BLG 312E COMPUTER OPERATING SYSTEMS

HOMEWORK 3

Banker's Algorithm

Deadline: 26.05.2024 23:59

The Banker's algorithm is a resource allocation and deadlock avoidance algorithm used in operating systems. It is designed to prevent deadlocks by determining whether a particular resource allocation request from processes can be granted safely without causing deadlock.

The Banker's algorithm works by considering the available resources, the resource needs of each process, and the already assigned resources. It performs a safety check to determine if granting the requested resources to a process would leave the system in a safe state, meaning that all processes can eventually complete their execution without getting into a deadlock situation.

When a new process arrives to the system, it has to specify how many instances of each resource type it will require. The total amount cannot exceed the entire number of system resources. When a user demands a set of resources, the system must decide if allocating these resources would keep the system secure. If it does, the resources are allocated; otherwise, the process must wait for another process to release sufficient resources.

In a system, there are five separate sources (R1, R2, R3, R4, and R5). The Table 1 shows the total number of resources available in the system. These resources are shared by five processes (P1, P2, P3, P4, and P5). Already allocated resources are listed in the Table 2 The resource allocation requests that the processes require to continue execution are listed in the Table 3.

R1	R2	R3	R4	R5
5	7	10	2	6

Table 1: All Resources

	R1	R2	R3	R4	R5
P1	3	0	1	1	0
P2	1	1	0	0	0
P3	0	3	0	0	0
P4	1	0	0	0	0
P5	0	1	4	0	0

Table 2: Allocations

	R1	R2	R3	R4	R5
P1	0	1	7	0	1
P2	0	0	1	0	3
P3	2	2	0	0	1
P4	1	0	1	0	2
P5	3	1	0	1	1

Table 3: Requests

Implementation

- You will be given three input files: resources.txt, allocations.txt, requests.txt. You should read the necessary information from input files. **Your code will be tested with different input files as well.**
- First implement your code and print the all necessary information **for each process respectively** as shown in Figure 1: resource allocations, and resource request (All numbers are given as 0 to illustrate the output example).

```
Information for process : P1:
Allocated recources : R1:0 R2:0 R3:0 R4:0 R5:0
Resource request :  R1:0 R2:0 R3:0 R4:0 R5:0
```

Figure 1: Necessary Informations

- State in what order processes run and terminate based on this information. If there is a deadlock in the system, explain which processes are involved and the causes for the deadlock as seen in Figure 2. Process names are given randomly as an example. Use process names P1, P2, P3 , P4 and P5 in your outputs.

```
Running order for processes: PX PW PZ
There is a deadlock. PQ and  PS are the cause of deadlock.
```

Figure 2: Output Format

Submission

- Submit a C code, makefile and 2 page report for this assignment. In your report, explain your answer and calculations step by step.
- Prepare your reports in LaTeX format to get full marks. You can use the following template
- Any form of plagiarism will not be tolerated. You must solve each question by yourself.
- You must implement your solution in the C programming language and it should work on a linux environment.
- For any question: erzurumluoglu18@itu.edu.tr