操作系统作业

姓名，学号

1. The first known correct software solution to the critical-section problem for two processes was developed by Dekker. The two processes, P0 and P1, share the following variables:

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

   The structure of process Pi (i == 0 or 1) is shown in Figure 1; the other process is Pj (j == 1 or 0). Prove that the algorithm satisfies all three requirements for the critical-section problem.

```
do {
   flag[i] = true;

   while (flag[j]) {
      if (turn == j) {
         flag[i] = false;
         while (turn == j)
            ; /* do nothing */
         flag[i] = true;
      }
   }

      /* critical section */

   turn = j;
   flag[i] = false;

      /* remainder section */
} while (true);
```

Figure 1: The structure of process Pi for Question 1.

2. Consider the code example for allocating and releasing processes shown in Figure 2.
   a. Identify the race condition(s).
   b. Assume you have a mutex lock named `mutex` with the operations acquire() and release(). Indicate where the locking needs to be placed to prevent the race condition(s).

```
#define MAX_PROCESSES 255
int number_of_processes = 0;

/* the implementation of fork() calls this function */
int allocate_process() {
int new_pid;

   if (number_of_processes == MAX_PROCESSES)
      return -1;
   else {
      /* allocate necessary process resources */
      ++number_of_processes;

      return new_pid;
   }
}

/* the implementation of exit() calls this function */
void release_process() {
    /* release process resources */
    --number_of_processes;
}
```

Figure 2: Code for Question 2.

3. Servers can be designed to limit the number of open connections. For example, a server may wish to have only N socket connections at any point in time. As soon as N connections are made, the server will not accept another incoming connection until an existing connection is released. Use semaphores to limit the number of concurrent connections in the server.

4. Consider the traffic deadlock depicted in Figure 3.
   a. Show that the four necessary conditions for deadlock indeed hold in this example.
   b. State a simple rule for avoiding deadlocks in this system.

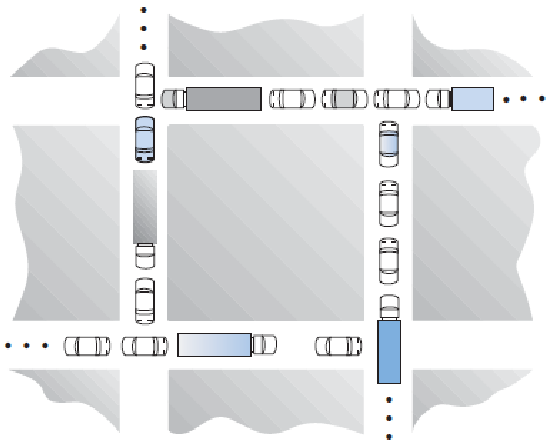

Figure 3. Traffic deadlock for Question 4.

5. Consider the deadlock situation that can occur in the diningphilosophers problem when the philosophers obtain the chopsticks one at a time. Discuss how the four necessary conditions for deadlock hold in this setting. Describe a deadlock-free solution, and discuss which necessary conditions are eliminated in your solution.

6. Discuss how the following pairs of scheduling criteria conflict in certain settings.
   a. CPU utilization and response time
   b. Average turnaround time and maximum waiting time
   c. I/O device utilization and CPU utilization

7. Consider the exponential average formula used to predict the length of the next CPU burst. What are the implications of assigning the following values to the parameters used by the algorithm?
   a. $\alpha = 0$ and $\tau_0 = 100$ milliseconds
   b. $\alpha = 0.99$ and $\tau_0 = 10$ milliseconds

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

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	3
P_4	1	4
P_5	5	2

The processes are assumed to have arrived in the order P1, P2, P3, P4, P5, 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 smaller priority number implies a higher priority), and RR (quantum = 1).
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)?

9. Which of the following scheduling algorithms could result in starvation?
   a. First-come, first-served
   b. Shortest job first
   c. Round robin
   d. Priority

10. 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 0.1millisecond and that all processes are long-running tasks. Describe is the CPU utilization for a round-robin scheduler when:
    a. The time quantum is 1 millisecond
    b. The time quantum is 10 milliseconds

11. Assume that two tasks Aand B are running on a Linux system. The nice values of Aand 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:
    a. Both Aand B are CPU-bound.
    b. A is I/O-bound, and B is CPU-bound.
    c. A is CPU-bound, and B is I/O-bound.

12. Give an example to illustrate under what circumstances rate-monotonic scheduling is inferior to earliest-deadline-first scheduling in meeting the deadlines associated with processes?