Abstract

Deadlock can occur wherever multiple processes interact. System deadlock is a serious problem in a multiprogramming environment. The approaches to this problem can be divided into three categories: (1) prevention, (2) detection and recovery, and (3) avoidance. This paper proposes a variation of the first approach, partially applying ideas developed in the second and third approaches. This is an approach that is especially effective in process control computer systems, in which the application programs are usually fixed once designed. Using four predetermined application program parameters obtained in the program development stage, a directed graph model and a 'restriction' matrix model are introduced representing
the usage of common resources. Conditions sufficient for system deadlock prevention are presented along with algorithms for checking to see that the models meet these conditions. By using this approach, if a deadlock possibility is detected the causes can also be detected. The deadlock can thus be prevented during the program development stage. As the algorithms are not used in the real-time mode, there is no negative effect on the responsiveness of the system. A higher utilization rate of common resources is also ensured because the usage of resources is restricted only when the possibility of a deadlock is detected.

References

- Operating system concepts, Greg Gagne, Peter B. Galvin.
- J. W. Murphy, Resource allocation with interlock detection in a multitasking system.
- Dijkstra N. W. and Scholten C. S., Termination detection for diffusing computations, Inf.
- Murata, T., "Petri nets: properties, analysis, and applications".

Index Terms

Computer Science
Computer Architecture

Keywords

Deadlock Prevention Recovery Detection Avoidance