

Project 5: Bank Algorithm

Hao Wu

5140219226

May 16, 2017

Instructor: Ling Gong

1. Object:

For this project, you will write a multithreaded program that implements the banker's algorithm discussed in Section 7.5.3. Several customers request and release resources from the bank. The banker will grant a request only if it leaves the system in a safe state. A request that leaves the system in an unsafe state will be denied. This programming assignment combines three separate topics: (1) multithreading, (2) preventing race conditions, and (3) deadlock avoidance.

2. Screen Cut:

By typing the available source and initial situation of each customer in 1.txt, we can simulate the banker's algorithm using multi-threading, preventing race conditions and deadlock avoidance. As figure1 shows, we can try to allocate source to one specific customer, if the allocation is reasonable, the console will return "Approved". Anytime customer obtains all source it needs, it will release all the source and quit. Besides, like figure2 shows, we can choose to let one customer release some source it occupies. The error processing are presents in figure3, any source allocation and release which beyond really needed or occupied will be denied.

```

customerNumber  maximun allocation      need      available
Customer 0:      3 2 2      0 0 0      3 2 2      10 5 7
Customer 1:      9 0 2      0 0 0      9 0 2
Customer 2:      2 2 2      0 0 0      2 2 2
Customer 3:      4 3 3      0 0 0      4 3 3
RQ 0 1 1 1
*1*
*1*
*1*
Approved
*
customerNumber  maximun allocation      need      available
Customer 0:      3 2 2      1 1 1      2 1 1      9 4 6
Customer 1:      9 0 2      0 0 0      9 0 2
Customer 2:      2 2 2      0 0 0      2 2 2
Customer 3:      4 3 3      0 0 0      4 3 3
RL 0 1 1 1
*1*
*1*
*1*
*
customerNumber  maximun allocation      need      available
Customer 0:      3 2 2      0 0 0      3 2 2      10 5 7
Customer 1:      9 0 2      0 0 0      9 0 2
Customer 2:      2 2 2      0 0 0      2 2 2
Customer 3:      4 3 3      0 0 0      4 3 3

```

Fig2 Sources Release

The image consists of two screenshots of a terminal window. The top screenshot shows a table of resource allocation for three customers and a request that exceeds available resources, leading to an error message. The bottom screenshot shows the same table after the request has been adjusted to match available resources, resulting in an 'Approved' status.

Top Screenshot:

customerNumber	maximum allocation	need	available	
Customer 0:	3 2 2	0 0 0	3 2 2	10 5 7
Customer 1:	0 2	0 0 0	9 0 2	
Customer 2:	2 2 2	0 0 0	2 2 2	
Customer 3:	4 3 3	0 0 0	4 3 3	

Request: RQ 0 8 1 0

Error message: "The release amount of resources surpasses the resources what the process possesses! So we just release the resources that the process is allocated to!"

Bottom Screenshot:

customerNumber	maximum allocation	need	available
Customer 0:	9 0 2	0 0 0	9 0 2
Customer 1:	2 2 2	0 0 0	2 2 2
Customer 2:	4 3 3	0 0 0	4 3 3

Request: RQ 0 8 1 0

Status: Approved

Adjusted table:

customerNumber	maximum allocation	need	available
Customer 0:	9 0 2	8 0 0	1 0 2
Customer 1:	2 2 2	0 0 0	2 2 2
Customer 2:	4 3 3	0 0 0	4 3 3

Fig3 Error processing