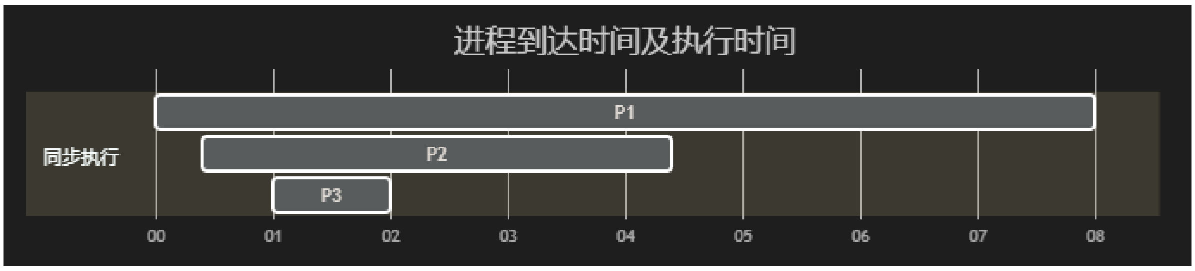
Chapter 6

6.2 Explain the difference between preemptive and nonpreemptive scheduling.

　抢占式调度允许在进程从运行状态转变为就绪状态和从等待状态转变为就绪状态时进行调度，而非抢占式调度只允许在进程从运行状态转变为等待状态和终止时进行调度。

6.3 Suppose that the following processes arrive for execution at the times indicated. Each process will run for the amount of time listed. In answering the questions, use nonpreemptive scheduling, and base all decisions on the information you have at the time the decision must be made.



Process	Arrival time	Burst time
P1	0.0	8
P2	0.4	4
P3	1.0	1

1. What is the average turnaround time for these processes with the FCFS scheduling algorithm?

2. What is the average turnaround time for these processes with the SJF scheduling algorithm?

3. The SJF algorithm is supposed to improve performance, but notice that we chose to run process *P*1 at time 0 because we did not know that two shorter processes would arrive soon. Compute what the average turnaround time will be if the CPU is left idle for the first 1 unit and then SJF scheduling is used. Remember that processes *P*1 and *P*2 are waiting during this idle time, so their waiting time may increase. This algorithm could be called future-knowledge scheduling.

1. `( 8 + (12 - 0.4) + (13 - 1)) / 3 = 10.53`
2. `( 8 + (9 - 1) + (13 – 0.4)) / 3 = 9.53`
3. `((2 - 1) + ( 6 – 0.4 ) + ( 14 - 0)) / 3 = 6.87`

6.6 Suppose that a scheduling algorithm (at the level of short-term CPU scheduling) favors those processes that have used the least processor time in the recent past. Why will this algorithm favor I/O-bound programs and yet not permanently starve CPU-bound programs?

因为I/O密集型程序的所需要的CPU事件较少，所以多优先执行，但是又由于这类程序有大量的I/O，所以在I/O密集型程序等待I/O的时候，可以执行CPU密集型程序，从而CPU密集型程序不会产生饥饿。

6.9 The traditional UNIX scheduler enforces an inverse relationship between priority numbers and priorities: the higher the number, the lower the priority. The scheduler recalculates process priorities once per second using the following function:

Priority = (recent CPU usage / 2) + base

where base = 60 and recent CPU usage refers to a value indicating how often a process has used the CPU since priorities were last recalculated.

Assume that recent CPU usage for process P1 is 40, for process P2 is 18, and for process P3 is 10. What will be the new priorities for these three processes when priorities are recalculated? Based on this information, does the traditional UNIX scheduler raise or lower the relative priority of a CPU-bound process?

P1 = 80

P2 = 69

P3 = 65

降低优先级

6.10 Why is it important for the scheduler to distinguish I/O-bound programs from CPU-bound programs?

I/O密集型程序使用大量的I/O而只使用少量的计算，而CPU密集型程序很少产生I/O请求，需要大量的计算。

将两者区分可以赋予I/O密集型程序更高的优先级，并且允许其在CPU密集型程序之前允许，从而更好的利用计算机资源。

6.13 In Chapter 5, we discussed possible race conditions on various kernel data structures. Most scheduling algorithms maintain a run queue, which lists processes eligible to run on a processor. On multicore systems, there are two general options: (1) each processing core has its own run queue, or (2) a single run queue is shared by all processing cores. What are the advantages and disadvantages of each of these approaches?

（1）的优点在于多个处理核之间不会竞争同一个进程并且能够更好的利用处理器的亲和性，（2）不具备这些优点，而（2）可以更好的进行负载平衡，但（1）需要进行周期性检查并进行推拉迁移才能负载平衡

6.16 Consider the following set of processes, with the length of the CPU burst given in milliseconds:

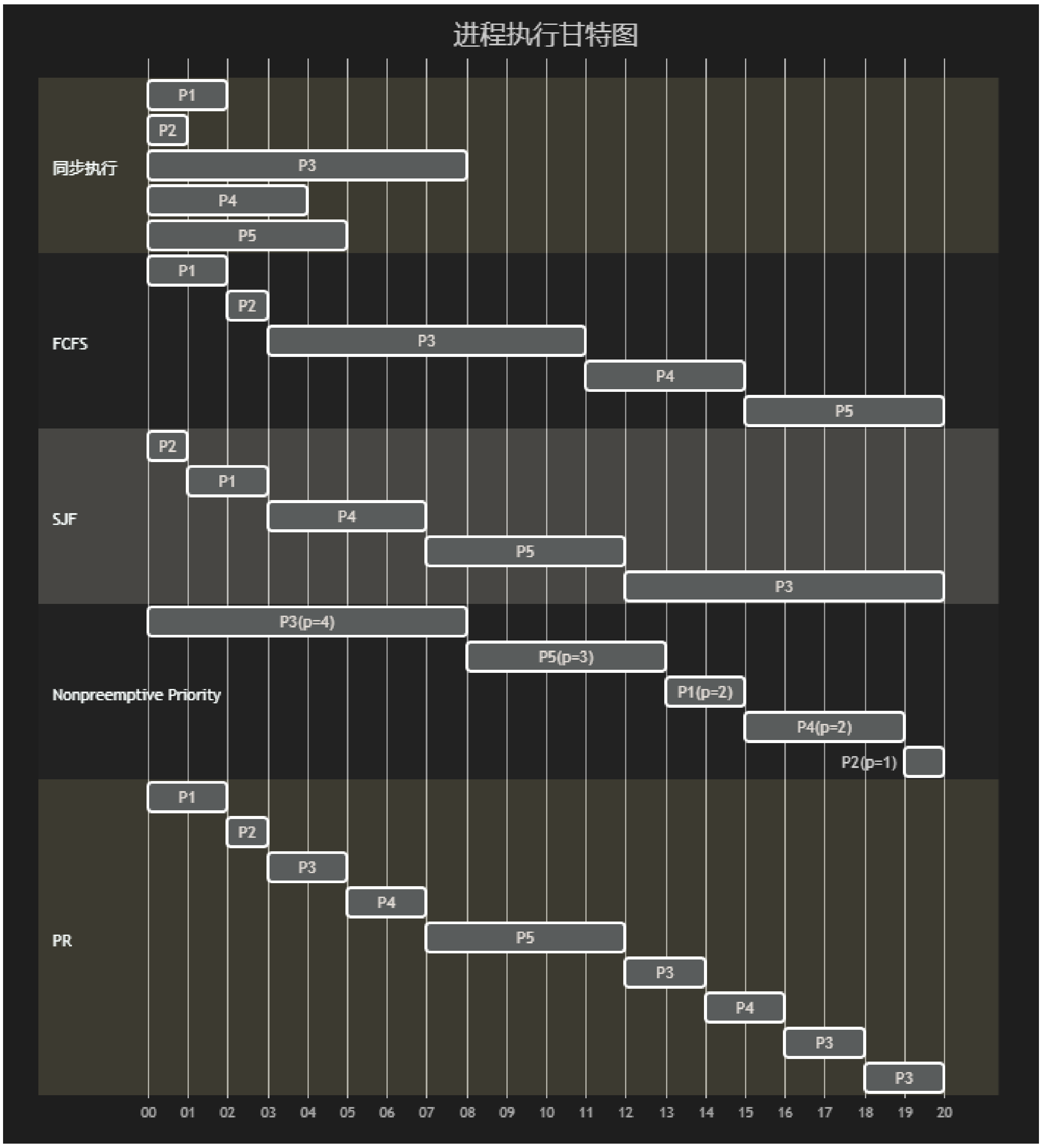
The processes are assumed to have arrived in the order *P*1, *P*2, *P*3, *P*4, *P*5, all at time 0.

a. Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: FCFS, SJF, nonpreemptive priority (a larger priority number implies a higher priority), and RR (quantum = 2).

b. What is the turnaround time of each process for each of the scheduling algorithms in part a?

c. What is the waiting time of each process for each of these schedul- ing algorithms?

d. Which of the algorithms results in the minimum average waiting time (over all processes)?



6.17 The following processes are being scheduled using a preemptive, round-robin scheduling algorithm. Each process is assigned a numerical priority, with a higher number indicating a higher relative priority. In addition to the processes listed below, the system also has an idle task (which consumes no CPU resources and is identified as Pidle). This task has priority 0 and is scheduled whenever the system has no other available processes to run. The length of a time quantum is 10 units. If a process is preempted by a higher-priority process, the preempted process is placed at the end of the queue.

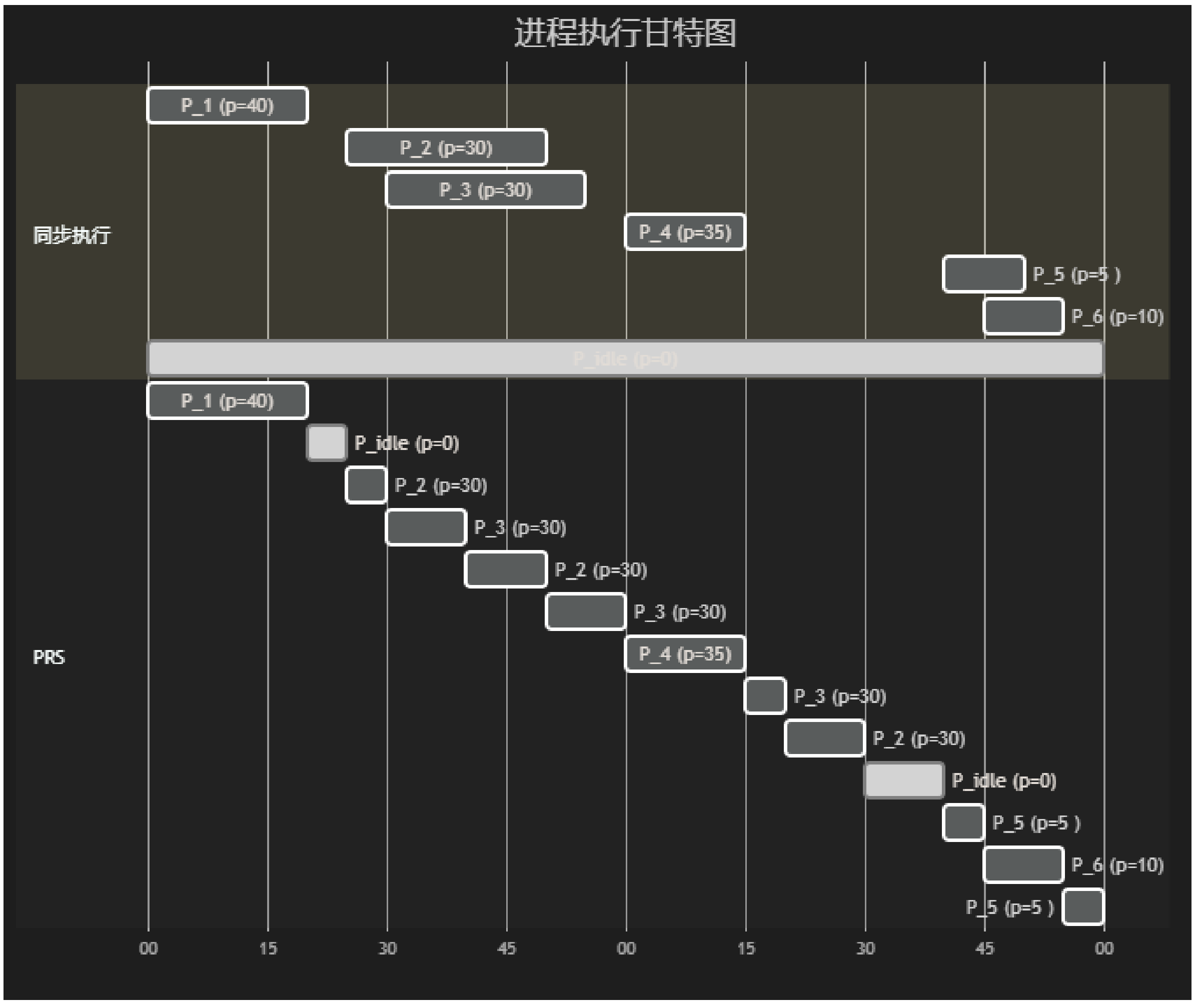
a. Show the scheduling order of the processes using a Gantt chart.

b. What is the turnaround time for each process?

c. What is the waiting time for each process?

d. What is the CPU utilization rate?

a.



b.

P1: `20-0 =20`, P2: `80-25 = 55`, P3: `90 - 30 = 60`, P4: `75-60 = 15`, P5: 120-100 = 20, P6: `115-105 = 10`

c.

P1: `0`, p2: `40`, P3: `35`, P4: `0`, P5: `10`, P6:`0`

d.

`105/120 = 87.5%`

6.19 Shortest job first and priority-based scheduling algorithms

1.First-come, first-served

2.Shortest job first

3.Round robin

4. Priority

(最短作业优先调度)Shortest job first 、 （优先级调度）priority可能会导致饥饿。

6.21 Consider a system running ten I/O-bound tasks and one CPU-bound task. Assume that the I/O-bound tasks issue an I/O operation once for every millisecond of CPU computing and that each I/O operation takes 10 milliseconds to complete. Also assume that the context-switching overhead is 0.1 millisecond and that all processes are long-running tasks. Describe the CPU utilization for a round-robin scheduler when:

a. The time quantum is 1 millisecond

b.The time quantum is 10 milliseconds

a.

`1/1.1=91%`

b.

`20/21.1*100=94.8%`

6.23 Consider a preemptive priority scheduling algorithm based on dynami- cally changing priorities. Larger priority numbers imply higher priority. When a process is waiting for the CPU (in the ready queue, but not running), its priority changes at a rate $ \alpha $. When it is running, its priority changes at a rate $ \beta $. All processes are given a priority of 0 when they enter the ready queue. The parameters $ \alpha $ and $ \beta $ can be set to give many different scheduling algorithms.

1. **What is the algorithm that results from $ \beta $ > $ \alpha $ > 0?**
2. **What is the algorithm that results from $ \alpha $< $ \beta $ < 0?**

1.FCFS

准备就绪队列中的所有进程都具有相同的初始优先级0。他们的优先级以相同的速度增加a（a>0）。因此，进程越早进入就绪队列，优先级就越高。一旦优先级最高的进程（与就绪队列中的其他进程相比先到先得）运行，其优先级的增加速度（b>a>0）高于准备就绪队列中这些进程的优先级。因此，没有其他过程会抢占它，直到运行完成。

2.LCFS

准备就绪队列中的所有进程都具有相同的初始优先级0。他们的优先级以相同的速度下降a（a<0）。因此，进程越早进入就绪队列，其优先级就越低。一旦具有最高优先级的进程（与就绪队列中的其他进程相比，最后来）运行，就绪队列中的其他进程都不会抢占它，因为它的优先级下降速度低于准备就绪队列中这些进程的优先级（a<b<0）。但任何即将到来的新进程都将抢占，因为新进程的优先级为0，这是尽可能高的优先级。

6.28 Assume that two tasks *A* and *B* are running on a Linux system. The nice values of *A* and *B* are −5 and +5, respectively. Using the CFS scheduler as a guide, describe how the respective values of vruntime vary between the two processes given each of the following scenarios:

- Both *A* and *B* are CPU-bound.
- *A* is I/O-bound, and *B* is CPU-bound.
- *A* is CPU-bound, and *B* is I/O-bound.

1. 由于A的优先级高于B，因此A的vruntime值较B更小（A减小，B增大）。如果A和B都是CPU密集型（也就是说，只要分配给它们，它们都会使用CPU），A的vruntime比B小，因此A运行的优先级更高。
2. A的vruntime比B的小得多，因为(1)由于优先级的差异，A的vruntime值较B更小；(2)A需要的CPU时间更少，A的vruntime值较B更小
3. 无法判断，因为(1)由于优先级的差异，A的vruntime值较B更小；(2)A需要的CPU时间更少，A的vruntime值较B更大，两个因素相互矛盾。