Lab : 5

1. Bankers algorithm

#include <stdio.h>

#include <stdbool.h>

int main() {

int n, m;

printf("Enter number of processes and resources:\n");

scanf("%d %d", &n, &m);

int alloc[n][m], max[n][m], avail[m];

printf("Enter allocation matrix:\n");

for (int i = 0; i < n; i++)

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

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

printf("Enter max matrix:\n");

for (int i = 0; i < n; i++)

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

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

printf("Enter available matrix:\n");

for (int i = 0; i < m; i++)

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

int need[n][m];

for (int i = 0; i < n; i++)

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

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

bool finish[n];

for (int i = 0; i < n; i++)

finish[i] = false;

int safeSeq[n];

int work[m];

for (int i = 0; i < m; i++)

work[i] = avail[i];

int count = 0;

while (count < n) {

bool found = false;

for (int p = 0; p < n; p++) {

if (!finish[p]) {

int j;

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

if (need[p][j] > work[j])

break;

if (j == m) {

for (int k = 0; k < m; k++)

work[k] += alloc[p][k];

safeSeq[count++] = p;

finish[p] = true;

found = true;

}

}

}

if (!found) {

printf("System is not in a safe state.\n");

return 0;

}

}

printf("System is in safe state.\nSafe sequence is: ");

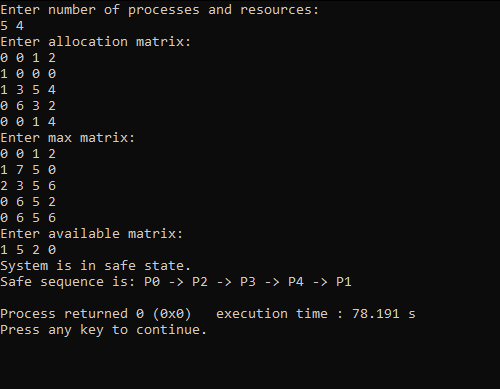
for (int i = 0; i < n; i++)

printf("P%d%s", safeSeq[i], (i == n - 1) ? "\n" : " -> ");

return 0;

}

Output:



1. Deadlock detection.

#include <stdio.h>

#include <stdbool.h>

int main() {

int n, m;

printf("Enter number of processes and number of resources:\n");

scanf("%d %d", &n, &m);

int alloc[n][m], request[n][m], avail[m];

printf("Enter Allocation Matrix (%d x %d):\n", n, m);

for (int i = 0; i < n; i++)

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

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

printf("Enter Request Matrix (%d x %d):\n", n, m);

for (int i = 0; i < n; i++)

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

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

printf("Enter Available Resources (%d values):\n", m);

for (int i = 0; i < m; i++)

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

int work[m];

for (int i = 0; i < m; i++)

work[i] = avail[i];

bool finish[n];

for (int i = 0; i < n; i++) {

bool hasAllocation = false;

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

if (alloc[i][j] != 0) {

hasAllocation = true;

break;

}

}

finish[i] = hasAllocation ? false : true;

}

while (true) {

bool progress = false;

for (int i = 0; i < n; i++) {

if (!finish[i]) {

bool canGrant = true;

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

if (request[i][j] > work[j]) {

canGrant = false;

break;

}

}

if (canGrant) {

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

work[j] += alloc[i][j];

finish[i] = true;

progress = true;

}

}

}

if (!progress)

break;

}

printf("\nDeadlock Detection Result:\n");

bool deadlock = false;

for (int i = 0; i < n; i++) {

if (!finish[i]) {

printf("Process P%d is DEADLOCKED\n", i);

deadlock = true;

} else {

printf("Process P%d is NOT deadlocked\n", i);

}

}

if (!deadlock)

printf("\nNo deadlock detected in the system.\n");

return 0;

}

