BLG 312E Operating Systems Homework 3 Report

Özkan Gezmiş 150200033

May 26, 2024

1 Introduction

The Banker's Algorithm is a resource allocation and deadlock avoidance algorithm used in operating systems. It ensures that resource requests are granted only if they leave the system in a safe state.

This report provides an overview of the implementation of the Banker's Algorithm in C, including the compilation and execution process using a Makefile, and demonstrates the algorithm's ability to detect deadlock through calculations.

2 Makefile

The Makefile automates the compilation process and ensures that all dependencies are correctly handled. The provided Makefile includes targets for both compiling the C program and executing it with the necessary input files.

```
banker: hw3_banker.c
    gcc -o banker hw3_banker.c

run:
    ./banker allocations.txt resources.txt requests.txt
```

3 Resource Allocation and Request Matrices

3.1 Requests Matrix

	$R1$	$R2$	$R3$	$R4$	$R5$
$P1$	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

3.2 Allocations Matrix

	$R1$	$R2$	$R3$	$R4$	$R5$
$P1$	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

3.3 Resources Matrix

$R1$	$R2$	$R3$	$R4$	$R5$
5	7	10	2	6

4 Deadlock Detection and Calculation

The Banker's Algorithm performs the following steps to determine if the system is in a safe state and to detect any potential deadlocks:

4.1 Initialization

4.1.1 Available Resources Calculation

The total available resources are calculated by subtracting the allocated resources from the total resources.

$$\text{Available} = \text{Resources} - \text{Allocations}$$

$$\text{Available} = \begin{bmatrix} 5 & 7 & 10 & 2 & 6 \end{bmatrix} - \begin{bmatrix} 5 & 5 & 5 & 1 & 0 \end{bmatrix}$$

$$\text{Available} = \begin{bmatrix} 0 & 2 & 5 & 1 & 6 \end{bmatrix}$$

4.2 Calculation

- P1 cannot run since there is not enough available R3.
- P2 can run since it doesn't request more than available resources. Execute P2 and mark it as finished, after it finishes it releases the allocated resources. New available resources matrix is: Available $= \begin{bmatrix} 1 & 3 & 5 & 1 & 6 \end{bmatrix}$
- P3 cannot run since there is not enough available R1.
- P4 can run since it doesn't request more than available resources. Execute P4 and mark it as finished, after it finishes it releases the allocated resources. New available resources matrix is: Available $= \begin{bmatrix} 2 & 3 & 5 & 1 & 6 \end{bmatrix}$
- P5 cannot run since there is not enough available R1.
- Start to iterate again, go over unfinished process.

- P1 still cannot run since there is not enough available R3.
- P3 can run now since it doesn't request more than available resources. Execute P4 and mark it as finished, after it finishes it releases the allocated resources. New available resources matrix is: Available = $\begin{bmatrix} 2 & 6 & 5 & 1 & 6 \end{bmatrix}$
- P5 still cannot run since there is not enough available R1.
- Again check P1 and P5. Since they cannot be executed algorithm ends. That means there is a deadlock.

Thus, the execution order for processes $[P2, P4, P3]$. Since P1 and P5 are unfinished, there is a deadlock.

5 Conclusion

The implementation of the Banker's Algorithm in C successfully reads the resource allocation, request, and resource matrices from files, calculates the available resources, and determines whether the system is in a safe state or if a deadlock occurs. The provided Makefile facilitates the compilation and execution of the program.