Banker's Algorithm in Operating System (OS)

Banker’s Algorithm is a deadlock avoidance algorithm . It is also used for deadlock detection. This algorithm tells that if any system can go into a deadlock or not by analyzing the currently allocated resources and the resources required by it in the future. There are various data structures which are used to implement this algorithm.

Overview

Bankers algorithm in Operating System is used to avoid deadlock and for resource allocation safely to each process in the system.

Bankers algorithm in OS is a combination of two main algorithms: safety algorithm (to check whether the system is in a safe state or not) and resource request algorithm (to check how the system behaves when a process makes a resource request).

What is the Bankers Algorithm in OS?

A process in OS can request a resource, can use the resource, and can also release it. There comes a situation of deadlock in OS in which a set of processes is blocked because it is holding a resource and also requires some other resources at the same time that are being acquired by other processes. So to avoid such a situation of deadlock, we have the Bankers algorithm in Operating System.

Bankers algorithm in OS is a deadlock avoidance algorithm and it is also used to safely allocate the resources to each process within the system.

Each process within the system must provide all the important necessary details to the operating system like upcoming processes, requests for the resources, delays, etc.

Based on these details, OS decides whether to execute the process or keep it in the waiting state to avoid deadlocks in the system. Thus, Bankers algorithm is sometimes also known as the Deadlock Detection Algorithm.

Data Structures used to implement Banker's Algorithm

- **Available**: It is a 1-D array that tells the number of each resource type (instance of resource type) currently available. Example: Available[R1]= A, means that there are A instances of R1 resources are currently available.
- **Max**: It is a 2-D array that tells the maximum number of each resource type required by a process for successful execution. Example: Max[P1][R1] = A, specifies that the process P1 needs a maximum of A instances of resource R1 for complete execution.
- **Allocation**: It is a 2-D array that tells the number of types of each resource type that has been allocated to the process. Example: Allocation[P1][R1] = A, means that A instances of resource type R1 have been allocated to the process P1.
- **Need:** It is a 2-D array that tells the number of remaining instances of each resource type required for execution. Example: Need[P1][R1]= A tells that A instances of R1 resource type are required for the execution of process P1.
- Need[i][j]= Max[i][j] - Allocation[i][j], where i corresponds any process P(i) and j corresponds to any resouce type R(j)

Advantages

Following are the essential characteristics of the Banker's algorithm:

- It contains various resources that meet the requirements of each process.
- Each process should provide information to the operating system for upcoming resource requests, the number of resources, and how long the resources will be held.
- It helps the operating system manage and control process requests for each type of resource in the computer system.
- The algorithm has a Max resource attribute that represents indicates each process can hold the maximum number of resources in a system.

Disadvantages

- It requires a fixed number of processes, and no additional processes can be started in the system while executing the process.
- The algorithm does no longer allows the processes to exchange its maximum needs while processing its tasks.
- Each process has to know and state their maximum resource requirement in advance for the system.

- The number of resource requests can be granted in a finite time, but the time limit for allocating the resources is one year.