UG Mid Sem Exam
Semester : Vth
Date : 11/01/2021

Name : Subhojit Ghimire

Sch Id : 1912160

Branch : CSE - B

Subject : Operating System

Subject Code : CS303

Q. 1.

Ans

(a) → TRUE : A page table provides protection in the capability style, not in the access-list style.

(b) → FALSE : Binary semaphores are those that are used by no more than two processes, but one process at a time to enter the critical section.

~~(c) → FALSE : Deadlock~~ .

~~[illegible]~~

(c) → FALSE : Deadlock can still occur when there is only a single independent processor. This happens if a process issues an I/O command, then it is suspended, awaiting the result.

Q.2.

Ans

(a) → The pseudo-language for co-ordination of the file access :

```
monitor fileshare {

        enum (THINKING, WAITING, READING)
         State [N]

         Condition self [N]
         int total
         void open(int i) {

                   State [i] = WAITING
                   if (i + total >= N) {

                           self[i].wait().
                   }
                   state [i]  = READING
                   total += i

}
void close (int i) {

        state [i] = THINKING
        total = total - i

        for (int x = N - total - 1 ; x >= 0 ; --x) {
                  if (state [x] == WAITING) {

                          self [x]. signal().

                  }

        }

 }
 initialisation.code () {
          for (int i = 0 ; i < N ; ++i) {

                  State [i] = THINKING
          }
          total = 0
}   }
```

Q. 20

∴ Ans

(b)→ FCFS (first come first serve) Scheduling and SJF (Shortest Job first) Scheduling give the minimum Content Switches for the n processes in the READY queue.

Reason:

Non-preemptive processes or scheduling execute the process in RUNNING queue completely before moving on to the next process, unlike preemptive, which switches contex more frequently based on time quantum or to give priority to priority processes by abruptly sending the RUNNING process back to wait queue.

Therefore, non-preemptive scheduling algorithms give the minimum content switches, which include FCFS and SJF.

Q.2. Ans

(C) Round robin is preemptive and is based on the criteria of Time Quantum.

(i) → Reasons why Round Robin provides a fair CPU allocation:

(a) It follows Time Quantum, that means, a fixed time is allocated to each process in RUNNING queue.

(b) Context switching, hence, every process get equal time and their states get saved when they are preempted.

(c) Responsiveness. Since every processes get equal time, and the processes at last do not need to wait forever for the CPU to give some time to it, no processes are kept waiting forever, i.e., no starvation.

(ii) → We can use I/O Blocking to give certain critical processes a greater share of the CPU without changing the schedular. This is achieved, as we a process to use the CPU more times by inserting the same process multiple times in the Round Robin ready queue. This ensures the process is Scheduled more times than rest of the processes.