Student name: Federico Watkins
CSIS 3810: Operating Systems
Chapter: 6 CPU Scheduling
Assignment: 4

Exercises (provide theoretical answers):

(1) → 6.11	(2) → 6.13	(3) → 6.16	(4) → 6.17	(5) → 6.28

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

> It is important for the scheduler to distinguish *I/O (Input/Output) – bound programs* from *CPU (Central Processing Unit) – bound programs* because doing so will allow it to determine the most efficient way to use the computer's resources. The scheduler's efficiency depends on the optimum utilization of CPU. *I/O-bound programs* utilize small amount of CPU resources. They have the property of performing small amount of computations before performing IO. *CPU-bound programs* utilize large amount of CPU resources. They have the property of performing computations without performing any blocking IO operations. Thus, *I/O-bound programs* do not use all of their CPU quantum while *CPU-bound programs* do utilize their entire CPU quantum.
>
> Based on these properties of *I/O-bound* and *CPU-bound programs*, **scheduler should distinguish and give high priority to *I/O-bound programs*.** It should allow *I/O-bound programs* to execute before the *CPU-bound programs*.

2. 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?

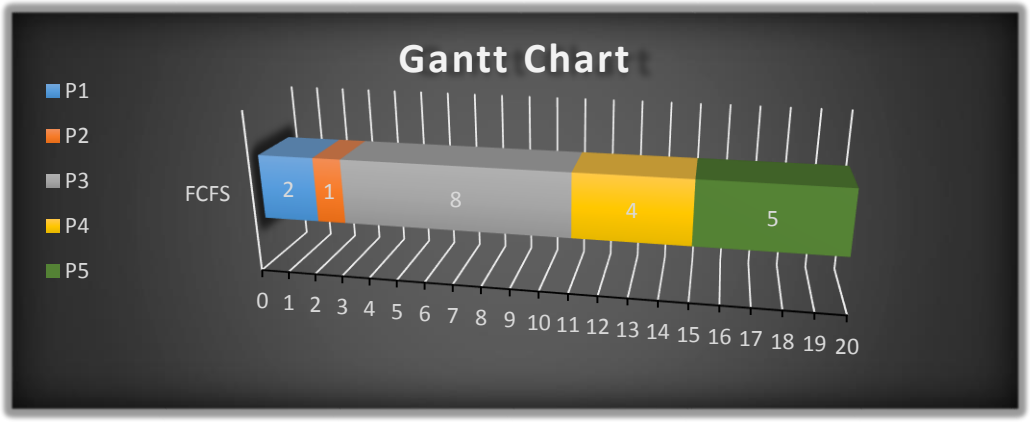
> With the approach of each processing core having its own run queue the main advantage is that there is no conflict over a single run queue when the scheduler is running synchronously on 2 or more processors. When a scheduling decision must be made for a processing core, the scheduler only need to look no further than its private run queue. A disadvantage of the approach of each processing core having its own run queue is that there must be some sort of load balancing between the different run queues. With the approach of a single run queue the advantage is that load balancing would likely not be an issue with a single run queue. A disadvantage is that it must be protected with locks to prevent a race condition. Hence, while a processing core may be available to run a thread, it must first acquire the lock to retrieve the thread from the single queue.

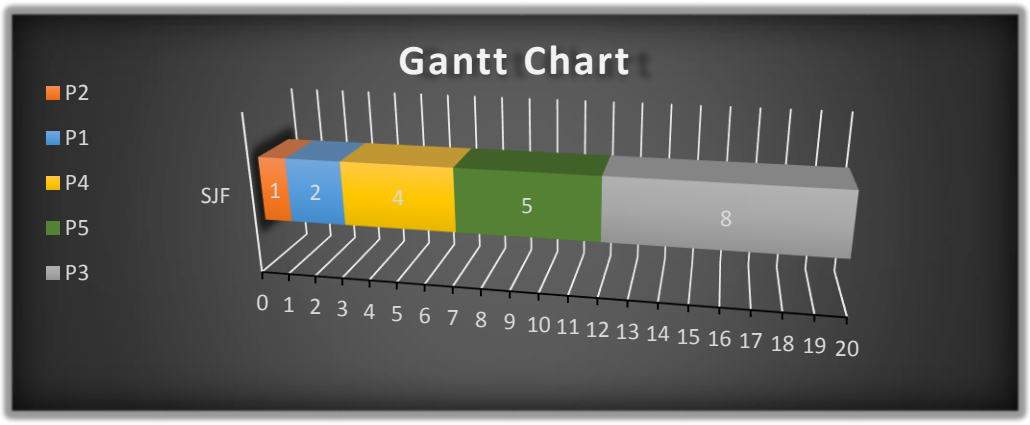
3. Consider the following set of processes, with the length of the CPU burst given in milliseconds:

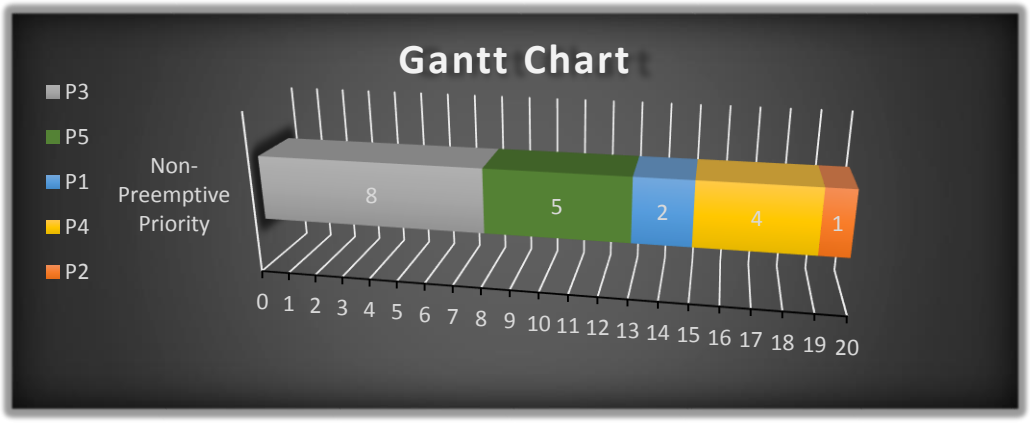
Process	Burst Time	Priority
P_1	2	2
P_2	1	1
P_3	8	4
P_4	4	2
P_5	5	3

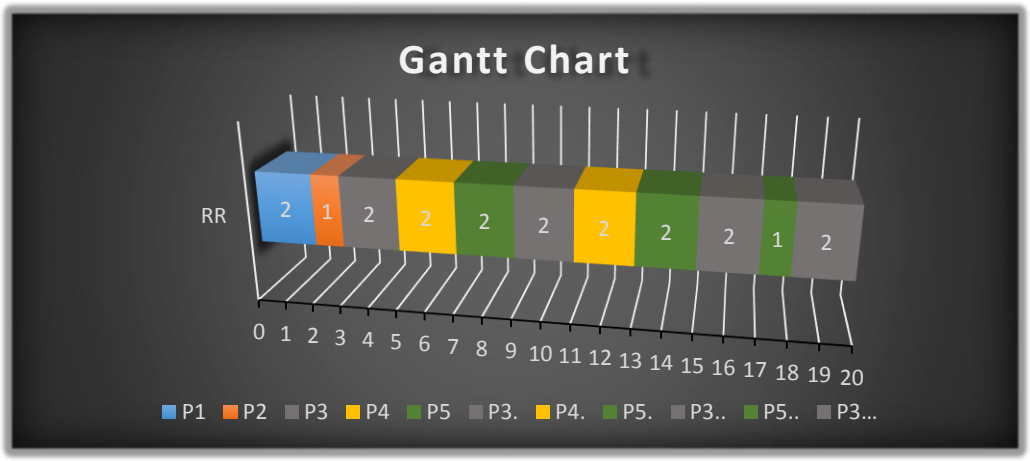
The processes are assumed to have arrived in the order P_1, P_2, P_3, P_4, P_5, all at time 0.

a. Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: FCFS, SJF, non-preemptive 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?

	FCFS	SJF	Priority	RR
P_1	2	3	15	2
P_2	3	1	20	3
P_3	11	20	8	20
P_4	15	7	19	13
P_5	20	12	13	18

c. What is the waiting time of each process for each of these scheduling algorithms?

	FCFS	SJF	Priority	RR
P_1	0	1	13	0
P_2	2	0	19	2
P_3	3	12	0	12
P_4	11	3	15	19
P_5	15	7	8	13

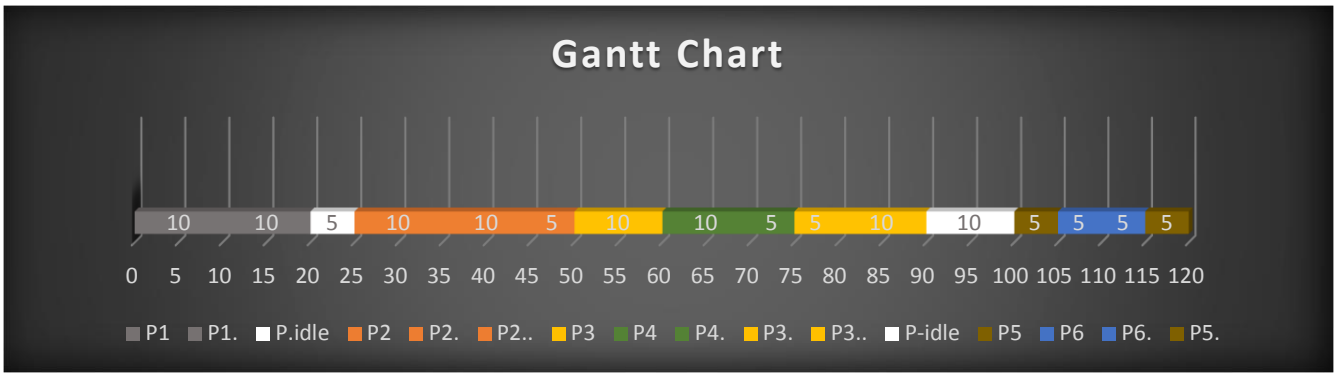
d. Which of the algorithms results in the minimum average waiting time (over all processes)?

The algorithm that results in the minimum average waiting time is "Shortest Job First" (SJF)

4. 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 P_{idle}). 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.

Thread	Priority	Burst	Arrival
*P*1	40	20	0
*P*2	30	25	25
*P*3	30	25	30
*P*4	35	15	60
*P*5	5	10	100
*P*6	10	10	105

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



b. What is the turnaround time for each process?

P_1 = 20-0 = 20
P_2 = 50-25 = 25
P_3 = 90-30 = 60
P_4 = 75-60 = 15
P_5 = 120-100 = 20
P_6 = 115-105 = 10

c. What is the waiting time for each process?

P_1 = 20-20 = 20
P_2 = 25-25 = 25
P_3 = 60-25 = 35
P_4 = 15-15 = 0
P_5 = 20-10 = 10
P_6 = 10-10 = 0

d. What is the CPU utilization rate?

$\frac{(120-15)}{120} \times 100 = 0.875 \times 100 = 87.5$

5. 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 *v*`runtime` vary between the two processes given each of the following scenarios:

- Both A and B are CPU-bound.

  V`runtime` will move slower for A than B because A has a higher priority than B. A will have a greater priority to run over B because, if A and B are CPU-bound, *v*`runtime` will usually be smaller for A than for B.

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

  A is I/O-bound requiring less CPU-time. Because of differences in priority *v* `runtime` will move slower for A than B.

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

  When task A is CPU-bound and task B is I/O-bound the *v*`runtime` of task A will be less than the value of *v*`runtime` of task B. That's due to the fact that CPU-bound processes consume its time period whenever it run on processor and it can be preempted by I/O-bound.

References:

1. Operating Systems Concepts (Abraham Silberschatz, Peter Baer Galvin, Greg Gagne) P. 261-263.
2. Operating Systems Concepts (Abraham Silberschatz, Peter Baer Galvin, Greg Gagne) P. 278-283.
3. Operating Systems Concepts (Abraham Silberschatz, Peter Baer Galvin, Greg Gagne) P. 266-277.
4. Operating Systems Concepts (Abraham Silberschatz, Peter Baer Galvin, Greg Gagne) P. 261-304.
5. Operating Systems Concepts (Abraham Silberschatz, Peter Baer Galvin, Greg Gagne) P. 290-304.

In this Chapter I learned that CPU scheduling is the basis of multi-programmed operating systems. Also that by switching the CPU among processes, the operating system can make the computer more productive. I was introduced to the basic CPU-scheduling concepts and presented with several CPU-scheduling algorithms. I explored the problem of selecting an algorithm for a particular system and examined the scheduling algorithm of a few operating systems. I learn to distinguish between actual process scheduling and thread scheduling. Went in detail over CPU-cycles, CPU-schedulers, Preemptive scheduling and Dispatcher. During the assignment I was able to demonstrate understanding of scheduling criteria, real-time CPU scheduling and algorithm evaluation.