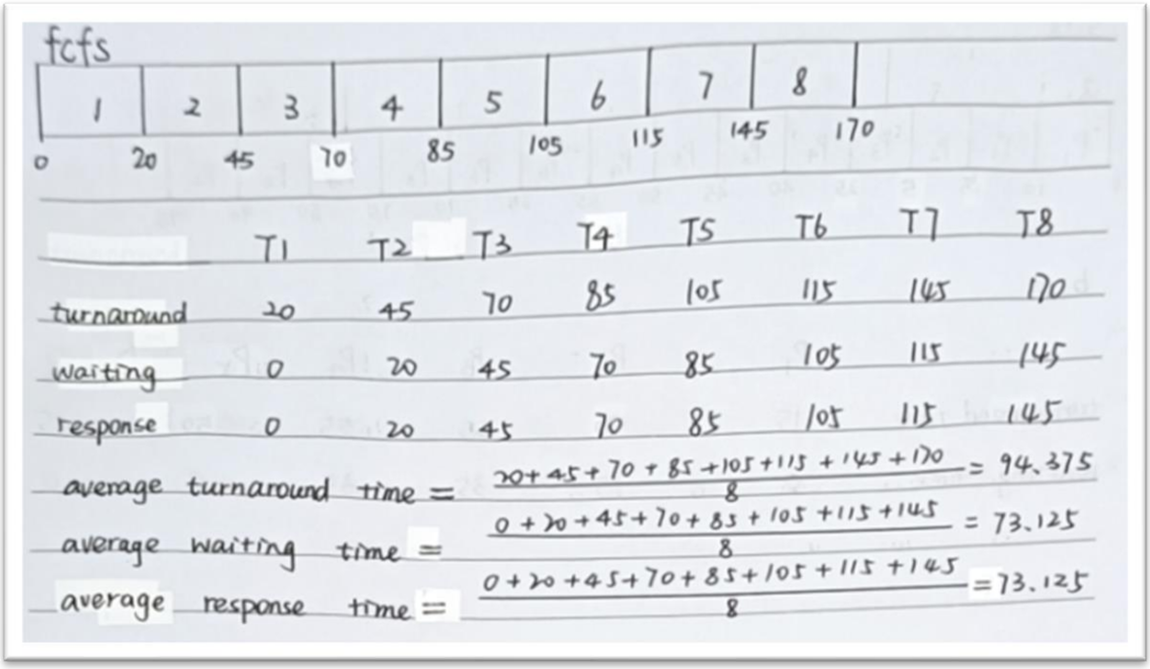
• Sol：

已在每个调度算法源文件的 schedule() 函数中通过 __sync_fetch_and_add(&tid_value)函数实现。

2、Calculate the average turnaround time, waiting time and response time for each of the scheduling algorithms.

•Sol:

1、FCFS



fcfs

1	2	3	4	5	6	7	8

0 20 45 70 85 105 115 145 170

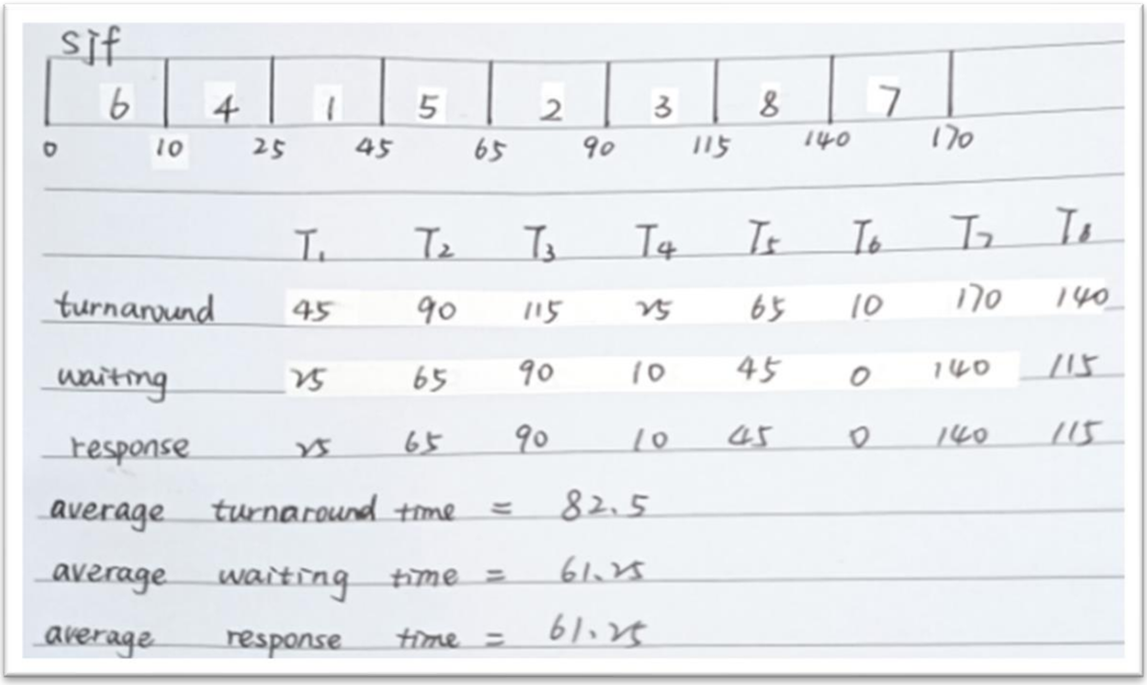
	T1	T2	T3	T4	T5	T6	T7	T8
turnaround	20	45	70	85	105	115	145	170
waiting	0	20	45	70	85	105	115	145
response	0	20	45	70	85	105	115	145

average turnaround time = $\frac{20+45+70+85+105+115+145+170}{8} = 94.375$

average waiting time = $\frac{0+20+45+70+85+105+115+145}{8} = 73.125$

average response time = $\frac{0+20+45+70+85+105+115+145}{8} = 73.125$

2、SJF



sjf

6	4	1	5	2	3	8	7

0 10 25 45 65 90 115 140 170

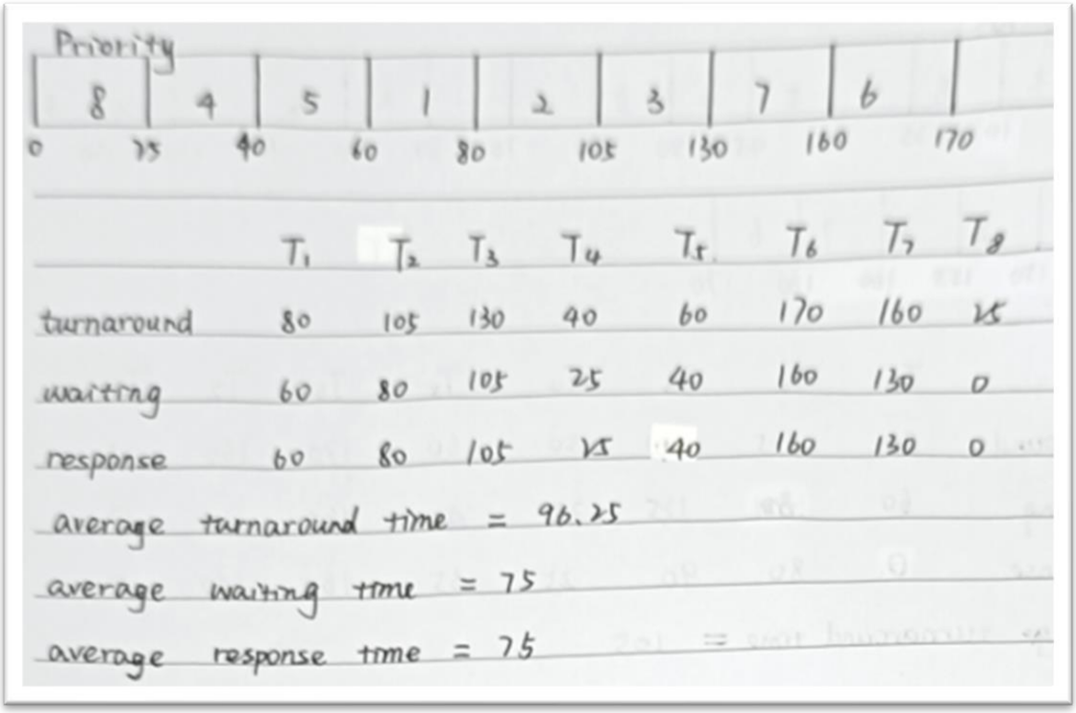
	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
turnaround	45	90	115	25	65	10	170	140
waiting	25	65	90	10	45	0	140	115
response	25	65	90	10	45	0	140	115

average turnaround time = 82.5

average waiting time = 61.25

average response time = 61.25

3、Priority



Priority

| 8 | 4 | 5 | 1 | 2 | 3 | 7 | 6 |

0 25 40 60 80 105 130 160 170

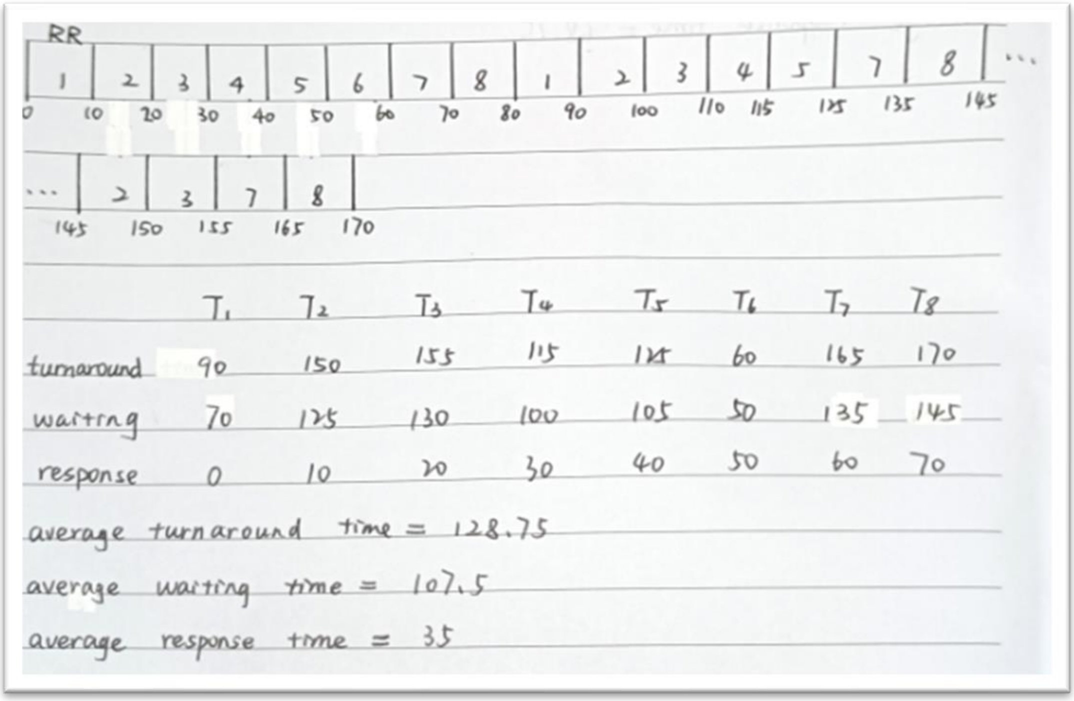
	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
turnaround	80	105	130	40	60	170	160	25
waiting	60	80	105	25	40	160	130	0
response	60	80	105	25	40	160	130	0

average turnaround time = 96.25

average waiting time = 75

average response time = 75

4、RR



RR

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 1 | 2 | 3 | 4 | 5 | 7 | 8 | ... |

0 10 20 30 40 50 60 70 80 90 100 110 115 125 135 145

... | 2 | 3 | 7 | 8 |

145 150 155 165 170

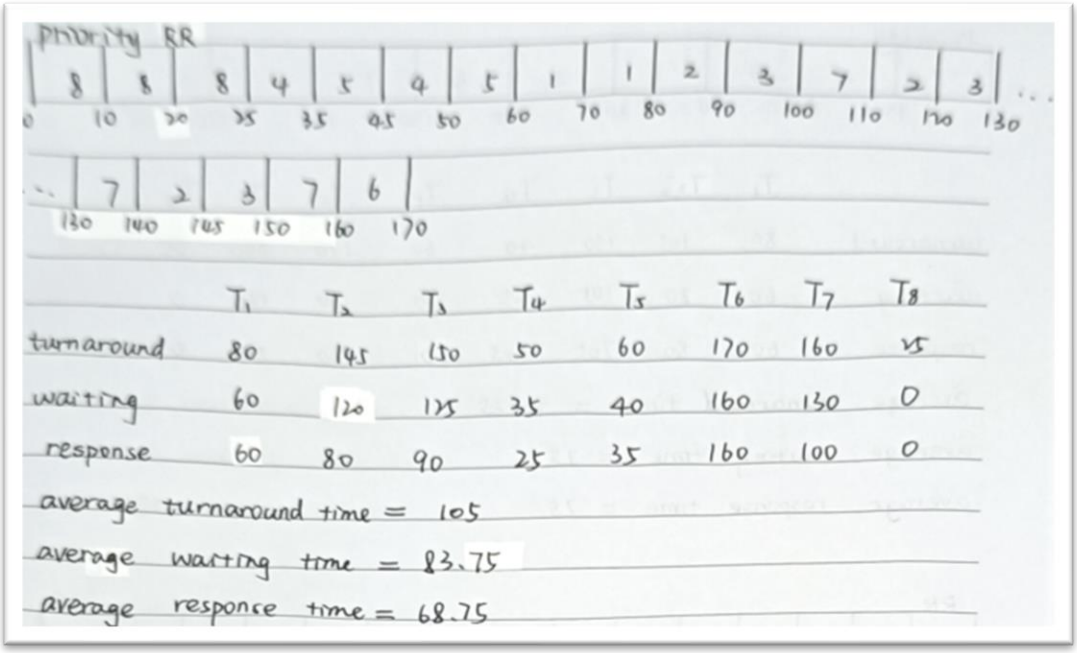
	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
turnaround	90	150	155	115	125	60	165	170
waiting	70	125	130	100	105	50	135	145
response	0	10	20	30	40	50	60	70

average turnaround time = 128.75

average waiting time = 107.5

average response time = 35

5、Priority RR



priority RR

```
| 8 | 8 | 8 | 4 | 5 | 4 | 5 | 1 | 1 | 2 | 3 | 7 | 2 | 3 | ...
0   10  20  25  35  45  50  60  70  80  90  100 110 120 130

... | 7 | 2 | 3 | 7 | 6 |
    130 140 145 150 160 170
```

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
turnaround	80	145	150	50	60	170	160	25
waiting	60	120	125	35	40	160	130	0
response	60	80	90	25	35	160	100	0

average turnaround time = 105

average waiting time = 83.75

average response time = 68.75