Date :

Name : Aakash A. Joshi
Roll no: 0077
Branch : Computer Batch : T4
Subject : System Programming & Operating Systems

Topic : Assignment 5 (Theory)

Date :

Questions:

1. Define deadlock. Explain the condition under which deadlock occurs.

2. Explain dining philopher's problem with example.

Answers:

1.

→ Deadlock is the problem of multiprogrammed system. Deadlock can be defined as the permanent blocking of a set of processes that either compete for system resources.

Deadlock can occur on shareable resources such as files, printers, database, disks, memory, GPU cycles, etc.

A process is in a deadlock state if it was waiting for particular event that will not occur.

There are four conditions that are necessary for deadlock:

a. Mutual Enclusion.
b. Hold & wait.
c. No preemption.
d. Circular Wait.

a. Mutual Enclusion:

A resource may be acquired exclusively by only one process at a time.

Date :

b. Hold & Wait:

Processes currently holding resources that were granted earlier can request new resources.

c. No preemption:

Once a process has obtained a resource the system cannot remove it from the process control until the process has finished using the resources.

d. Circular Wait:

A circular chain of hold & wait condition exists in the system.

All four of these conditions must be present for a resource deadlock to occur.

2.

→ Dining philosopher's problem is a classic synchronization & concurrency problem. That illustrates the challenges of ensuring that multiple processes can access shared resources without conflicts or deadlocks.

Example:

Imagine five philosophers sitting at a round table. They think and eat.

To eat, a philosopher needs two chopsticks, one for each hand.

Date :

The challenge is to come up with rules that allow the philosophers to eat without causing conflicts or getting stuck.

The rules are:

a. Each philosopher must pick up both the chopsticks on their left and right to each.

b. After eating, they put down both chopsticks for others to use.

c. Philosophers cannot eat together if they share a chopstick.

The problem is to find a way for the philosophers to take turns eating while avoiding deadlocks & ensuring everyone gets a chance to eat.

This illustrates the need for proper coordination in multithreaded or multiprocess applications.

11/11/23