Deadlock Algorithm

1. Problem Statement

The assigned task is to run a deadlock algorithm to determine how many processes are deadlocked based on the given information to the program thorough matrices and vectors. Four files are passed, request matrix, allocation matrix, resource vector, and availability vector. The contents of the files are passed into dynamic 2d and 1d arrays, and then the deadlock algorithm is run.

1. Approach

Using the C language four files were passed into four functions, each function passed its respective contents to a 2d or 1d array. For example, the request and allocation matrices were passed into 2d arrays and resource and availability vectors were passed into 1d arrays. The main function then used these matrices and vectors to perform a deadlock algorithm in which the processes that either require zero resources or can perform their process are marked. All remaining unmarked processes are considered deadlocked. The code was developed in CLion and tested on cslinux1.

1. Solution

To run the code on the Linux terminal please use the following commands:

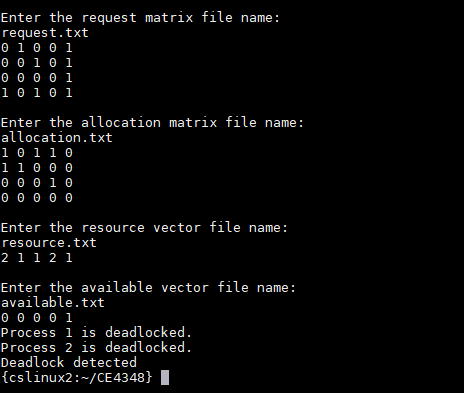
**$ gcc pgm3.c**

**$ ./a.out (number of processes) (number of resources)**

The command line arguments must be integers and need to be in this order shown above. When the code is executed, the code will request the name of the files, the request matrix file, the allocation matrix file, the resource vector file and the availability vector file, in that order. Please make sure to include the (filename).txt.

Example:

**$ ./a.out 4 5**

****

The above image shows an example found in the textbook. The program will ask the user for the various file names and display contents of each file and then finally perform the deadlock algorithm. Prints the results of the algorithm and whether deadlocks are detected.