

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Programme | : | **BTech. CSE Core** | Semester | : | **Win 2021-22** |
| Course | : | **Operating Systems** | Code | : | **CSE2005** |
| Faculty | : | **Dr. Shyamala L** | Slot | : | **L25+L26** |
| Name | : | **Hariket Sukesh Kumar Sheth** | Register No. | : | **20BCE1975** |

**Lab 7:**Deadlock in OS

**1**

|  |  |  |  |
| --- | --- | --- | --- |
| Date: 25-03-2022 | LAB 07 | Deadlock in OS |  |

**LAB 7**

Sample Question:

1. Consider the following snapshot of a system in which four resources A, B, C and D are available. The system currently contains 6 instances of A, 4 of resource B, 4 of resource C, 2 resources D after allocation.

Write a C/ C++ code to do deadlock avoidance using banker’s algorithm:

* Compute what each process might still request and fill this in under the column Need.
* Is the system in a safe state? Why or why not?
* Is the system deadlocked? Why or why not?



**CODE:**

**#include <stdio.h>**

**int current[5][5], maximum\_claim[5][5], available[5];**

**int allocation[5] = {0, 0, 0, 0, 0};**

**int max\_res[5], running[5], safe = 0;**

**int counter = 0, i, j, exec, resources, processes, k = 1;**

**int main(){**

**printf("\nEnter number of processes: ");**

**scanf("%d", &processes);**

**for (i = 0; i < processes; i++0){**

**running[i] = 1;**

**counter++;**

**}**

**printf("\nEnter number of resources: ");**

**scanf("%d", &resources);**

**2**

**printf("\nEnter Claim Vector:");**

**for (i = 0; i < resources; i++)**

**scanf("%d", &max\_res[i]);**

**printf("\nEnter Allocated Resource Table:\n");**

**for (i = 0; i < processes; i++)**

**for(j = 0; j < resources; j++)**

**scanf("%d", &current[i][j]);**

**printf("\nEnter Maximum Claim Table:\n");**

**for (i = 0; i < processes; i++)**

**for(j = 0; j < resources; j++)**

**scanf("%d", &maximum\_claim[i][j]);**

**printf("\nThe Claim Vector is: ");**

**for (i = 0; i < resources; i++)**

**printf("\t%d", max\_res[i]);**

**printf("\nThe Allocated Resource Table:\n");**

**for (i = 0; i < processes; i++){**

**for (j = 0; j < resources; j++)**

**printf("\t%d", current[i][j]);**

**printf("\n");**

**}**

**printf("\nThe Maximum Claim Table:\n");**

**for (i = 0; i < processes; i++){**

**for (j = 0; j < resources; j++)**

**printf("\t%d", maximum\_claim[i][j]);**

**printf("\n");**

**}**

**for (i = 0; i < processes; i++)**

**for (j = 0; j < resources; j++)**

**allocation[j] += current[i][j];**

**printf("\nAllocated resources:");**

**for (i = 0; i < resources; i++)**

**printf("\t%d", allocation[i]);**

**for (i = 0; i < resources; i++)**

**available[i] = max\_res[i] - allocation[i];**

**printf("\nAvailable resources:");**

**for (i = 0; i < resources; i++)**

**printf("\t%d", available[i]);**

**printf("\n");**

**while (counter != 0){**

**safe = 0;**

**for (i = 0; i < processes; i++){**

**if (running[i]){**

**exec = 1;**

**for (j = 0; j < resources; j++){**

**if (maximum\_claim[i][j] - current[i][j] > available[j]){**

**exec = 0;**

**break;**

**}**

**}**

**if (exec){**

**printf("\nProcess%d is executing\n", i + 1);**

**running[i] = 0;**

**counter--;**

**safe = 1;**

**3**

**for (j = 0; j < resources; j++)**

**available[j] += current[i][j];**

**break;**

**}**

**}**

**}**

**if (!safe){**

**printf("\nThe processes are in unsafe state.\n");**

**break;**

**}**

**else{**

**printf("\nThe process is in safe state");**

**printf("\nAvailable vector:");**

**for (i = 0; i < resources; i++)**

**printf("\t%d", available[i]);**

**printf("\n");**

**}**

**}**

**return 0;**

**}**

**OUTPUT:**

**4**

