

Description

The file uses bankers algorithm to determine if a list of processes and their memory allocations will cause deadlock

Algorithm Mechanics

The algorithm operates as follows:

1. Initialize with available resources and process data.
2. Repeatedly scan processes to find one whose needs are fully satisfiable by current available resources.
3. If found, simulate process completion: add its allocated resources back to the available pool and remove it from consideration.
4. Continue until all processes are scheduled (safe state) or no eligible process remains (unsafe state/deadlock).

Code Implementation Details

The C++ code utilizes standard libraries for efficiency:

- **Data Structures:**
 - **Filename** – the name of the input file
 - **Pcs** – a map of processes (keys) and their associated tuple (value)
 - The tuples contain two vectors, allocated memory, and maximum memory usage for each memory value
 - **Avl** – a vector containing the current running available memory values
- **Functions:**
 - **start():** Handles file input and validation, ensuring vector sizes match.
 - **bankersAlgo():** Core logic, iterating through processes and updating resources upon safe execution.
- **Error Handling:** Checks for file access and data consistency, terminating with errors if issues arise.

Input

The input file contains:

3 2 2

0 1 0 7 5 3

2 0 0 3 2 2

3 0 2 9 0 2

2 2 1 2 2 2

0 0 2 4 3 3

This represents 3 units of resource A, 2 of B, and 2 of C available, with five processes (P0 to P4) and their allocations/maxima.

Applying the Banker's Algorithm yields a safe sequence: P1, P3, P4, P0, P2. This order ensures no deadlock, as each process's needs are met sequentially without resource contention. For instance, P1 is selected first because its requirements align with available resources, freeing them for subsequent processes.

The system is in a safe state. The algorithm successfully identifies an execution order that avoids deadlock. If the input were altered to create an unsafe state, the program would detect and report a deadlock.