Operating system

Lab task 3

By: Sparsh Arya

Registration number: 17BEC0656

Slot: L7

Implement Bankers Algorithm for Dead Lock Avoidance

Code

```
#include <stdio.h>

int main()
{
    int count = 0, m, n, process, temp, resource;
    int allocation_table[5] = {0, 0, 0, 0, 0};
    int available[5], current[5][5], maximum_claim[5][5];
    int maximum_resources[5], running[5], safe_state = 0;
    printf("\nEnter The Total Number Of Processes:\t");
    scanf("%d", &process);
    for(m = 0; m < process; m++)
    {
        running[m] = 1;
        count++;
    }
    printf("\nEnter The Total Number Of Resources To Allocate:\t");
    scanf("%d", &resource);
    printf("\nEnter The Claim Vector:\t");
    for(m = 0; m < resource; m++)
    {
        scanf("%d", &maximum_resources[m]);
    }
    printf("\nEnter Allocated Resource Table:\n");
```

```
for(m = 0; m < process; m++)
{
    for(n = 0; n < resource; n++)
    {
        scanf("%d", &current[m][n]);
    }
}
printf("\nEnter The Maximum Claim Table:\n");
for(m = 0; m < process; m++)
{
    for(n = 0; n < resource; n++)
    {
        scanf("%d", &maximum_claim[m][n]);
    }
}
printf("\nThe Claim Vector \n");
for(m = 0; m < resource; m++)
{
    printf("\t%d ", maximum_resources[m]);
}
printf("\n The Allocated Resource Table\n");
for(m = 0; m < process; m++)
{
    for(n = 0; n < resource; n++)
    {
        printf("\t%d", current[m][n]);
    }
    printf("\n");
}
printf("\nThe Maximum Claim Table \n");
```

```
for(m = 0; m < process; m++)
{
    for(n = 0; n < resource; n++)
    {
        printf("\t%d", maximum_claim[m][n]);
    }
    printf("\n");
}
for(m = 0; m < process; m++)
{
    for(n = 0; n < resource; n++)
    {
        allocation_table[n] = allocation_table[n] + current[m][n];
    }
}
printf("\nAllocated Resources \n");
for(m = 0; m < resource; m++)
{
    printf("\t%d", allocation_table[m]);
}
for(m = 0; m < resource; m++)
{
    available[m] = maximum_resources[m] - allocation_table[m];
}
printf("\nAvailable Resources:");
for(m = 0; m < resource; m++)
{
    printf("\t%d", available[m]);
}
printf("\n");
```

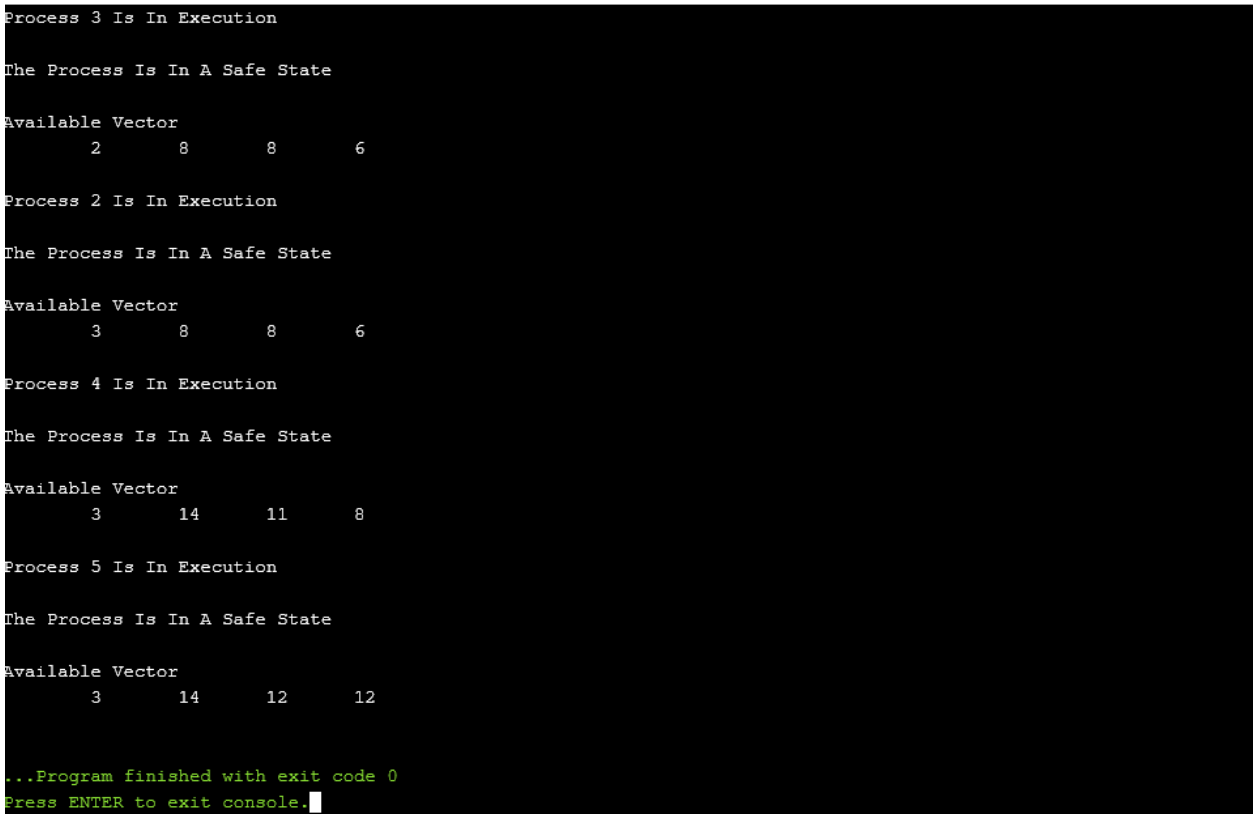
```
while(count != 0)
{
    safe_state = 0;
    for(m = 0; m < process; m++)
    {
        if(running[m])
        {
            temp = 1;
            for(n = 0; n < resource; n++)
            {
                if(maximum_claim[m][n] - current[m][n] > available[n])
                {
                    temp = 0;
                    break;
                }
            }
            if(temp)
            {
                printf("\nProcess %d Is In Execution \n", m + 1);
                running[m] = 0;
                count--;
                safe_state = 1;
                for(n = 0; n < resource; n++)
                {
                    available[n] = available[n] + current[m][n];
                }
                break;
            }
        }
    }
```

```
        if(!safe_state)
        {
            printf("\nThe Processes Are In An Unsafe State \n");
            break;
        }
        else
        {
            printf("\nThe Process Is In A Safe State \n");
            printf("\nAvailable Vector\n");
            for(m = 0; m < resource; m++)
            {
                printf("\t%d", available[m]);
            }
            printf("\n");
        }
    }
    return 0;
}
```

Output.

```
Enter The Total Number Of Processes:    5

Enter The Total Number Of Resources To Allocate:        4

Enter The Claim Vector: 3 14 12 12

Enter Allocated Resource Table:
0 0 1 2 1 0 0 0 1 3 5 4 0 6 3 2 0 0 1 4

Enter The Maximum Claim Table:
0 0 1 2 1 7 5 0 2 3 5 6 0 6 5 2 0 6 5 6

The Claim Vector
        3       14      12      12
 The Allocated Resource Table
        0       0       1       2
        1       0       0       0
        1       3       5       4
        0       6       3       2
        0       0       1       4

The Maximum Claim Table
        0       0       1       2
        1       7       5       0
        2       3       5       6
        0       6       5       2
        0       6       5       6

Allocated Resources
        2       9       10      12
```

```
Allocated Resources
        2       9       10      12
Available Resources:    1       5       2       0

Process 1 Is In Execution

The Process Is In A Safe State

Available Vector
        1       5       3       2

Process 3 Is In Execution

The Process Is In A Safe State

Available Vector
        2       8       8       6

Process 2 Is In Execution

The Process Is In A Safe State

Available Vector
        3       8       8       6

Process 4 Is In Execution

The Process Is In A Safe State

Available Vector
```


Process 3 Is In Execution
The Process Is In A Safe State
Available Vector
2 8 8 6
Process 2 Is In Execution
The Process Is In A Safe State
Available Vector
3 8 8 6
Process 4 Is In Execution
The Process Is In A Safe State
Available Vector
3 14 11 8
Process 5 Is In Execution
The Process Is In A Safe State
Available Vector
3 14 12 12
...Program finished with exit code 0
Press ENTER to exit console.

End.