

## **Deadlock Detection**

### **Problem Statement**

The assigned task was to implement a given deadlock detection algorithm. The project is important because it ensures that students have the ability to convert concepts and algorithms into code. It also encourages using best practices, as students will have difficulty debugging if they don't make proper use of functions for business logic abstraction.

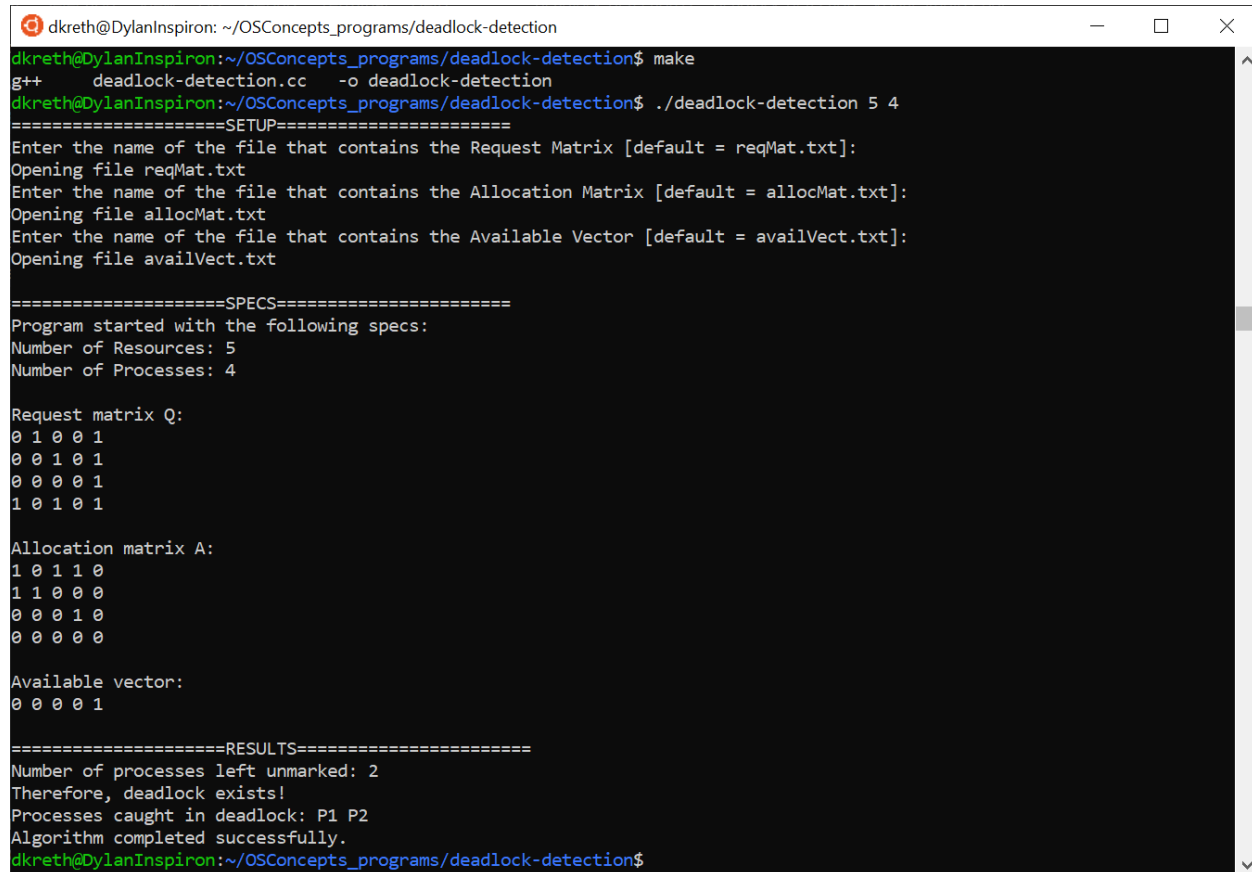
### **Approach**

C++ was used because it has convenient file I/O. A Makefile was used to build the project during development and keep track of dependencies. Git was used to incrementally develop and back up the code to a private GitLab repository. TA or Professor access to this repository is available on request. The majority of the source code is written in `deadlock-detection.cc`. Header file `deadlock-detection.h` declares constants and a function prototype. The results are printed to the console (`stdout`).

Dylan Kreth  
CE 4348.001 S20  
Program 3

## Solution

The project can be built by running `make` and then executed with `./deadlock-detection 5 4`. Replace 5 with the number of resources, and replace 4 with the number of processes. Enter the appropriate file names as they are requested, or press enter to use the default file names. The user-defined specs will be printed to the console (stdout). The deadlock-detection algorithm will run and print its results to the console (stdout). The below picture shows example program execution. `make deepclean` will remove deadlock-detection (along with any other temp files).



```
dkreth@DylanInspiron: ~/OSConcepts_programs/deadlock-detection
dkreth@DylanInspiron:~/OSConcepts_programs/deadlock-detection$ make
g++    deadlock-detection.cc  -o deadlock-detection
dkreth@DylanInspiron:~/OSConcepts_programs/deadlock-detection$ ./deadlock-detection 5 4
=====SETUP=====
Enter the name of the file that contains the Request Matrix [default = reqMat.txt]:
Opening file reqMat.txt
Enter the name of the file that contains the Allocation Matrix [default = allocMat.txt]:
Opening file allocMat.txt
Enter the name of the file that contains the Available Vector [default = availVect.txt]:
Opening file availVect.txt

=====SPECS=====
Program started with the following specs:
Number of Resources: 5
Number of Processes: 4

Request matrix Q:
0 1 0 0 1
0 0 1 0 1
0 0 0 0 1
1 0 1 0 1

Allocation matrix A:
1 0 1 1 0
1 1 0 0 0
0 0 0 1 0
0 0 0 0 0

Available vector:
0 0 0 0 1

=====RESULTS=====
Number of processes left unmarked: 2
Therefore, deadlock exists!
Processes caught in deadlock: P1 P2
Algorithm completed successfully.
dkreth@DylanInspiron:~/OSConcepts_programs/deadlock-detection$
```

## Results

Running the algorithm on the given data set shows that there exists a deadlock between process 1 and process 2. However, altering the request matrix of process 1 to be 0 0 0 0 1 caused the deadlock to go away. Another important thing to note is that the program will assert that there is a deadlock if there are not sufficient resources to complete any of the given processes, even if there is only one process left. This functionality is intentional.