Operating Systems Lab

Assignment 5

Name: Aditya Kangune **Batch**: K11
Roll number: 33323 **Date of Submission**: 23/10/2021

Problem Statement:

Implement the C program for Deadlock Avoidance Algorithm: Bankers Algorithm.

Theory:

Deadlock:

- A deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.
- Deadlock can arise if the following four conditions hold simultaneously:
- **Mutual Exclusion**: One or more than one resources are non-shareable (Only one process can use at a time)
- **Hold and Wait**: A process is holding at least one resource and waiting for resources.
- **No Pre-emption**: A resource cannot be taken from a process unless the process releases the resource.
- **Circular Wait**: A set of processes are waiting for each other in circular form.

Bankers Algorithm:

- The banker's algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation for predetermined maximum possible amounts of

all resources, then makes a “safe state” check to test for possible activities, before deciding whether allocation should be allowed to continue.

- Data Structures used in Banker’s algorithm Consider there are a total of n process and m resources.
- The data structures used are as follows :
    - Available matrix: Array of length m representing a number of available resources for each type. •
    - Max matrix: n x m matrix representing the maximum number of resources a process can request of each type.
    - Allocation matrix: n x m matrix representing the number of resources of each type allocated to a process.
    - Need matrix: n x m matrix representing the number of resources required of each type for the processes.

Characteristics of Banker’s algorithm:

- Consider there are total of n process and m resources. The data structures used are as follows : • If any process requests for a resource, then it has to wait. • This algorithm consists of advanced features for maximum resource allocation. • There are limited resources in the system we have. • In this algorithm, if any process gets all the needed resources, then it is that it should return the resources in a restricted period

Critical Section

- The critical section is termed as that section where the thread accesses the shared resources.
- The thread that accesses the shared resources can either modify, update or wipe out the underlying data structures which may cause conflicts with other threads if not handled properly.

- Hence, a section of code (Instruction trace) is defined as the critical section of the thread wherein the thread accesses the shared resources.
- In order to avoid conflicts, the principle of mutual exclusion comes into the picture.
- Here, the section where the producer or consumer threads request access to the buffer is termed as the critical section.

Details of Banker's Algorithm:

- Banker's Algorithm comprises of 2 algorithms :
    - **Safety Algorithm**: Determine whether the system is in a safe state.
    - **Resource Request Algorithm**: Determine whether the provided resource can be granted in a safe state

Conclusion:

- The meaning and cause of Deadlock were studied.
- Bankers Algorithm was implemented in order to avoid deadlock and use the strategy of a real-life bank to solve an OS problem.

Main program:

CASE 1: No Safe Sequence exists

Compiling and executing:

```
PS C:\Users\Adi\Desktop\College new\OS\Assignment 5> gcc assignment_5.c
PS C:\Users\Adi\Desktop\College new\OS\Assignment 5> ./a
```

Input of number of processes and resources:

```
+------------------------------+ INPUT +------------------------------+
Number of processes:
5
Number of resources:
3
```

Maximum needed:

```
+------------------------------+ MAX NEEDED +------------------------------+
Enter [0,0] of maximum needed matrix
7
Enter [0,1] of maximum needed matrix
5
Enter [0,2] of maximum needed matrix
3
Enter [1,0] of maximum needed matrix
3
Enter [1,1] of maximum needed matrix
2
Enter [1,2] of maximum needed matrix
2
Enter [2,0] of maximum needed matrix
9
Enter [2,1] of maximum needed matrix
0
Enter [2,2] of maximum needed matrix
2
Enter [3,0] of maximum needed matrix
2
Enter [3,1] of maximum needed matrix
2
Enter [3,2] of maximum needed matrix
2
Enter [4,0] of maximum needed matrix
4
Enter [4,1] of maximum needed matrix
3
Enter [4,2] of maximum needed matrix
3
Maximum needed matrix:
7       5       3
3       2       2
9       0       2
2       2       2
4       3       3
```

Allocated:

```
+------------------------------+ ALLOCATED +------------------------------+
Enter [0,0] of allocated matrix
0
Enter [0,1] of allocated matrix
1
Enter [0,2] of allocated matrix
0
Enter [1,0] of allocated matrix
2
Enter [1,1] of allocated matrix
0
Enter [1,2] of allocated matrix
0
Enter [2,0] of allocated matrix
3
Enter [2,1] of allocated matrix
0
Enter [2,2] of allocated matrix
2
Enter [3,0] of allocated matrix
2
Enter [3,1] of allocated matrix
1
Enter [3,2] of allocated matrix
1
Enter [4,0] of allocated matrix
0
Enter [4,1] of allocated matrix
0
Enter [4,2] of allocated matrix
0
Allocated matrix:
0       1       0
2       0       0
3       0       2
2       1       1
0       0       0
```

Resources:

```
+------------------------------+ RESOURCES +------------------------------+
Enter availiblity of resources [1]
2
Enter availiblity of resources [2]
2
Enter availiblity of resources [3]
1
```

Result:

```
Need matrix:
7       4       3
1       2       2
6       0       0
0       1       1
4       3       3
Completed matrix:
0
0
0
0
0

Completed matrix after traversal:
0       0       0       1       0

Completed matrix after traversal:
0       1       1       1       1
ERROR!! NOT POSSIBLE!!

+--------------------------------+ MENU  +--------------------------------+
```

MENU:

```
+--------------------------------+ MENU  +--------------------------------+

1) Press + to enter new request.
2) Any other key to terminate.

 +--------------------------------+ THANK YOU  +--------------------------------+
PS C:\Users\Adi\Desktop\College new\OS\Assignment 5>
```

CASE 2: Safe sequence is possible

Compiling and executing:

```
PS C:\Users\Adi\Desktop\College new\OS\Assignment 5> gcc assignment_5.c
PS C:\Users\Adi\Desktop\College new\OS\Assignment 5> ./a
```

Input of number of processes and resources:

```
PS C:\Users\Adi\Desktop\College new\OS\Assignment 5> ./a

+-------------------------------+ INPUT +-------------------------------+
Number of processes:
5
Number of resources:
3
```

Maximum needed:

```
+------------------------------+ MAX NEEDED +------------------------------+
Enter [0,0] of maximum needed matrix
7
Enter [0,1] of maximum needed matrix
5
Enter [0,2] of maximum needed matrix
3
Enter [1,0] of maximum needed matrix
3
Enter [1,1] of maximum needed matrix
2
Enter [1,2] of maximum needed matrix
2
Enter [2,0] of maximum needed matrix
9
Enter [2,1] of maximum needed matrix
0
Enter [2,2] of maximum needed matrix
2
Enter [3,0] of maximum needed matrix
2
Enter [3,1] of maximum needed matrix
2
Enter [3,2] of maximum needed matrix
2
Enter [4,0] of maximum needed matrix
4
Enter [4,1] of maximum needed matrix
3
Enter [4,2] of maximum needed matrix
3
Maximum needed matrix:
7       5       3
3       2       2
9       0       2
2       2       2
4       3       3
```

Allocated:

```
+------------------------------+ ALLOCATED +------------------------------+
Enter [0,0] of allocated matrix
0
Enter [0,1] of allocated matrix
1
Enter [0,2] of allocated matrix
0
Enter [1,0] of allocated matrix
2
Enter [1,1] of allocated matrix
0
Enter [1,2] of allocated matrix
0
Enter [2,0] of allocated matrix
3
Enter [2,1] of allocated matrix
0
Enter [2,2] of allocated matrix
2
Enter [3,0] of allocated matrix
2
Enter [3,1] of allocated matrix
1
Enter [3,2] of allocated matrix
1
Enter [4,0] of allocated matrix
0
Enter [4,1] of allocated matrix
0
Enter [4,2] of allocated matrix
2
Allocated matrix:
0       1       0
2       0       0
3       0       2
2       1       1
0       0       2
```

Resources:

```
+------------------------------+ RESOURCES +------------------------------+
Enter availiblity of resources [1]
3
Enter availiblity of resources [2]
3
Enter availiblity of resources [3]
2
```

Result:

```
Need matrix:
7       4       3
1       2       2
6       0       0
0       1       1
4       3       1
Completed matrix:
0
0
0
0
0

Completed matrix after traversal:
0       1       0       1       1

Completed matrix after traversal:
1       1       1       1       1

COMPLETED!!

+------------------------------+ SAFE SEQUENCE +------------------------------+
2 -> 4 -> 5 -> 1 -> 3

+------------------------------++------------------------------+

+------------------------------+ MENU  +------------------------------+

1) Press + to enter new request.
2) Any other key to terminate.

 +------------------------------+ THANK YOU  +------------------------------+
```

MENU:

```
+------------------------------+ MENU  +------------------------------+

1) Press + to enter new request.
2) Any other key to terminate.

 +------------------------------+ THANK YOU  +------------------------------+
PS C:\Users\Adi\Desktop\College new\OS\Assignment 5>
```

CASE 3: Resource allocation is rejected because requested resources exceed the maximum needed by that process.

Compiling and executing:

```
PS C:\Users\Adi\Desktop\College new\OS\Assignment 5> gcc assignment_5.c
PS C:\Users\Adi\Desktop\College new\OS\Assignment 5> ./a
```

Input of number of processes and resources:

```
PS C:\Users\Adi\Desktop\College new\OS\Assignment 5> ./a

+------------------------------+ INPUT +------------------------------+
Number of processes:
5
Number of resources:
3
```

Maximum needed:

```
+------------------------------+ MAX NEEDED +------------------------------+
Enter [0,0] of maximum needed matrix
7
Enter [0,1] of maximum needed matrix
5
Enter [0,2] of maximum needed matrix
3
Enter [1,0] of maximum needed matrix
3
Enter [1,1] of maximum needed matrix
2
Enter [1,2] of maximum needed matrix
2
Enter [2,0] of maximum needed matrix
9
Enter [2,1] of maximum needed matrix
0
Enter [2,2] of maximum needed matrix
2
Enter [3,0] of maximum needed matrix
2
Enter [3,1] of maximum needed matrix
2
Enter [3,2] of maximum needed matrix
2
Enter [4,0] of maximum needed matrix
4
Enter [4,1] of maximum needed matrix
3
Enter [4,2] of maximum needed matrix
3
Maximum needed matrix:
7       5       3
3       2       2
9       0       2
2       2       2
4       3       3
```

Allocated:

```
+------------------------------+ ALLOCATED +------------------------------+
Enter [0,0] of allocated matrix
0
Enter [0,1] of allocated matrix
1
Enter [0,2] of allocated matrix
0
Enter [1,0] of allocated matrix
2
Enter [1,1] of allocated matrix
0
Enter [1,2] of allocated matrix
0
Enter [2,0] of allocated matrix
3
Enter [2,1] of allocated matrix
0
Enter [2,2] of allocated matrix
2
Enter [3,0] of allocated matrix
2
Enter [3,1] of allocated matrix
1
Enter [3,2] of allocated matrix
1
Enter [4,0] of allocated matrix
0
Enter [4,1] of allocated matrix
0
Enter [4,2] of allocated matrix
2
Allocated matrix:
0       1       0
2       0       0
3       0       2
2       1       1
0       0       2
```

Resources:

```
+------------------------------+ RESOURCES +------------------------------+
Enter availiblity of resources [1]
3
Enter availiblity of resources [2]
3
Enter availiblity of resources [3]
2
```

Result:

```
Need matrix:
7       4       3
1       2       2
6       0       0
0       1       1
4       3       1
Completed matrix:
0
0
0
0
0

Completed matrix after traversal:
0       1       0       1       1

Completed matrix after traversal:
1       1       1       1       1

COMPLETED!!

+------------------------------+ SAFE SEQUENCE +------------------------------+
2 -> 4 -> 5 -> 1 -> 3

+------------------------------++------------------------------+

+------------------------------+ MENU  +------------------------------+

1) Press + to enter new request.
2) Any other key to terminate.

 +------------------------------+ THANK YOU  +------------------------------+
```

```
+------------------------------+ MENU  +------------------------------+

1) Press'+' to enter new request.
2) Press '-' to terminate.
+

Enter process number: 4

Enter the units of resource [1] requested: 3

Enter the units of resource [2] requested: 2

Enter the units of resource [3] requested: 4

ERROR: Process exceededing maximum limit of resources!!
Request CANNOT be accepted!
+------------------------------+ MENU  +------------------------------+

1) Press'+' to enter new request.
2) Press '-' to terminate.
```

CASE 4: Resource allocation is rejected because requested resources exceed resources available

Compiling and executing:

```
PS C:\Users\Adi\Desktop\College new\OS\Assignment 5> gcc assignment_5.c
PS C:\Users\Adi\Desktop\College new\OS\Assignment 5> ./a
```

Input of number of processes and resources:

```
PS C:\Users\Adi\Desktop\College new\OS\Assignment 5> ./a

+------------------------------+ INPUT +------------------------------+
Number of processes:
5
Number of resources:
3
```

Maximum needed:

```
+------------------------------+ MAX NEEDED +------------------------------+
Enter [0,0] of maximum needed matrix
7
Enter [0,1] of maximum needed matrix
5
Enter [0,2] of maximum needed matrix
3
Enter [1,0] of maximum needed matrix
3
Enter [1,1] of maximum needed matrix
2
Enter [1,2] of maximum needed matrix
2
Enter [2,0] of maximum needed matrix
9
Enter [2,1] of maximum needed matrix
0
Enter [2,2] of maximum needed matrix
2
Enter [3,0] of maximum needed matrix
2
Enter [3,1] of maximum needed matrix
2
Enter [3,2] of maximum needed matrix
2
Enter [4,0] of maximum needed matrix
4
Enter [4,1] of maximum needed matrix
3
Enter [4,2] of maximum needed matrix
3
Maximum needed matrix:
7       5       3
3       2       2
9       0       2
2       2       2
4       3       3
```

Allocated:

```
+------------------------------+ ALLOCATED +------------------------------+
Enter [0,0] of allocated matrix
0
Enter [0,1] of allocated matrix
1
Enter [0,2] of allocated matrix
0
Enter [1,0] of allocated matrix
2
Enter [1,1] of allocated matrix
0
Enter [1,2] of allocated matrix
0
Enter [2,0] of allocated matrix
3
Enter [2,1] of allocated matrix
0
Enter [2,2] of allocated matrix
2
Enter [3,0] of allocated matrix
2
Enter [3,1] of allocated matrix
1
Enter [3,2] of allocated matrix
1
Enter [4,0] of allocated matrix
0
Enter [4,1] of allocated matrix
0
Enter [4,2] of allocated matrix
2
Allocated matrix:
0       1       0
2       0       0
3       0       2
2       1       1
0       0       2
```

Resources:

```
+------------------------------+ RESOURCES +------------------------------+
Enter availiblity of resources [1]
3
Enter availiblity of resources [2]
3
Enter availiblity of resources [3]
2
```

Result:

```
Need matrix:
7       4       3
1       2       2
6       0       0
0       1       1
4       3       1
Completed matrix:
0
0
0
0
0

Completed matrix after traversal:
0       1       0       1       1

Completed matrix after traversal:
1       1       1       1       1

COMPLETED!!

+------------------------------+ SAFE SEQUENCE +------------------------------+
2 -> 4 -> 5 -> 1 -> 3

+------------------------------++------------------------------+

+------------------------------+ MENU  +------------------------------+

1) Press + to enter new request.
2) Any other key to terminate.

 +------------------------------+ THANK YOU  +------------------------------+
```

```
+--------------------------------+ MENU  +--------------------------------+

1) Press'+' to enter new request.
2) Press '-' to terminate.
+

Enter process number: 2

Enter the units of resource [1] requested: 1

Enter the units of resource [2] requested: 1

Enter the units of resource [3] requested: 1

ERROR: Process exceededing available resources!!
Request CANNOT be accepted!
+--------------------------------+ MENU  +--------------------------------+

1) Press'+' to enter new request.
2) Press '-' to terminate.
```

CASE 5: Request rejected because no safe sequence is possible.

```
+--------------------------------+ MENU  +--------------------------------+

1) Press'+' to enter new request.
2) Press '-' to terminate.
+

Enter process number: 1

Enter the units of resource [1] requested: 1

Enter the units of resource [2] requested: 1

Enter the units of resource [3] requested: 1
ERROR!! NOT POSSIBLE!!

+-------------------------------+ REQUEST REJECTED +-------------------------------+

+-------------------------------+ NO SAFE SEQUENCE +-------------------------------+
```

CODE:

```
// COMPILE: gcc assignment_5.c
```

```c
// RUN: ./a
/*
33323
PROBLEM STATEMENT:
Implement the Deadlock Avoidance Algorithm: Bankers Algorithm.
*/
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <math.h>


//Main function
int main() {

    int numberOfProcesses;
    int numberOfResources;
    printf(" \n+-------------------------------+ INPUT
+-------------------------------+\n");
    // Input of number of processes and resources
    printf("Number of processes: \n");
    scanf("%d",&numberOfProcesses);
    printf("Number of resources: \n");
    scanf("%d",&numberOfResources);

    // Declaring matrices
    int available[numberOfResources];
    int availableOriginal[numberOfResources];
    int maxNeeded[50][50];
    int allocated[50][50];
    int needMatrix[50][50];
    int completed[50];
    int safeSequence[50];
    int safeIndex = 0;
    int requestedResources[numberOfResources];
    printf(" \n+-------------------------------+ MAX NEEDED
+-------------------------------+\n");
    for(int i=0; i<numberOfProcesses; i++)
    {
        for(int j=0; j<numberOfResources; j++)
```

```
        {
            int temp;
            printf("Enter [%d,%d] of maximum needed matrix \n",i,j);
            scanf("%d",&temp);
            maxNeeded[i][j]=temp;
        }
    }
    printf("Maximum needed matrix: \n");
    for(int i=0;i<numberOfProcesses;i++)
    {
        completed[i]=0;
        for(int j=0;j<numberOfResources;j++)
        {
            printf("%d \t",maxNeeded[i][j]);
        }
        printf("\n");
    }
    printf(" \n+------------------------------+ ALLOCATED
+------------------------------+\n");
    for(int i=0;i<numberOfProcesses;i++)
    {
        for(int j=0;j<numberOfResources;j++)
        {
            int temp;
            printf("Enter [%d,%d] of allocated matrix \n",i,j);
            scanf("%d",&temp);
            allocated[i][j]=temp;
        }
    }
    printf("Allocated matrix: \n");
    for(int i=0;i<numberOfProcesses;i++)
    {
        completed[i]=0;
        for(int j=0;j<numberOfResources;j++)
        {
            printf("%d \t",allocated[i][j]);
        }
        printf("\n");
    }
```

```
    printf(" \n+-------------------------------+ RESOURCES
+-------------------------------+\n");

    for(int i=0;i<numberOfResources;i++)
    {
        int temp;
        printf("Enter availiblity of resources [%d] \n", i+1);
        scanf("%d",&temp);
        available[i]=temp;

    }
    for(int i=0;i<numberOfResources;i++)
    {
        availableOriginal[i]=available[i];

    }

    printf("Need matrix: \n");
    for(int i=0;i<numberOfProcesses;i++)
    {
        completed[i]=0;
        for(int j=0;j<numberOfResources;j++)
        {
            needMatrix[i][j]=maxNeeded[i][j]-allocated[i][j];
            printf("%d \t",needMatrix[i][j]);
        }
        printf("\n");
    }
    printf("Completed matrix: \n");
    for(int i=0;i<numberOfProcesses;i++)
    {
        printf("%d \t",completed[i]);
        printf("\n");
    }

    // Calculation
    int calculate(){
        // Setting completed to zero
        for(int i=0;i<numberOfProcesses;i++)
        {
```

```
            completed[i]=0;
        }
        for(int i=0;i<numberOfResources;i++)
        {
            available[i]=availableOriginal[i];

        }

        safeIndex=0;

        while(1)
        {
            int allDone=1;
            // If all processes completed => break
            for(int i=0;i<numberOfProcesses;i++)
            {
                if(completed[i]==0)
                {
                    allDone=0;
                }
            }
            if(allDone==1)
            {
                printf("\nCOMPLETED!!\n");
                printf(" \n+------------------------------+ SAFE SEQUENCE
+------------------------------+\n");
                for(int i=0;i<numberOfProcesses;i++)
                {
                    if(i==numberOfProcesses-1)
                    {
                        printf("%d ",safeSequence[i]);
                        printf("\n");
                    }
                    else
                    {
                        printf("%d -> ",safeSequence[i]);
                    }

                }
```

```
                printf("
\n+-----------------------------++-----------------------------+\n");
                return 1;
            }

            int zeroChanges=1;
            for(int i=0;i<numberOfProcesses;i++)
            {
                if(completed[i]==0)
                {
                    int allNeedsLesser=1;
                    for(int j=0;j<numberOfResources;j++)
                    {
                        if(needMatrix[i][j]>available[j])
                        {
                            allNeedsLesser=0;
                        }
                    }
                    if(allNeedsLesser==1)
                    {
                        for(int j=0;j<numberOfResources;j++)
                        {
                            available[j]=available[j]+allocated[i][j];

                        }
                        zeroChanges=0;
                        safeSequence[safeIndex]=i+1;
                        safeIndex++;
                        completed[i]=1;
                    }

                }
            }
            if(zeroChanges)
            {
                printf("ERROR!! NOT POSSIBLE!! \n");

                return 0;
            }
            printf("\nCompleted matrix after traversal: \n");
```

```
            for(int i=0;i<numberOfProcesses;i++)
            {
                printf("%d \t",completed[i]);

            }
            printf("\n");

        }
    }
    calculate();
    void resourceRequest()
    {
          int processNo;
          printf("\nEnter process number: ");
          scanf("%d",&processNo);

          for(int i=0;i<numberOfResources;i++)
          {
            printf("\nEnter the units of resource [%d] requested: ",i+1);
            scanf("%d",&requestedResources[i]);
          }

          for(int j=0;j<numberOfResources;j++)
          {
            // Checking if resources requested do no cross max needed for
that process
            if(requestedResources[j]<=maxNeeded[processNo][j])
            {
            // Checking if resources requested do no cross available
resources
            if(requestedResources[j]<=available[j])
            {

availableOriginal[j]=availableOriginal[j]-requestedResources[j];

allocated[processNo][j]=allocated[processNo][j]+requestedResources[j];

needMatrix[processNo][j]=needMatrix[processNo][j]-requestedResources[j];
             }
            else
```

```
            {
                  printf("\nERROR: Process exceededing available
resources!!\nRequest CANNOT be accepted!");
                  return;
            }
            }
            else
            {
                printf("\nERROR: Process exceededing maximum limit of
resources!!\nRequest CANNOT be accepted!");
                return;
            }
          }
        int status=calculate();
        if(status==1)
        {
            printf(" \n+-------------------------------+ REQUEST ACCEPTED
+-------------------------------+\n");
        }
        else
        {
             for(int j=0;j<numberOfResources;j++)
            {

availableOriginal[j]=availableOriginal[j]+requestedResources[j];

allocated[processNo][j]=allocated[processNo][j]-requestedResources[j];

needMatrix[processNo][j]=needMatrix[processNo][j]+requestedResources[j];
            }

            printf(" \n+-------------------------------+ REQUEST REJECTED
+-------------------------------+\n");
            printf(" \n+-------------------------------+ NO SAFE SEQUENCE
+-------------------------------+\n");
        }


    }
```

```
    while(1)
    {
        printf(" \n+-------------------------------+ MENU
+-------------------------------+\n");
        printf("\n1) Press'+' to enter new request.\n2) Press '-' to
terminate. \n");
        char choice;
        scanf("%c",&choice);
        if(choice=='+')
        {
            resourceRequest();
        }
        else if(choice == '-')
        {
            break;
        }
    }
    printf("\n +-------------------------------+ THANK YOU
+-------------------------------+\n");
    return 0;

}
```