Project Report #3

Deadlock Detection Algorithm

Sujeet Ojha
sxo160430@utdallas.edu
March 1, 2020

I. Problem Statement

The assigned task was to implement the deadlock detection algorithm as highlighted in Section 6.4 of *Operating Systems – Internals and Design Principles by William Stallings*. A total of three files containing the request matrix, allocation matrix, resource vector, or the available vector were required to be read in in order to implement the algorithm. The files, along with the algorithm, is used to detect a deadlock by taking account of tasks that need to be completed.

II. Importance of Deadlock Detection

In the world of programming, deadlocks can be disastrous as they can kill the performance of an application. Although there are deadlock prevention strategies that can be used to avoid deadlocks, these strategies generally limit resource access and impose significant restrictions. Therefore, it is essential to detect deadlocks without limiting resources or restricting process. The algorithm implemented in this task will ensure that process requests are analyzed, and deadlocks are detected.

III. Algorithm Usage

The deadlock detection algorithm can be used to detect deadlocks as frequently, or as marginally and possible. The algorithm can be used at every resource request to detect deadlocks early or be implemented in a way where the algorithm is used only if there are high chance of deadlocks occurring. Checking for deadlocks more frequently might increase the chance of detecting deadlocks. However, this method results in a chunk of the processor time being used. On the other hand, checking less frequently consumes less processor time but hinders the chances of early detection. As there are drawbacks and benefits to all methods of implementation, it is up the system manager, along with the programmer, to determine the most efficient method of usage based on the specific system.

III. Approach

The programming language selected for this project is C. The approach taken was to first gather information regarding the total number of processes, and the total number of resources for each process from the user. The user was also given the ability to enter the names of the text files they would like the data read in from. Text files storing data regarding the allocation matrix, the request matrix, and the resource vector were then read in as arrays. The available vector was calculated using the allocation matrix and the resource vector. After all important matrices and vectors were either read in or generated, the deadlock detection algorithm was then implemented and used to determine which processes are deadlocked.

IV. Solution

Implementation of the deadlock detection algorithm was a straight-forward task. The program first gathers user input regarding the problem size and the names of the three distinct text files containing the allocation matrix, the request matrix, and the resource vector. Information such as the total number of processes and the total number of resources were gathered to dynamically initialize storage variables. Once the important files were read in, and the contents were stored into either single or multi-dimensional arrays. After storing all of the required information, Processes that do not require any resources were marked as they cannot possibly result in deadlocks. The remaining processes were then analyzed using the request matrix, the allocation matrix, and the available vector in order to determine if deadlock exists. The algorithm was tested an abundant amount of times with different matrices and vectors in order to ensure that the program is functioning correctly. Numerous error handling methods were also utilized to ensure that the program does not face any preventable bugs.

Figures:

- *Plot (1):* Plot (1) depicts the sample matrices and vectors used in the algorithm.
- *Plot (2):* Plot (2) depicts the build and execution of the program in the terminal.

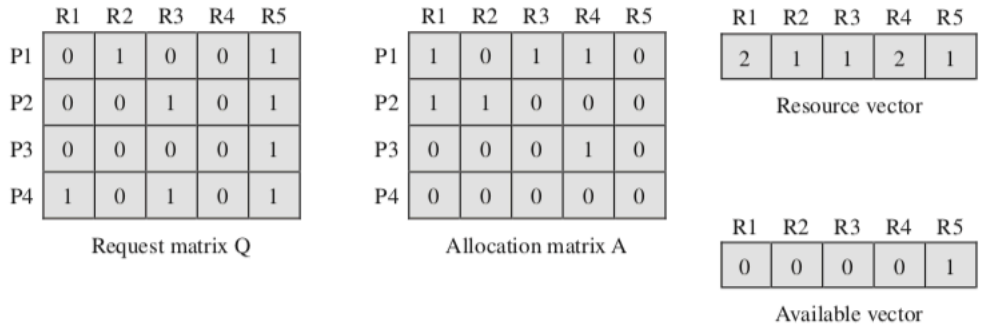

Figure 6.10 Example for Deadlock Detection

Plot (1): Sample Matrices and Vectors

The figure above displays a sample set of matrices and vectors used to determine which processes are deadlocked. This set of data illustrates the request matrix, the allocation matrix, the resource vector, and the available vector that define a unique set of possibilities for sequencing tasks. This report will utilize this sample data, along with additional samples, in order to test the reliability of the deadlock detection algorithm.

```
[cometnet-10-21-67-234:~ Sid$ gcc dlockDetect.c -o dead
[cometnet-10-21-67-234:~ Sid$ ./dead

Please enter the total number of processes: 4

Please enter the number of resources for each process: 5

Selection Menu
[0] - Type in file names
[1] - Automatically initialize file names
Please enter option 0 or 1: 0

Please enter the text file name for the Resource Vector: resourceVector.txt

Please enter the text file name for the Request Matrix: requestMatrix.txt

Please enter the text file name for the Allocation Matrix: allocationMatrix.txt


Resource Vector Display:
2 1 1 2 1

Request Matrix Display:
0 1 0 0 1
0 0 1 0 1
0 0 0 0 1
1 0 1 0 1

Allocation Matrix Display:
1 0 1 1 0
1 1 0 0 0
0 0 0 1 0
0 0 0 0 0

Available Vector Display:
0 0 0 0 1

Process 1 is deadlocked.
Process 2 is deadlocked.
```

Plot (2): Build and Execution of the program

The local terminal was used to build and execute the code. The program was traditionally compiled without added statements. Upon execution of the program, the user is asked to enter the total number of processes, along with the number of resources for each process in order to determine the size of the matrices. The user is also given the option to either automatically initialize text file names or enter in names of the text files. The contents of the read-in text files containing the set of matrices and vectors are then displayed. The deadlock detection algorithm is then performned and determination is made regarding which processes, if any, are deadlocked. In this specific case of execution with the sample data, it is concluded that Process 1 and Process 2 are deadlocked.