OPERATING SYSTEM LAB

Name: Thorat Amey Arun Reg No.: 23MCS1004

LAB EXPERIMENT 8
Deadlock Avoidance Algorithm

Write a C/ C++ code to do deadlock avoidance using banker's algorithm to find Whether the system is in a safe state or not?

Sample input only May run with other input:

Consider the following snapshot of a system at time t0 in which four resources A, B, C and D are available. The system totally contains 6 instances of A, 4 of resource B, 4 of resource C, 2 resources D before allocation.

	Allocation				*Max*				*Need*				*Available*			
	A	*B*	*C*	*D*	*A*	*B*	*C*	*D*	*A*	*B*	*C*	*D*	*A*	*B*	*C*	*D*
P_0	2	0	1	1	3	2	1	1					6	4	4	2
P_1	1	1	0	0	1	2	0	2								
P_2	1	0	1	0	3	2	1	0								
P_3	0	1	0	1	2	1	0	1								

If P0 request < 1 1 0 0> at time t1, will the request be granted? Apply resource request algorithm and find the Safe sequence.

Note:
Write as a single program to run safety algorithm as well resource request algorithm. Make it as generic algorithm by taking user input than hard coding the matrix value.

Program:

```
#include <stdio.h>
#define MAX_PROCESSES 10
#define MAX_RESOURCES 10
int processes, resources;
int allocation[MAX_PROCESSES][MAX_RESOURCES];
int max[MAX_PROCESSES][MAX_RESOURCES];
int need[MAX_PROCESSES][MAX_RESOURCES];
int available[MAX_RESOURCES];
void inputData() {
printf("Enter the number of processes: ");
scanf("%d", &processes);
```

```
printf("Enter the number of resources: ");
scanf("%d", &resources);
printf("Enter the allocation matrix:\n");
for (int i = 0; i < processes; i++)
for (int j = 0; j < resources; j++)
scanf("%d", &allocation[i][j]);
printf("Enter the maximum matrix:\n");
for (int i = 0; i < processes; i++)
for (int j = 0; j < resources; j++) {
scanf("%d", &max[i][j]);
need[i][j] = max[i][j] - allocation[i][j];
}
printf("Enter the available resources: ");
for (int i = 0; i < resources; i++)
scanf("%d", &available[i]);
}

int checkSafety() {
int work[MAX_RESOURCES];
int finish[MAX_PROCESSES] = {0};
int safeSequence[MAX_PROCESSES];
int safeCount = 0;
for (int i = 0; i < resources; i++) {
work[i] = available[i];
}
while (safeCount < processes) {
int found = 0;
for (int i = 0; i < processes; i++) {
if (!finish[i]) {
int canFinish = 1;
for (int j = 0; j < resources; j++) {
if (need[i][j] > work[j]) {
canFinish = 0;
break;
}
}
if (canFinish) {
for (int j = 0; j < resources; j++) {
work[j] += allocation[i][j];
```

```
}
finish[i] = 1;
safeSequence[safeCount++] = i;
found = 1;
}
}
}
if (!found) {
return 0; // No safe sequence found
}
}
printf("Safe sequence: ");
for (int i = 0; i < processes; i++) {
printf("P%d -> ", safeSequence[i]);
}
printf("\n");
return 1; // Safe sequence found
}

int requestResource(int pid, int request[]) {
for (int i = 0; i < resources; i++) {
if (request[i] > need[pid][i] || request[i] > available[i]) {
return 0; // Request cannot be granted
}
}
// Temporarily allocate resources
for (int i = 0; i < resources; i++) {
allocation[pid][i] += request[i];
need[pid][i] -= request[i];
available[i] -= request[i];
}
if (checkSafety()) {
return 1; // Request granted
} else {
// Rollback changes
for (int i = 0; i < resources; i++) {
allocation[pid][i] -= request[i];
need[pid][i] += request[i];
available[i] += request[i];
```

```
}
return 0; // Request denied
}
}

int main() {
inputData();
if (checkSafety()) {
printf("System is in a safe state.\n");
}
else {
printf("System is in an unsafe state.\n");
}
int pid, request[MAX_RESOURCES];
printf("Enter the process requesting resources: ");
scanf("%d", &pid);
printf("Enter the resource request for process P%d: ", pid);
for (int i = 0; i < resources; i++) {
scanf("%d", &request[i]);
}
if (requestResource(pid, request)) {
printf("Request granted.\n");
}
else {
printf("Request denied. Granting the request would lead to an unsafe state.\n");
}
return 0;
}
```

Output:

OPERATING SYSTEM LAB

Name: Thorat Amey Arun Reg No.: 23MCS1004

```
student1@student1-VirtualBox:~$ cd Desktop
student1@student1-VirtualBox:~/Desktop$ gcc -o lab8 lab8.c
student1@student1-VirtualBox:~/Desktop$ ./lab8
Enter the number of processes: 4
Enter the number of resources: 4
Enter the allocation matrix:
2 0 1 1
1 1 0 0
1 0 1 0
0 1 0 1
Enter the maximum matrix:
3 2 1 1
1 2 0 2
3 2 1 0
2 1 0 1
Enter the available resources: 6 4 4 2
Safe sequence: P0 -> P1 -> P2 -> P3 ->
System is in a safe state.
Enter the process requesting resources: 0
Enter the resource request for process P0: 1 1 0 0
Safe sequence: P0 -> P1 -> P2 -> P3 ->
Request granted.
student1@student1-VirtualBox:~/Desktop$
```

```
student1@student1-VirtualBox:~/Desktop$ ./lab8
Enter the number of processes: 4
Enter the number of resources: 4
Enter the allocation matrix:
2 0 1 1
1 1 0 0
1 0 1 0
0 1 0 1
Enter the maximum matrix:
3 2 1 1
1 2 0 2
3 2 1 0
2 1 0 1
Enter the available resources: 0 4 4 2
Safe sequence: P1 -> P0 -> P2 -> P3 ->
System is in a safe state.
Enter the process requesting resources: 1
Enter the resource request for process P1: 2 0 0 0
Request denied. Granting the request would lead to an unsafe state.
student1@student1-VirtualBox:~/Desktop$
```