CS372 Operating System
HW3

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

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

Process	Burst Time	Priority
P1	2	2
P2	1	1
P3	8	4
P4	4	2
P5	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, 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 scheduling algorithms?
d. Which of the algorithms results in the minimum average waiting time (over all processes)?

5.14 Consider a variant of the RR scheduling algorithm where the entries in the ready queue are pointers to the PCBs.

a. What would be the effect of putting two pointers to the same process in the ready queue?
a. What would be two major advantages and disadvantages of this scheme?
b. How would you modify the basic RR algorithm to achieve the same effect without the duplicate pointers?

6.9 The first known correct software solution to the critical-section problem for two processes was developed by Dekker. The two processes, **P**0 and **P**1, share the following variables:

```
boolean flag[2]; /* initially false */
int turn;
```

The structure of process **Pi** (i == 0 or 1) is shown in Figure 6.43; the other process is

Pj (j == 1 or 0). Prove that the algorithm satisfies all three requirements for the critical-section problem.

6.11 What is the meaning of the term *busy waiting*? What other kinds of waiting are there in an operating system? Can busy waiting be avoided altogether? Explain your answer.

6.21 Write an algorithm for a bounded-buffer monitor in which the buffers (portions) are embedded within the monitor itself.