Sessional Assignment

Course Name: Operating System

Submitted By:

Abdul Razzaq (12938)

BS (SE-8) Section: A

Submitted To:

Sir Daud Khan

Dated: 12th July 2020

Department of Computer Science,
IQRA National University, Peshawar Pakistan

Operating System Concepts

Sessional Assignment Spring 2020

Marks : 20

1. Explain the necessary conditions that may lead to a deadlock situation. 2. What are the various methods for handling deadlocks?

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

3. Consider a system consisting of 4 resources of the same type that are shared by 3 processes, each of which needs at most 2 resources. Show that the system is deadlock free.

4. What is a resource allocation graph? How do you obtain a wait-for graph from it? Explain their uses.

5. Can a system detect that some of its processes are starving? If you answer "yes," explain how it can. If you answer "no," explain how the system can deal with the starvation problem.

6. On a disk with 1000 cylinders, number 0 to 999, compute the number of tracks the disk arm must move to satisfy all the requests in the disk queue. Assume the last request serviced was at track 345 and the head is moving toward track 0. The queue in FIFO order contains requests for the following tracks: 123, 847, 692, 475, 105, 376. Perform the computations for the following disk scheduling algorithms:

    - FCFS
    - SSTF

Q1

Answer:

Necessary Conditions:

There are four necessary conditions that may lead to deadlock:

- **Mutual Exclusion:** A minimum of one resource should be held during a non-sharable mode; if the other process requests this resource, then that process should await the resource to be free.

- **Hold and Wait:** A process should be at the same time holding a minimum of one resource and awaiting a minimum of one resource that's currently being held by another process.

- **No preemption:** Once a process is holding a resource then that resource can't be removed from that process till the process voluntarily releases it.

- **Circular Wait:** In circular wait a set of processes {P0, P1, P2, . . ., PN} should exist such that each P[i] is awaiting P[(i + 1) % (N + 1)]. (This condition implies the hold-and-wait condition; however it's easier to manage the conditions if the four are thought of individually.)

Methods for Handling Deadlocks:

Generally speaking there are 3 ways of handling deadlocks:

- **Deadlock prevention or avoidance:** Don't permit the system to urge into a deadlocked state.

- **Deadlock detection and recovery:** Abort a process or preempt some resources once deadlocks are detected.

- **Ignore the problem all together:** If deadlocks solely occur once a year approximately, it should be good to simply allow them to happen and revive as necessary than to incur the constant overhead and system performance penalties related to deadlock prevention or detection. This can be the approach that both Windows and UNIX take.

- In order to avoid deadlocks, the system should have extra info concerning all processes. Above all, the system should understand what resources a process can or could request within the future. (Ranging from a straightforward worst-case maximum to an entire resource request and release plan for every process, looking on the particular algorithm)

- Deadlock detection is fairly simple, however deadlock recovery needs either aborting processes or preempting resources, neither of that is an attractive alternative.

- If deadlocks are not prevented and also not detected, then once a deadlock happens the system can gradually slow down, as more and more processes become stuck awaiting resources currently held by the deadlock and by different waiting processes. Sadly this holdup will be indistinguishable from a general system holdup once a real-time process has significant computing needs.

Q2

Answer:

A deadlock situation can arise if the following four conditions occur in a system.

- Mutual Exclusion.
- Hold and Wait.
- No Preemption.
- Circular-wait.

It is unfeasible to have circular wait with just one process, therefore failing a necessary condition for Circular wait. There's no second process to create a circle with the primary one. So it isn't possible to have a deadlock involving only one single process. The deadlock implies a circular "hold-and-wait" condition between 2 or more processes, thus "one" process cannot hold a resource, yet be awaiting another resource that it's holding.

Q3

Answer:

Suppose the system is deadlocked. This means that every process is holding one resource and is awaiting an additional. Since there are 3 processes and 4 resources, one process should be able to get 2 resources. This process needs no additional resources and thus it'll return its resources once done.

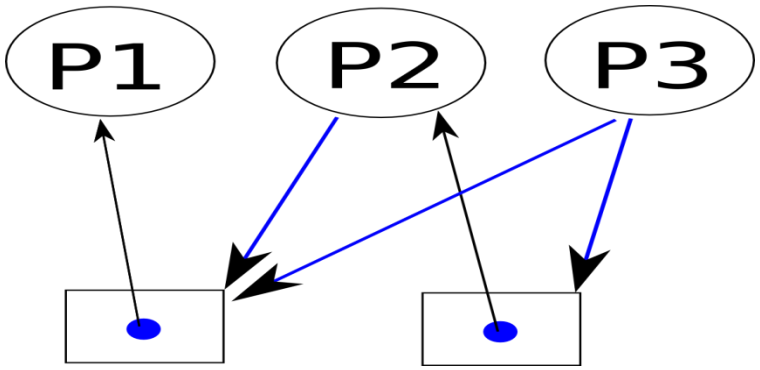
Q4

Answer:

A resource allocation graph tracks that resource is held by which process and which process is awaiting a resource of a selected type. It's very powerful and straightforward tool to illustrate how interacting processes will deadlock. If a process is using a resource, arrow is drawn from the resource node to the process node. Arrow is drawn from the process node to the resource node, if a process is requesting a resource.

If there's a cycle within the Resource Allocation Graph and every resource within the cycle provides just one instance, then the processes can deadlock. for instance, if process one holds resource A, process two holds resource B and process one is awaiting B and process two is awaiting A, then process one and process two are deadlocked.

Another example, that shows Processes one and 2 getting resources one and 2 whereas process three is waiting to acquire each resources. In this example there's no deadlock as a result of there's no circular dependency.



Q5

Answer:

Starvation: A process should wait on the far side an affordable amount of time perhaps indefinitely before receiving a requested resource. A way of detecting starvation would be to 1st determine an amount of time T that's considered unreasonable.

A timer is started, when a process requests a resource. If the time period exceeds T, then the process is taken into account to be starved.

One strategy for managing starvation would be to adopt a policy wherever resources are allotted solely to the process that has been waiting the longest. For instance, if process Pa has been waiting longer for resource X than process pb , the request from process pb would be delayed till process Pa's request has been completed.

Another strategy would be less strict than what was simply mentioned. During this situation, a resource can be granted to a process that has waited but another process, providing that the other process isn't starving. However, if another process is taken into account to be starving, its request would be completed 1st.

Q6

Answer:

FIFO

The FIFO schedule is 345,123,874,692,475,105 and 376.

Total head movement = (345-123) + (874-123) + (874-692) + (692-475) + (475-105) + (376-105) =2013

SSTF

The SSTF schedule is 345,376,475,692,874,123 and 105.

Total head movement = (376-345) + (475-376) + (692-475) + (847-692) + (874-123) +(123-105) = 1298