Danny Hido

11/20/2025

CS 33211 Operating Systems

Assignment Two Documentation

Overview:

This project implements the Banker's Algorithm. The Bankers' Algorithm is a deadlock-avoidance method used in operating systems to ensure resource allocation to processes never enters an unsafe state. The algorithm checks whether a system has enough available resources to satisfy the maximum needs of every process without causing deadlock.

This project implements a simulation of the Banker's Algorithm in C++ using a sample system state and processes. The file main.cpp reads input.txt and decides whether the system is safe or not, using the Bankers Algorithm.

Algorithm Explanation:

The Banker's Algorithm is a deadlock-avoidance algorithm that ensures a system never enters a state where processes become permanently blocked waiting for resources. The algorithm proactively checks whether future resource allocations can be completed safely. It is called the Banker's Algorithm because the analogy of a bank loan officer who only approves a loan if the bank can satisfy all customers' maximum possible withdrawals in some order is an accurate analogy of what the operating system is doing.

- The algorithm takes into account processes, resources of different types, instances, and the maximum resources needed for each process.
- In this project, we are given a matrix for processes and the systems allocation, max, and available resources. The need matrix Is calculated by Max – Allocation.

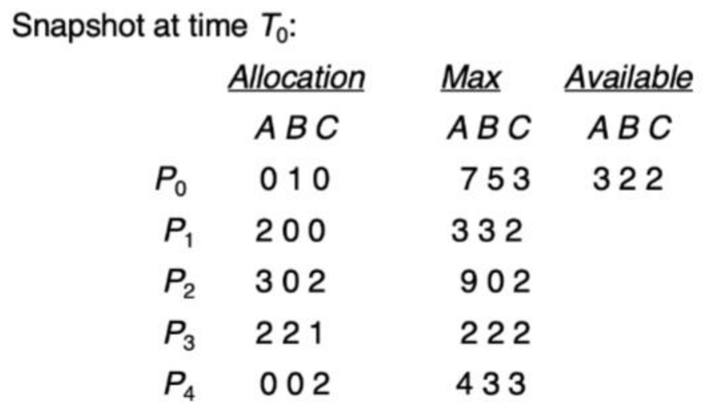
Algorithm Process:

1. Check if the request is valid. A process cannot request more than its maximum needs.
2. Check if resources are available. If there is not enough, the process must wait.

3. Allocate the resource *tentatively*. It is not certain whether it is safe yet.
4. Check for safety. If the Need is less than or equal to available. It simulates running that process. The simulation repeats for each process. If they all finish, the state is safe. If no runnable process remains, it is unsafe.
5. Approve or reject. If the state is deemed safe, the temporary allocation becomes permanent. If it is unsafe, the system reverts to the original safe state.

Project System Result:

After running the Banker's Algorithm on this system:



Snapshot at time T_0:

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

The system is confirmed to be in a **safe state**.

The safe sequence is P3, P4, P1, P2, and P0.

How to Run + Example Output:

This program is designed to run in a Linux environment with a g++ compiler.

For non-Linux users, Docker is a recommended virtual environment, and is where this project was built.

Download Docker first and follow these commands.

<u>Docker setup for non-linux users:</u>

```
docker run -it -v %cd%:/workspace ubuntu bash

apt update

cd /workspace
```

Once program is local, enter these commands to run:

<u>g++ installation</u>

```
apt install -y g++ make
```

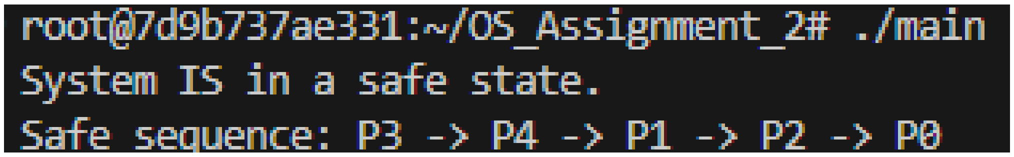
<u>Program compilation</u>

```
g++ main.cpp -o main
```

<u>Run Program</u>

```
./main
```

Output will be in the terminal as follows:



```
root@7d9b737ae331:~/OS_Assignment_2# ./main
System IS in a safe state.
Safe sequence: P3 -> P4 -> P1 -> P2 -> P0
```