Yash Rajesh Akolu
21BCE5780

School of Computer Science Engineering (SCOPE)

B. Tech – Computer Science and Engineering

BCSE303L – Operating System

DA1 & DA2

Submitted By

Yash Rajesh Akolu <-> 21BCE5780

Submitted To

Dr. Premalatha

DATE: 9/06/2024

Yash Rajesh Akolu
21BCE5780

Operating System

The topic on which the project is done is:

Resource Management and Allocation System
Designing a comprehensive system for managing and allocating system resources like CPU, memory, and I/O devices.

Video Link:

https://youtu.be/QEpOIhBDJJY

GitHub link for code:

https://github.com/yAsh2537/Operating-System-DA_21BCE5780.git

This C++ program implements the Banker's Algorithm, a cornerstone method in resource management and allocation systems. The Banker's Algorithm helps in managing and allocating system resources such as CPU, memory, and I/O devices efficiently. Its primary goal is to prevent deadlock by ensuring the system remains in a safe state, even as multiple processes request and release resources dynamically.

Detailed Explanation

Initialization and Input

1. **Resource and Process Information**:
   - The program starts by taking input for the number of resources (`countofr`, up to 5) and the number of processes (`countofp`, up to 10).
   - It then asks for the maximum instances of each resource available in the system.

2. **Allocation and Maximum Demand Matrices**:

   The program captures the current allocation of resources to each process.

   It also captures the maximum demand of each resource by each process.

3. **Need Matrix Calculation**:

   The need matrix is calculated by subtracting the allocation matrix from the maximum demand matrix. This matrix represents the remaining resources needed by each process to complete its execution.

Display of Input Data

The program prints out the total resources in the system, the allocation matrix, the maximum demand matrix, and the calculated need matrix to provide a clear overview of the current resource allocation state.

Safety Check Using Banker's Algorithm

The Banker's Algorithm checks if the system can allocate resources in such a way that all processes can complete without leading to a deadlock. It follows these steps:

- It iterates through each process, checking if it can satisfy its remaining needs with the available resources.
- If a process can be completed, it is marked as finished, and its resources are released back to the system.
- This process continues until all processes are completed or no further processes can be completed (indicating an unsafe state).

Result Output

Finally, the program outputs whether the system is in a safe or unsafe state. If the system is safe, it prints the safe sequence of process execution.

Relevance to Resource Management and Allocation System

- **Dynamic Allocation**: The program dynamically allocates resources to processes based on their current needs and maximum demands.
- **Deadlock Avoidance**: By using the Banker's Algorithm, it ensures that the system avoids deadlocks, thus maintaining overall system stability and efficiency.
- **Resource Utilization**: It helps in optimal resource utilization by ensuring that resources are allocated in a manner that all processes can eventually complete without causing a deadlock.
- **Safety Check**: The program checks the safety of the current resource allocation state, ensuring that the system does not enter an unsafe state where deadlocks are possible.

Yash Rajesh Akolu
21BCE5780

CODE :

```
#include<iostream>
#include<conio.h>

using namespace std;

int main()
{
    int instance[5], count, sequence[10], safe, s = 0, j, completed, i;
    int available[5], allocation[10][5];
    int need[10][5], process, P[10], countofr, countofp, running[10];
    int maxDemand[10][5];

    cout << "\n Enter the number of resources (<=5): ";
    cin >> countofr;

    for(int i = 0; i < countofr; i++)
    {
        cout << "\n Enter the max instances of resource R[" << i << "]:";
        cin >> instance[i];
        available[i] = instance[i];
    }

    cout << "\n Enter the number of processes (<=10): ";
    cin >> countofp;
    cout << "\n Enter the allocation matrix \n";

    for(i = 0; i < countofp; i++)
    {
        cout << "FOR THE PROCESS :P[" << i << "]" << endl;
        for(int j = 0; j < countofr; j++)
        {
            cout << "allocation of resource R[" << j << "] is : ";
            cin >> allocation[i][j];
            available[j] -= allocation[i][j];
        }
    }

    cout << "\nEnter the MAX matrix \n\n";

    for(i = 0; i < countofp; i++)
    {
        cout << "FOR THE PROCESS P[" << i << "]" << endl;
        for(int j = 0; j < countofr; j++)
        {
            cout << "max demand of resource R[" << j << "] is : ";
            cin >> maxDemand[i][j];
```

Yash Rajesh Akolu
21BCE5780

```
        }
    }

    cout << "\n the given data are : \n";

    cout << endl << "\nTotal resources in system: \n\n ";
    for(i = 0; i < countofr; i++){
        cout << "R[" << i << "] ";
    }
    cout << endl;
    for(i = 0; i < countofr; i++){
        cout << "     " << instance[i];
    }

    cout << "\n\n Allocation matrix \n\nt";
    for(j = 0; j < countofr; j++){
        cout << "R[" << j << "] ";
    }
    cout << endl;

    for(i = 0; i < countofp; i++)
    {
        cout << "P[" << i << "] ";
        for(j = 0; j < countofr; j++)
            cout << "     " << allocation[i][j];
        cout << endl;
    }

    cout << "\n\n Max matrix \n\nt";
    for(j = 0; j < countofr; j++)
        cout << "R[" << j << "] ";
    cout << endl;

    for(i = 0; i < countofp; i++){
        cout << "P[" << i << "] ";
        for(j = 0; j < countofr; j++)
            cout << "     " << maxDemand[i][j];
        cout << endl;
    }

    for(i = 0; i < countofp; i++)
    {
        for(j = 0; j < countofr; j++)
        {
            need[i][j] = maxDemand[i][j] - allocation[i][j];
        }
    }
```

Yash Rajesh Akolu
21BCE5780

```
    cout << "\n\n NEED matrix \n\nt";
    for(j = 0; j < countofr; j++)
        cout << "R[" << j << "] ";
    cout << endl;

    for(i = 0; i < countofp; i++){
        cout << "P[" << i << "] ";
        for(j = 0; j < countofr; j++)
            cout << "    " << need[i][j];
        cout << endl;
    }

    cout << "\n NOW to check whether above state is safe";
    cout << "\n sequence in which above requests can be fulfilled";
    cout << "\n press any key to continue";
    getch();

    count = countofp;

    for(i = 0; i < countofp; i++){
        running[i] = 1;
    }

    while(count){
        safe = 0;
        for(i = 0; i < countofp; i++)
        {
            if(running[i]){
                completed = 1;
                for(j = 0; j < countofr; j++)
                {
                    if(need[i][j] > available[j])
                    {
                        completed = 0;
                        break;
                    }
                }
                if(completed)
                {
                    running[i] = 0;
                    count--;
                    safe = 1;
                    for(j = 0; j < countofr; j++)
                    {
                        available[j] += allocation[i][j];
                    }
                    sequence[s++] = i;
                    cout << "\n\n Running process P[" << i << "]";
```

Yash Rajesh Akolu
21BCE5780

```
                    cout << endl << "\n\nTotal resources now available:\n\n";
                    for(int k = 0; k < countofr; k++)
                        cout << "R[" << k << "] ";
                    cout << endl;
                    for(int k = 0; k < countofr; k++)
                        cout << "     " << available[k];
                    break;
                }
            }
        }
        if(!safe)
            break;
    }

    if(safe)
    {
        cout << "\nThe System is in safe state";
        cout << "\nSafe sequence is:";
        for(i = 0; i < countofp; i++)
        {
            cout << "\t" << "P[" << sequence[i] << "]";
        }
    }
    else
    {
        cout << "\nThe System is in unsafe state";
    }
    getch();
}
```

Yash Rajesh Akolu
21BCE5780

OUTPUT :

Safe State

```
PS Y:\yAsh\codevs\C++> cd "y:\yAsh\codevs\C++\"

 Enter the number of resources (<=5): 3

 Enter the max instances of resource R[0]:10

 Enter the max instances of resource R[1]:5

 Enter the max instances of resource R[2]:7

 Enter the number of processes (<=10): 5

 Enter the allocation matrix
FOR THE PROCESS :P[0]
allocation of resource R[0] is : 0
allocation of resource R[1] is : 1
allocation of resource R[2] is : 0
FOR THE PROCESS :P[1]
allocation of resource R[0] is : 2
allocation of resource R[1] is : 0
allocation of resource R[2] is : 0
FOR THE PROCESS :P[2]
allocation of resource R[0] is : 3
allocation of resource R[1] is : 0
allocation of resource R[2] is : 2
FOR THE PROCESS :P[3]
allocation of resource R[0] is : 2
allocation of resource R[1] is : 1
allocation of resource R[2] is : 1
FOR THE PROCESS :P[4]
allocation of resource R[0] is : 0
allocation of resource R[1] is : 0
allocation of resource R[2] is : 2
```

Yash Rajesh Akolu
21BCE5780

```
Enter the MAX matrix

FOR THE PROCESS P[0]
max demand of resource R[0] is : 7
max demand of resource R[1] is : 5
max demand of resource R[2] is : 3
FOR THE PROCESS P[1]
max demand of resource R[0] is : 3
max demand of resource R[1] is : 2
max demand of resource R[2] is : 2
FOR THE PROCESS P[2]
max demand of resource R[0] is : 9
max demand of resource R[1] is : 0
max demand of resource R[2] is : 0
FOR THE PROCESS P[3]
max demand of resource R[0] is : 2
max demand of resource R[1] is : 2
max demand of resource R[2] is : 2
FOR THE PROCESS P[4]
max demand of resource R[0] is : 4
max demand of resource R[1] is : 3
max demand of resource R[2] is : 3

 the given data are :


Total resources in system:

 R[0] R[1] R[2]
    10    5    7
```

Yash Rajesh Akolu
21BCE5780

```
Total resources in system:

 R[0] R[1] R[2]
    10    5    7

 Allocation matrix

tR[0] R[1] R[2]
P[0]     0    1    0
P[1]     2    0    0
P[2]     3    0    2
P[3]     2    1    1
P[4]     0    0    2


 Max matrix

tR[0] R[1] R[2]
P[0]     7    5    3
P[1]     3    2    2
P[2]     9    0    0
P[3]     2    2    2
P[4]     4    3    3


 NEED matrix

tR[0] R[1] R[2]
P[0]     7    4    3
P[1]     1    2    2
P[2]     6    0    -2
P[3]     0    1    1
P[4]     4    3    1

 NOW to check whether above state is safe
 sequence in which above requests can be fulfilled
 press any key to continue
```

Yash Rajesh Akolu
21BCE5780

```
 Running process P[1]


Total resources now available:

R[0] R[1] R[2]
    5    3    2

 Running process P[3]


Total resources now available:

R[0] R[1] R[2]
    7    4    3

 Running process P[0]


Total resources now available:

R[0] R[1] R[2]
    7    5    3

 Running process P[2]


Total resources now available:

R[0] R[1] R[2]
    10    5    5

 Running process P[4]


Total resources now available:

R[0] R[1] R[2]
    10    5    7
The System is in safe state
Safe sequence is:       P[1]    P[3]    P[0]    P[2]    P[4]
```

Yash Rajesh Akolu
21BCE5780

Unsafe State

```
PS Y:\yAsh\codevs\C++> cd "y:\yAsh\codevs\C++\"

 Enter the number of resources (<=5): 3

 Enter the max instances of resource R[0]:12

 Enter the max instances of resource R[1]:1

 Enter the max instances of resource R[2]:13

 Enter the number of processes (<=10): 11

 Enter the allocation matrix
FOR THE PROCESS :P[0]
allocation of resource R[0] is : 11
allocation of resource R[1] is : 2
allocation of resource R[2] is : 1
FOR THE PROCESS :P[1]
allocation of resource R[0] is : 3
allocation of resource R[1] is : 2
allocation of resource R[2] is : 2
FOR THE PROCESS :P[2]
allocation of resource R[0] is : 1
allocation of resource R[1] is : 13
allocation of resource R[2] is : 2
FOR THE PROCESS :P[3]
allocation of resource R[0] is : 1
allocation of resource R[1] is : 3
allocation of resource R[2] is : 3
FOR THE PROCESS :P[4]
allocation of resource R[0] is : 2
allocation of resource R[1] is : 2
allocation of resource R[2] is : 2
FOR THE PROCESS :P[5]
allocation of resource R[0] is : 3
allocation of resource R[1] is : 2
allocation of resource R[2] is : 1
```

```
FOR THE PROCESS :P[6]
allocation of resource R[0] is : 22
allocation of resource R[1] is : 2
allocation of resource R[2] is : 2
FOR THE PROCESS :P[7]
allocation of resource R[0] is : 3
allocation of resource R[1] is : 2
allocation of resource R[2] is : 2
FOR THE PROCESS :P[8]
allocation of resource R[0] is : 3
allocation of resource R[1] is : 3
allocation of resource R[2] is : 3
FOR THE PROCESS :P[9]
allocation of resource R[0] is : 1
allocation of resource R[1] is : 1
allocation of resource R[2] is : 1
FOR THE PROCESS :P[10]
allocation of resource R[0] is : 1
allocation of resource R[1] is : 1
allocation of resource R[2] is : 1

Enter the MAX matrix

FOR THE PROCESS P[0]
max demand of resource R[0] is : 3
max demand of resource R[1] is : 1
max demand of resource R[2] is : 3
FOR THE PROCESS P[1]
max demand of resource R[0] is : 2
max demand of resource R[1] is : 2
max demand of resource R[2] is : 1
FOR THE PROCESS P[2]
max demand of resource R[0] is : 1
max demand of resource R[1] is : 3
max demand of resource R[2] is : 1
FOR THE PROCESS P[3]
max demand of resource R[0] is : 1
max demand of resource R[1] is : 1
max demand of resource R[2] is : 1
FOR THE PROCESS P[4]
max demand of resource R[0] is : 2
max demand of resource R[1] is : 2
max demand of resource R[2] is : 2
```

```
FOR THE PROCESS P[5]
max demand of resource R[0] is : 2
max demand of resource R[1] is : 3
max demand of resource R[2] is : 1
FOR THE PROCESS P[6]
max demand of resource R[0] is : 1
max demand of resource R[1] is : 1
max demand of resource R[2] is : 1
FOR THE PROCESS P[7]
max demand of resource R[0] is : 1
max demand of resource R[1] is : 1
max demand of resource R[2] is : 1
FOR THE PROCESS P[8]
max demand of resource R[0] is : 11
max demand of resource R[1] is : 12
max demand of resource R[2] is : 11
FOR THE PROCESS P[9]
max demand of resource R[0] is : 123
max demand of resource R[1] is : 112
max demand of resource R[2] is : 12
FOR THE PROCESS P[10]
max demand of resource R[0] is : 11
max demand of resource R[1] is : 1
max demand of resource R[2] is : 1

 the given data are :


Total resources in system:

 R[0] R[1] R[2]
    12    1    13
```

Yash Rajesh Akolu
21BCE5780

```
 Allocation matrix

tR[0] R[1] R[2]
P[0]     11    2    1
P[1]     3    2    2
P[2]     1    13    2
P[3]     1    3    3
P[4]     2    2    2
P[5]     3    2    1
P[6]     22    2    2
P[7]     3    2    2
P[8]     3    3    3
P[9]     1    1    1
P[10]     0    0    0


 Max matrix

tR[0] R[1] R[2]
P[0]     3    1    3
P[1]     2    2    1
P[2]     1    3    1
P[3]     1    1    1
P[4]     2    2    2
P[5]     2    3    1
P[6]     1    1    1
P[7]     1    1    1
P[8]     11    12    11
P[9]     123    112    12
P[10]     11    1    1
```

Yash Rajesh Akolu
21BCE5780

```
 NEED matrix

tR[0] R[1] R[2]
P[0]     -8    -1    2
P[1]     -1    0    -1
P[2]     0    -10    -1
P[3]     0    -2    -2
P[4]     0    0    0
P[5]     -1    1    0
P[6]     -21    -1    -1
P[7]     -2    -1    -1
P[8]     8    9    8
P[9]     122    111    11
P[10]     11    1    1

 NOW to check whether above state is safe
 sequence in which above requests can be fulfilled
 press any key to continue
The System is in unsafe state
PS Y:\yAsh\codevs\C++>
```

---THE END---