Project: Banker's Algorithm

CSCE 311 - Operating Systems

April 08, 2025

Purpose

In this project, you will experiment with the **Safety** and **Banker's** Algorithms. To that end, you will provide a management class which ensures that if a solution is possible, the threads are allowed to run.

Task

You must implement two applications:

- **BankersResourceManager:** Is a singleton-type class, i.e., only one instance will exist in the application, but it need not be implemented as an actual singleton pattern.

  It must expose the following public interface:

  + `BankersResourceManager::BankersResourceManager(const std::vector<std::size_t>&)`
    * The parameter represents the manager's beginning **Available** array, i.e., **Max** arrays must be the same size.
  + `void BankersResourceManager::AddMax(const std::vector<std::size_t>&)`
    * The parameter represents the n^{th} Thread's **Max** need.
  + `bool BankersResourceManager::Request(std::size_t,`
    `const std::vector<std::size_t>& request)`
    1. The first parameter is the index of the thread making request.
    2. The second parameter is the request.

    * The method returns `true` and makes the allocation if *safe* as per **Safety Algorithm**. It does not allocate and returns `false` if not *safe*.
    * On success, the function prints, for example,

      ```
      Thread 0 requested: { 1 1 0 }
        Need: { 2 3 1 }
        Available: { 4 5 2 }
        Safe. Request allocated. Order: {P0 P1 P2 P3 }
        Available: { 3 4 2 }
        Need: { 1 2 1 }
      ```

    * On fail, the function prints, for example,

      ```
      Thread 1 requested: { 0 3 5 }
        Need: { 1 5 5 }
        Available: { 3 4 1 }
        Not Available, request denied.
      ```

    + `void BankersResourceManager::Release(std::size_t id)`
        * The parameter represents the index of thread calling release.
        * On fail, the function prints, for example,
          `Thread 2 has released { 2 3 3 }`

- Your `::BankersResourceManager` **MUST USE** the `::ThreadMutex` and `::ThreadMutexGuard` from the `sync/include/thread_mutex.[h,cc]` header and source for request and release synchronization.

Execution:

All of the following should finish. Try increasing one Max value above Available to see one that cannot finish.

- `bankers-threads 7 "5 5 5" "2 3 4" "1 5 5" "2 3 3" "5 5 1"`
- `bankers-threads 13 "7 5 9" "1 1 9" "7 1 1" "1 1 9" "7 5 9"`
- `bankers-threads 29 "10 9 7" "10 9 7" "8 5 6" "7 6 7" "6 8 5"`

Deliverables

You must provide a zipped directory containing at least the following files:

```
proj3/
+-- include/
|   +-- bankers_resource_manager.h
|
+-- src/
|   +-- bankers_resource_manager.cc
|
+-- README.md
```

Note that any other files will be ignored. I have included a library (`<thread_mutex.h>`) for your use in synchronizing your code (in creating a critical section). It will be available for your code to compile against, but you will not be able to change it because we will not copy any files other than those mentioned above. Note that I will add logging info to the class and function to ensure you used them.

Grading Criteria

Build (20%)

`bankers_resource_manager.[h,cc]` successfully build with `bankers_thread.cc` and `thread_mutex.[h,cc]` to make `bankers-threads`.

Execution (30%)

Executable `bankers-threads` from above runs on input

`bankers-threads 0 "5 5 5" "1 1 1"`

Without crashing, but not necessarily printing a correct answer.

Correctness (40%)

Given input which allow all threads to finish and the program finishes: 20%.

Given input which does not allow all threads to finish and the program does not finish: 20%.

README.md (10%)

- Provide a reasonable `README.md` file, including build and execution instructions, file structure (only files you provide), and file contents for a free 10%.

Academic Integrity

By now, you know my stance on generative A.I.—it is a tool, not a panacea for all your C++ woes. I will ask three Generative A.I. for solutions and will include those in the stack of programs submitted for comparison. Given the breadth of freedom you have with solving this problem, I do not expect a great deal of code overlap.