Operating System Design

Homework:

- Operating System Review 2
  Chapter 5 to 7

NAME:
STUDENT ID:
DATE:

Questions

1) Explain the difference between preemptive and nonpreemptive scheduling.

Preemptive scheduling:
This is a feature /or ability of the system /kernel to preemp or stop a currently scheduled task /process in a favor of a higher priority task /process.
This means the infinite loop in a user space cannot block the system processing.

Non preemptive
A non preemptive system allows the process to utilize the processor until they are ready to give up (cooperative multitasking).
In this case, a single process could easily hog the machine's CPU and in effect, bring the machine to a halt.

2) Which of the following scheduling algorithms could result in starvation?
   a. First-come, first-served

      No, every process is executed eventually (FCFS order)

   b. Shortest job first with preemption (shortest remaining time first)

      Yes, longer jobs may never execute if smaller ones keep coming.

   c. Shortest job first with no preemption

      Yes, //

   d. Round robin

      No, every process has a fair chance to execute.

   e. Priority

      Yes, low priority processes may never execute.

3) Explain the differences in how much the following scheduling algorithms discriminate in favor of short processes over long processes:
   a) FCFS - discriminates against short jobs since any short jobs arriving after a long jobs will have longer waiting time.

   b) RR - treats all jobs equally, regardless of short or long jobs.

   c) Multilevel feedback queues - this system moves long jobs into a lower priority queue, therefore it discriminates favorably toward short jobs.

4) The following pairs of scheduling algorithms have similarities when additional criterias are applied. Describe how one algorithms is actually a subset of the other algorithm.

   FCFS (with preemption) and RR scheduling algorithms:
   HINT: Consider quantum and real-time clock is used for preemption.
   The RR is a subset of FCFS with additional preemption capability base on the quantum time. If quantum is large enough, RR is identical to FCFS.

SJF is a priority scheduling where priority is the predicted next CPU burst time.

Shortest Job First and Priority scheduling algorithms:
HINT: Consider the length of the CPU burst time as a priority and preemption.

5) Describe the critical section problem that occurs on multi-processor or multi-core computer systems. What is a solution for the critical section problem?

Problem: In multi-processor or multi core system, there are processes or threads that share common variables, or table, or files. Each process has a code segment, call critical Section (CS), in which the shared data is accessed. The problem is to ensure that when one process is executing in its CS, no other process is allowed to execute in its CS.

Solutions: Solving this problem solves a locking / mutual exclusion problem, so it should meet 3 conditions:
- Mutual Exclusion: if process p_i is executing in its critical section, no other process is executing in its critical section.
- Progress: process wishing to enter their Critical Section can enter if no processes are currently in their Critical Section.
- Bounded Waiting: there is a bound or limit on the number of times other processes are allowed to enter their CS, after a process has requested to enter its CS.

6) Linux implement multiple locking mechanisms. Describe the circumstances under which they use spinlocks, mutexes, and semaphores.

Spin locks are used for multiprocessor systems where a thread can run in a busy loop (for a short period of time) rather than incurring the overhead of being put in a sleep queue

Mutexes are useful for locking resources.

Semaphores are the tools for synchronization when a resource must be held for a longer period of time, since spinning is inefficient for a long duration.

7) 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.

Busy waiting means that a process is waiting for a condition to be satisfied in a tight loop without relinquishing the processor. Alternatively, a process could wait by relinquishing the processor, and block on a condition & wait to be awakened at some appropriate time in the future. It can be avoided but incurs the overhead assoc. with putting a process to sleep & having to wake it up when the appropriate program state is reached.

8) A possible method for preventing deadlocks is to have a single, higher order resource that must be requested before any other resource. For example, if multiple threads attempt to access the synchronization objects A,B,C,D,and E, deadlock is possible (such synchronization objects maybe mutexes, semaphores, or condition variables). We can prevent the deadlock by adding a sixth object F. Whenever a thread wants to acquire the synchronization lock for any object A, B, C, D, and E, it must first acquire the lock for object F.This solution is known as containment: the locks for objects A, B, C, D, and E are contained within the lock for object F. Compare this scheme with the circular-wait scheme.

Both schemes will prevent the deadlock. But the Circular-Wait algo will be more efficient than the Containment algo. Since Circular-Wait allows multiple processors use their resources as long as there is no deadlock. But Containment only allows one process to use those resources at a time. For the implementation, the Containment is much easier than Circular-Waiting. For some processes that cannot predict how many resources it will use, implementing Circular-Waiting algo will be even harder.

9) Is it possible to have a deadlock involving only a single process? Explain your answer.

No. There is only one process and hence no condition about Mutual Exclusion.

10) Describe Priority Inversion and describe a technique that can be used to resolve Priority Inversion? Priority Inversion happens when a higher priority task waiting or a semaphore held by a lower priorty task. Before the lower priority task release the semaphore, it is interrupted by another task and the higher priority task has to wait.

Technique to resolve priority inversion: using Priority Inheritance Protocol → raise priority of all processes holding lock on resource needed by a higher priority process → return to normal priority after releasing the Critical Section.

11) Describe the methods that an Operating System can use to handle Deadlock condition in a process. Which method is generally used to handle Deadlock?

There are 3 ways in general to handle deadlocks:
- Deadlock prevention & avoidance - Do not allow the system to get into deadlock state.
- Deadlock detection & recovery - Abort a process or preempt some resources when deadlocks are detected
- If deadlock rarely happens (1,2 times a year), just ignore the problem altogether & simply reboot as nessesary than to incur the constant overhead & system penalties assoc. w/ deadlocks prevention or detection. This is the approach that both Windows & Unix take.