“Operating System” Experimental Guide

Experiment 4: Banker Algorithm for Avoiding Process Deadlocks

Experimental Hours: 6 hours recommended

Experimental Platform: Visual C++ or Visual Studio

[Objective]

Through this experiment, the understanding of the process deadlock is deepened, and the allocation of process resources, the method of judging the safe state, and the avoidance of deadlocks are grasped.

[Content]

Problem Description:

To design a process simulating the work process of banker algorithm to avoid process deadlocks. Suppose there are n processes $P_1,...,P_n$ in the system, there are m resources type with the resource number $R_1,... R_m$, respectively. At T_0 time, the process Pi is allocated resource type j, which is represented by $Allocation_{ij}$, it also needs the number of resource type j as $Need_{ij}$, the system currently has the remaining the number of resource type j as $Work_j$. At T_1 time, there is a process k makes a further request $Request_k$ for the resources. Using banker algorithm to judge whether the request can be granted or not, so as to avoid a deadlock.

The program requirements are as follows:

1) For the time T_0, determine whether the current state is safe, if it is safe, give a security sequence.

2) For the next time T_1, a certain process P_k will make a request $Request_k$ $(R1, ..., Rm)$, then judge that whether the resources can be allocated to the P_k process as it requests.

3) Input the following information, or prepare it in a file. If using a file way, input the file name, and output the information in the file on the screen as follows.

(1) The number of processes n, the number of resource types m, and the number of resources for each resource type, for example:

```
The number of processes: 5
The number of resource types: 3
The numbers of the resources for each resource type: 10, 5, 7
```

(2) The current state of the resource allocation in T_0, each process’s max need for the resources, and the rest available resources, for example:

	Allocation	Max	Available
	1 2 3	1 2 3	1 2 3
P_0	0 1 0	7 5 3	3 3 2
P_1	2 0 0	3 2 2	
P_2	3 0 2	9 0 2	
P_3	2 1 1	2 2 2	
P_4	0 0 2	4 3 3	

(3) For T_1, the number of the process that makes the request, and specific resource number of the request, for example:

Who makes a further request (the number of the process): 2
The request: 1,0,2

4) Output:

(1) The intermediate results（中间结果）of the safety algorithm, for example:

	Allocation	Max	Available	Need	work	Finish[i]
	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	
			5 3 2		5 3 2	
P_0	0 1 0	7 5 3		7 4 3		false
P_1	2 0 0	3 2 2		1 2 2		true
P_2	3 0 2	9 0 2		6 0 0		false
P_3	2 1 1	2 2 2		0 1 1		false
P_4	0 0 2	4 3 3		4 3 1		false

Sequence：P1

(2) If the state is secure, outputs the security sequence of processes.

(3) The conclusion whether the request can be granted.

Implementation Tips:

- Realize two algorithms for the banker's method in a program.

1. A safety algorithm that tests the current sate of the system and sees if there is a deadlock or not.

2. A resource-request algorithm that grants a request for a process and allocates the requested resources to the process.

3. In the main function, to complete:

(1) Input the information needed and the current state of the system.

(2) Display the state of the system in T_0 time.

(3) Invoke a safety algorithm to see if the current state is safe or not, and display the intermediate results.

(4) Output the judgement of the state, and if the state is safe, output the security sequence of the processes.

(5) Input the requests in T_1 time.

(6) Invoke a resource-request algorithm to allocate the resource requested by the

process.

(7) Display the state of the system in T_I time.

(8) Invoke a safety algorithm to see if the current state is safe or not, and display the intermediate results（中间结果）.

(9) Output the judgement of the state and the conclusion whether the request can be granted or not. If the state is safe, output the security sequence of the processes.

[Requirements]:

According to the specific experimental requirements, complete the experimental report, including:

(1) Requirements analysis

- The task of programming with ambiguous statements emphasizes what the program should do. And clearly stipulates:
- The form of input and the range of input values;
- The form of output;
- The functions that the procedure can achieve;
- Test data: including correct input and output results, and incorrect input and output results.

(2) Outline design

Describes the definition of all the abstract data types used in this program, the flow of the main program, and the hierarchical (invocation) relationship between the program modules.

(3) **Detailed design**:Realize the specific algorithm of program module.

(4) Debug analysis

The recommended contents may include:

- How to solve the problems encountered in the debugging process and the analysis of the design and implementation;
- Spatio-temporal analysis of the algorithm (including the analysis of time and space complexity of basic operations and other algorithms) and improvement ideas;
- Experience.

(5) **User instructions:** Explains how to use the program you write and lists each step in detail.

(6) **Test results:**

List your test results, including input and output. The test data here should be complete and strict. Attach the screenshot of the program results.

(7) **Appendix**. Annotated source program.