A Software Aided Approach in Avoiding Deadlock Problems in Operating Systems Using Banker’s Algorithm

Okojie Cyriacus O.1, Daramola Olaife2, Onyegu Osokam S.3, Aliyu Nuhu4 and Aromolaran Damilola5

1Department of Mathematics/Computer Science, University of Port Harcourt, Nigeria
2Department of Computer Science, Federal University of Technology, Akure, Nigeria
3Department of Mechanical Engineering, University of Port Harcourt, Nigeria
4Department of Electrical and Computer Engineering, Ahmadu Bello University, Nigeria
5Department of Computer Science/Education, BOWEN University, Nigeria

Corresponding Author E-Mail/Phonl: profosom@yahoo.com/+2347034599022

Received 2 August 2012; Accepted 7 September 2012

Abstract

When two computer programs sharing the same resources are preventing each other from accessing the resources resulting in both the programs ceasing to function, this process is said to be a deadlock. In order to prevent deadlock in systems, there should be another system that monitors/manages the processes. The implementation of banker’s algorithm can be seen as a perfect measure in solving this deadlock problem. This implementation was done using developed software (Visual Basic 6.0) by exploring the Dijkstra’s banker’s algorithm for deadlock prevention. This is because of its embedded features which makes it readily and easy to handle processes that involve resource.

Keywords: Bankers algorithm, Operating Systems. Resources, System Design and Analysis

Introduction

Deadlock is the situation in which there exists a set of concurrent processes with each process in the set awaiting an event that can be cause only by another process in the set. Deadlock is a state in which two or more processes are stuck in a circular wait state, a deadlock processes are waiting for resources held by other processes because most systems are non-preemptive (that is will not take resources held by a process away from it), and employ a hold and wait method for
dealing with system resources (that is, once a process gets a certain resources, it will not give it up voluntarily). Deadlock is a dangerous state that will cause poor system performance.

The banker’s algorithm is a resource allocation and deadlock avoidance algorithm developed by Edsger Dijkstra that test for safety by simulating the allocation of pre-determined maximum possible amounts of all resource and then makes a safe state check to test for possible deadlock conditions for all other pending activities before deciding whether allocation should be allowed to continue.

The algorithm was developed in the design process for the operating system and originally described in Dutch and is run by the operating system. The algorithm prevents deadlock by denying or postponing the request if it determines that accepting the request could put the system in an unsafe state (one where deadlock could occur). when a new process enters a system, it must declare the maximum number of instances of each resource type that may not exceed the total number of resources in the system.

For the banker’s algorithm to work it needs to know three things; How much of each resource each process could possible request; How much of each resource each process is currently holding; How much of each resource the system currently has available.

Resources may be allocated only if it satisfies the following conditions. Request <= max, else set error condition as process has crossed maximum claim made by it; Request <= available, else process wait until resources are available.

The earliest computer operating system ran only one program at a time; all of the resources of the system were available to this one program (Lane and Mooney, 1988). Programs were required to specify in advance what resources they need in order to avoid conflict with other programs running at the same time. Eventually, some operating system offered dynamic allocation of resources. Programs could request further allocation of resources after they have begun running; this often led to the problem of deadlock. If the necessary conditions for a deadlock are in place, it is still possible to avoid deadlock by being careful when resources are allocated. Perhaps the most famous deadlock avoidance algorithm, due to Dijkstra is the Banker’s algorithm, so named because the process is analogous to that used by a banker in deciding if a loan can be safely made.
The banker’s algorithm is thus to consider each request as it occurs, and see if granting it leads to a safe state. If it does, the request is granted, otherwise, it postponed until later. Haberman (1969) has shown that executing of the algorithm has complexity proportional to $N^2$ where $N$ is the number of processes and since the algorithm is executed each time a resource request occurs, the overhead is significant. Lawley and others (1998) carried out the application and evaluation of banker’s algorithm for deadlock-free buffer space allocation in flexible manufacturing systems. Ho (2004) carried out a work on decomposition approach of banker’s algorithm in design and concurrency analysis problems to avoid deadlock in process design.

The aim of this project is to study the problems of deadlock as well as banker’s algorithm and also to highlight the application of the algorithm in solving and preventing deadlock problems in a centralized database system thus helping resource managers in the cause of managing their resource.

### Conditions for Deadlock

- Mutual exclusion: resources cannot be shared.
- Hold and wait: processes request resources incrementally, and hold on to what they’ve got.
- No preemption: resources cannot be taken away from the process that has them
- Circular wait: circular chain of waiting, in which each process is waiting for a resource held by the next process in the chain.

### Strategies for Dealing with Deadlock

- Ignore the problem altogether i.e. ostrich algorithm it may occur very infrequently, cost of detection/prevention may not be worth it.
- detection and recovery
- avoidance by careful resource allocation
- Prevention by structurally negating one of the four necessary conditions.
Difference from avoidance is that here, the system itself is built in such a way that there are no deadlocks. Make sure at least one of the 4 deadlock conditions is never satisfied. This may however be even more conservative than deadlock avoidance strategy.

- Attacking Mutex condition
- Attacking preemption
- Attacking hold and wait condition
- Attacking circular wait

Resources numbered 1 ... n. Resources can be requested only in increasing order. That is, you cannot request a resource whose no is less than any you may be holding.

In general, there are four strategies of dealing with deadlock problem:

1. The Ostrich Approach Just ignore the deadlock problem altogether.
2. Deadlock Detection and Recovery Detect deadlock and, when it occurs, take steps to recover.
3. Deadlock Avoidance Avoid deadlock by careful resource scheduling.
4. Deadlock Prevention Prevent deadlock by resource scheduling so as to negate at least one of the four conditions.

Process Analysis

The concept of deadlock and banker’s algorithm has been dealt with extensively in this research project. The importance of this project is to foster means of avoiding deadlock in operating system by using the banker’s algorithm which is just mathematical analysis of systems. However, there exist many areas where deadlock can occur, the major one and where this project will consider is the resource allocation within a computer processing system. The software will be able to take numerical quantities as input to the program, allocate resources based on demand. It will also detect when there is a deadlock and suggest a corrective measure (Silberschatz and Galvin, 1994).

System Design Methodology
The design methodology that is adopted for this project is the top down design methodology and the system will be designed in modules of separate functions which will be called together and coordinated by a main program.

System Specification

Input specification: the input specification for this system is limited to the numeric data which represent the number of total resources, the maximum demand and current resources needed by the processor.

Output specification: The output to this program will also be in form of numeric values. These values show the allocated resource, available resources and claimed resources.

Hardware Specification:

The efficiency and functionality of this software will depend on the system used. The system should have a minimum of; Pentium processor and above, CD ROM Drive(for installation and setup), 120 Gigabyte hard disk drive, 256 RAM memory, Good VGA card.

Safe And Unsafe State of Banker’s Algorithm

A safe state is considered safe if it is possible for all processes to finish executing (terminate). Since the system cannot know when a process will terminate or how many resources it will have requested by then, the system assumes that all processes will eventually attempt to acquire their stated maximum resources and terminate soon afterward. This is a reasonable assumption in most cases since the system is not particularly concerned with how long each process runs. Also if a process terminates without acquiring its maximum resources, it only makes it easier on the system.

Application of Banker’s Algorithm on Deadlock Problems

The banker’s algorithm seeks to prevent deadlock by becoming involved in the granting or denying of system resources. Whenever a process request resources, the allocator has the opportunity to block the process. Whenever a process releases resources, the allocator also has

http://www.projectsolutiondomain.com
the opportunity to unblock some waiting process. The allocator is in a sense a scheduler of resource allocation. Each time a process makes a request for resource, the allocator will determine whether or not a deadlock state could possible arise in the future if the request is granted. This determination is made by pretending to grant the request and then looking at the resulting post-granted request system state. After granting a resource request, there will be an amount of that resource left free in the system.

Further, there may be other processes in the system and each of these other processes state the maximum amount of all system resources they need to terminate up front.

Processes cannot be allocated more resources than the total available. In fact processes will typically request at most a number of resources far less than the total. The total is chosen so that there are enough resources to ensure maximum parallel resource utilization.

In some cases, temporarily blocking a process instead of granting a request will avoid deadlock. However, additional information is needed to determine whether a process should be blocked.

The banker’s algorithm is more or less summarized such that it follows a definite pattern when being applied. The algorithm is as below.

- Process need = maximum quantity of each resources that process requires during its lifetime
- Resource manager(banker) = needs prior knowledge of all process claims
- Process request one resource at a time
- Current process request is allowed if the request plus current usage of the processes is less than the need after allocating the request there exists a sequence in which all processes can run to completion even if they demand their full claim - SAFE STATE
- User must guarantee to release resources in finite time
- Resource manager guarantee to allocate resources in finite time process.

Bankers Algorithm Pseudocode

```javascript
function bankers_algorithm(set of processes P, currently available resources A) {
    while (P not empty) {
        boolean found = false
        for each (process p in P) {
            // Allocation logic here...
        }
    }
}
```
System Implementation

System implementation can be described as the realization of an application, or execution of a software plan, idea, model, design, specification, standard, algorithm, or policy.

Implementation is a realization of a technical specification or algorithm of a program, software component or other computer system. Many implementation may exist for a given specification or standard. For example web browser contains implementation of World Wide Web consortium-recommended specification and software development tools contain implementation of programming language (Tanenbaum, 1992).

In the information technology industry, implementation refers to post-sales process of guiding a client from purchase to use of the software or hardware that was purchase. This includes requirement analysis, scope analysis, customization, system integration, user policies, user training and delivery. These steps are often overseen by a project manager using project management methodologies set forth in the project management body of knowledge. For this project work, the implementation will be considered and looked at as the translation of the algorithm and flowcharts into a computer understandable instruction using a preferable programming language called Microsoft visual basic.
Types of Implementation

There are basically three types of implementation. They include:

- Parallel run: the parallel run involves running the newly developed system alongside the existing system.
- Pilot run: this method is implemented when the system is a very large one. Some part of the system is implemented for a period of time then the other part of the system follows suit. Each part of the system is also designed to function independently.
- Direct change over: in the direct change over method of implementation, the old system is totally dropped and the new system is put to use. This is a bad system of implementation. If there is any shortfall in the new system, all activities come to a halt until the error is rectified. This method does not give room for output comparison and hence may not allow the developer know if the system is producing the required result.

Choice of Implementation Technique

The type of implementation that is preferred by most system developers is parallel run implementation since the output of the new system can be compared with the output of the existing system thereby enabling all system errors to be corrected within the testing period.

Setup and Installation

In order to test the developed system, the system must be installed first. The following steps are followed:

I. Insert and open the CD ROM that contain the setup file
II. Double-click on setup.exe
III. Follow the onscreen instruction till the installation is complete.

There is also another method of doing this.

I. Insert and open the CD ROM that contain the setup file
II. Copy the folder ‘banker algorithm’ to drive C
III. Create a shortcut by right-clicking on the executable file then navigate to ‘sent to’ then choose desktop

System Testing
System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system’s compliance with its specified requirement. Testing the developed system involves running the executable file and choosing from the different menu option which is limited to the capability of the software. To test the software after installation, follow the steps below

I. Click on start menu
II. Point to program files
III. Point to banker algorithm
IV. Locate and click on banker algorithm

**System Output**

The output of program is shown below.

![Fig. 1 Computer interface of the output.](http://www.projectssolutionsdomain.com)

**Example on the application of banker’s algorithm**

Assuming that the system distinguishes between four types of resources (A, B, C and D) the following as an example of how those resources could be distributed. Note that this example shows the system at an instant before a new request for resources arrives. Also, the types and number of resources are abstracted. Real systems for example would deal with much larger quantities of each resource.

Available system resources are
A B C D
3 1 1 2

Processes (currently allocated resources)

A B C D
P1 1 2 2 1
P2 1 0 3 3
P3 1 1 1 0

Processes (maximum resources)

A B C D
P1 3 3 2 2
P2 1 2 3 4
P3 1 1 5 0

We can show that the state above is a safe state by showing that it is possible for each process to acquire its maximum resources and terminate. This process is achieved with a computer program developed using the banker algorithm approach.

Fig. 2 Computer interface for implementing the banker’s algorithm

Discussion of the result
Fig. 3 Computer interface showing the results of a safe state.

Output analysis

P1 acquires 2A, 1B and 1D more resources, achieving its maximum. The system now still has 1A, no B, 1C and 1D resource available.

P1 terminates returning 3A, 3B, 2C, and 2D resources to the system. The system now has 4A, 3B, 3C and 3D resources available.

P2 acquires 2B and 1D extra resource, and then terminates returning all its resources. The system now has 5A, 3B, 6C, and 6D resources.

P3 acquires 4C resources and terminates. The system now has all resources: 6A, 4B, 7C and 6D resources.

Because all processes were able to terminate, this state is safe.

Example of an unsafe state in a system includes:

Available system resources

A B C D

http://www.projectsolutionsdomain.com
Processes (currently allocated resources)

A B C D

P1  1 2 2 1
P2  1 1 3 3
P3  1 1 1 0

Processes (maximum resources)

A B C D

P1  3 3 2 2
P2  1 2 3 4
P3  1 1 5 0

Fig. 4 Computer interface showing results for an unsafe state

Discussion of the Results
P1 is unable to acquire enough of B resources

P2 is unable to acquire enough B resources

P3 is unable to acquire enough C resources

No process can acquire enough resources to terminate, so this state is unsafe.

Note that in this example, no process was able to terminate. It is possible that some processes will be able to terminate but not all of them that would still be an unsafe state.

**Conclusion**

In this project, we have presented the bankers algorithm for prevention of deadlock problems in a computer operating system. Its application was described and a software was developed using visual basic 6.0 to illustrate and implement the algorithm to further show how it is applied on resource allocation.

The knowledge and implementation of banker’s algorithm in resource allocation in the area of computer operation is a welcome development since it helps in a very successful way to prevent a non-yielding process and also prevent a continuous execution of a process.

A typical system that has a deadlock preventive measure using the banker’s algorithm should know

- How much of each resource each process could possibly request
- How much of each resource each process is currently holding
- How much of each resource the system has available.

**References**


Lane, M.G. and Mooney, J.D, (1988) "A Practical Approach to Operating Systems", Boyd and Fraser


