Operating System [Unit 2]

Assignment 2

Deadline: 2080-09-02

1. Define and explain
    a. Semaphore b. Mutex c. Sleep and wake up d. Test and Set Lock
    e. Monitors f. Message Passing
2. Differentiate between:
    a. Process and Program b. Process and Thread c. User and Kernel Thread
3. Explain Process models.
4. Discuss the process states.
5. Explain in detail about PCB and how it works.
6. Explain IPC and cooperating and independent processes.
7. Define critical section and race condition.
8. Describe mutual exclusion with busy waiting.
9. Discuss with codex about:
    a. Producer-consumer problem b. Sleeping Barber Problem
    c. Dining Philosopher Problem
10. How process synchronization is handled using semaphore? Explain with algorithms.
11. How thread-based execution minimizes the context switching problem of process-based execution? Explain the different multithreading model.
12. Compute FCFS, SJF, SRTF, RR(q=2) and RR(q=3)

Processes	Arrival Time	CPU Time
A	0.000	3
B	1.001	6
C	4.001	4
D	6.002	2

13. Five batch jobs A through E, arrive at a computer center at almost same time. They have estimated running times of 10, 8, 4, 2, and 6. Their priorities are 3, 5, 2, 4 and 1 respectively with 5 being the highest priority. For each of the scheduling algorithms determine all the computations as well as average turn-around time.
14. For each of the following transitions between the process's states, indicate whether the transition is possible. If it is possible, give an example of one thing that would cause it.
    a. Running -> Ready b. Running -> Blocked c. Blocked -> Running
15. When are threads better than processes. Define and describe the term "schedular" in OS.
16. How multithreading improves performance over single threaded solution.
17. Explain the concept of process hierarchy. How parent controls the child?
18. What are scheduling criteria? Differentiate between batch scheduling and interactive scheduling.
19. What is the purpose of system call in OS. Differentiate busy waiting and blocking.
20. Why semaphores are critical for programmers? What resources are used to create threads?