Experiment Number: 9

# Problem Statement: Write a program to check whether given system is in safe state or not using Banker’s Deadlock Avoidance algorithm.

NAME: Manoj Dhanraj MuleROLLNO:71

CLASS: IT-B BATCH: B3

DATE OF PERFORMANCE: 26/09/2024

#include <stdio.h> #include <stdbool.h>

int main() {

int n, m, i, j, k;

// n = Number of processes, m = Number of resource types printf("Enter the number of processes: ");

scanf("%d", &n);

printf("Enter the number of resource types: "); scanf("%d", &m);

int allocation[n][m], max[n][m], need[n][m], available[m]; bool finish[n];

int safeSequence[n]; int count = 0;

// Input the Allocation Matrix

printf("Enter the Allocation Matrix (%d x %d):\n", n, m); for (i = 0; i < n; i++) {

for (j = 0; j < m; j++) { scanf("%d", &allocation[i][j]);

}

}

// Input the Maximum Matrix

printf("Enter the Maximum Matrix (%d x %d):\n", n, m); for (i = 0; i < n; i++) {

for (j = 0; j < m; j++) { scanf("%d", &max[i][j]);

}

}

// Input the Available Resources

printf("Enter the Available Resources (1 x %d):\n", m); for (i = 0; i < m; i++) {

scanf("%d", &available[i]);

}

// Calculate the Need Matrix (Need = Max - Allocation) for (i = 0; i < n; i++) {

for (j = 0; j < m; j++) {

need[i][j] = max[i][j] - allocation[i][j];

}

}

// Initialize Finish array to false for (i = 0; i < n; i++) {

finish[i] = false;

}

// Banker's Algorithm: Find a safe sequence while (count < n) {

bool found = false; for (i = 0; i < n; i++) {

if (!finish[i]) { // Check if the process is unfinished bool canAllocate = true;

for (j = 0; j < m; j++) {

if (need[i][j] > available[j]) { // If need is greater than available, skip this process canAllocate = false;

break;

}

}

if (canAllocate) { // If resources can be allocated for (k = 0; k < m; k++) {

available[k] += allocation[i][k]; // Release the resources

}

safeSequence[count++] = i; finish[i] = true;

found = true;

}

}

}

if (!found) { // If no process could be found in this pass, system is not in a safe state printf("The system is not in a safe state.\n");

return 0;

}

}

// If we found a safe sequence

printf("The system is in a safe state.\nSafe sequence is: "); for (i = 0; i < n; i++) {

printf("%d ", safeSequence[i]);

}

printf("\n");

return 0;

}

# Output:

# manoj@manoj-VirtualBox:~/oslab$ gcc banker.c -o banker

# manoj@manoj-VirtualBox:~/oslab$ ./banker

# 5

# Enter the number of resource types: 3

# Enter the Allocation Matrix (5 x 3):

# 0 1 0

# 2 0 0

# 3 0 2

# 2 1 1

# 0 0 2

# Enter the Maximum Matrix (5 x 3):

# 7 5 3

# 3 2 2

# 9 0 2

# 2 2 2

# 4 3 3

# Enter the Available Resources (1 x 3):

# 3 3 2

# The system is in a safe state.

# Safe sequence is: 1 3 4 0 2

# manoj@manoj-VirtualBox:~/oslab$

