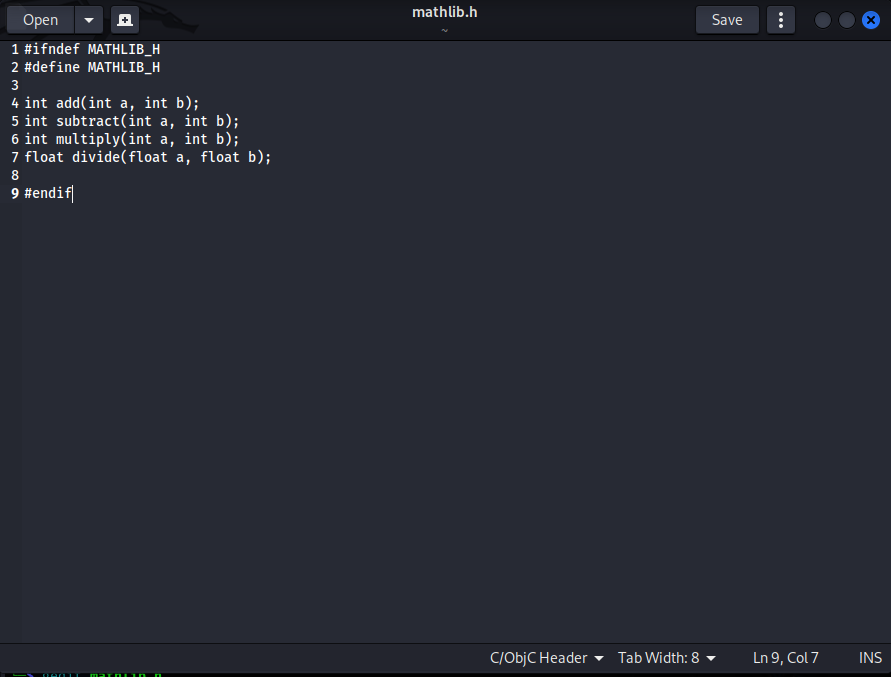
Lab-4 system security

Praneesh R V

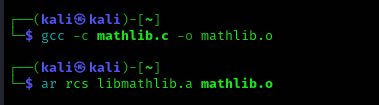
CB.SC.U4CYS23036

Creating static library:



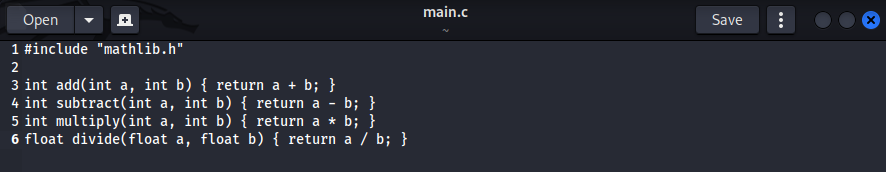
A screenshot of a computer

Description automatically generated



A black screen with green text

Description automatically generated



A screen shot of a computer

Description automatically generated

A computer code on a black background

Description automatically generated

Evaluation Questions

**1. Theoretical Understanding**

a Explain the key differences between static and dynamic linking

Ans.) Static Linking: The library code is copied into the executable at compile time.

Dynamic Linking: The library is loaded at runtime, enabling shared use across programs.

b. What are the advantages and disadvantages of each linking method?

Ans.) Static Linking Advantages: Independence from external files at runtime, simpler

deployment.

Static Linking Disadvantages: Larger executable size, need for recompilation after library updates.

Dynamic Linking Advantages: Smaller executables, runtime flexibility, shared memory usage.

Dynamic Linking Disadvantages: Dependency issues, potential version conflicts.

c. How does the memory usage differ between static and dynamic libraries?

Ans.) Static libraries increase memory usage as each program includes its own copy of the library.

Dynamic libraries optimize memory by sharing the same library across multiple programs.

**2. Technical Analysis**

a. Compare the file sizes of the executables created with static and dynamic linking

Ans.) Use ls -lh or dir to compare the file sizes of static and dynamically linked executables.

b. What happens if you try to run the dynamically linked program without setting LD\_LIBRARY\_PATH?

Ans.) The dynamically linked program will fail with a "library not found" error unless the library is in a standard path.

c. How would version conflicts arise in dynamic libraries, and how can they be managed?

Ans.) Conflicts arise if different programs require different versions of a shared library. These can be managed using tools like ldconfig or versioned library filenames (e.g., libname.so.1).

**3. Practical Application**

a. When would you choose static linking over dynamic linking in a real-world project?

Ans.) Use static linking for standalone applications with no runtime dependencies. Dynamic linking is preferable for modular, large-scale systems.

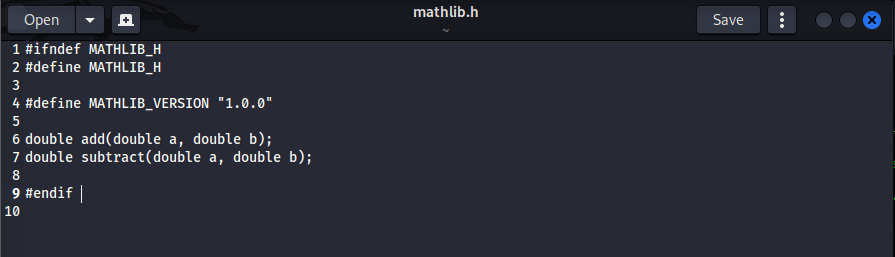
b. How does static linking affect application deployment compared to dynamic linking?

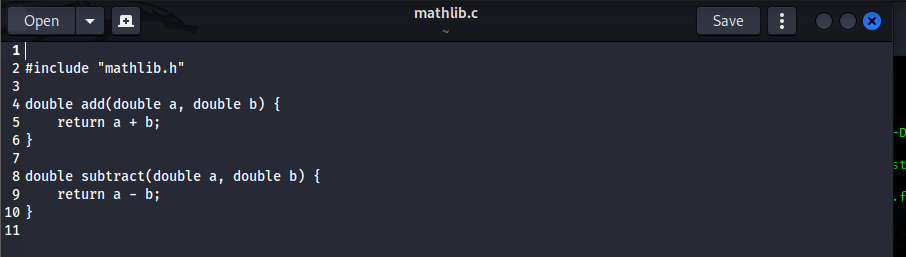
Ans.) Static linking simplifies deployment but increases the size of executables. Dynamic linking requires ensuring all required libraries are available at runtime.

**Advanced Challenge Tasks**

1. Library Version Management:

a.) Modify the math library to include version information

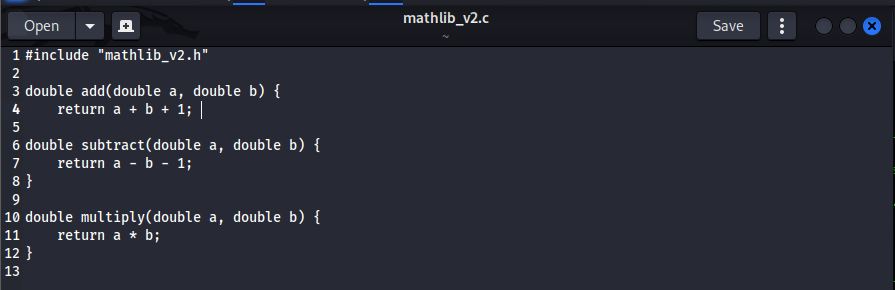




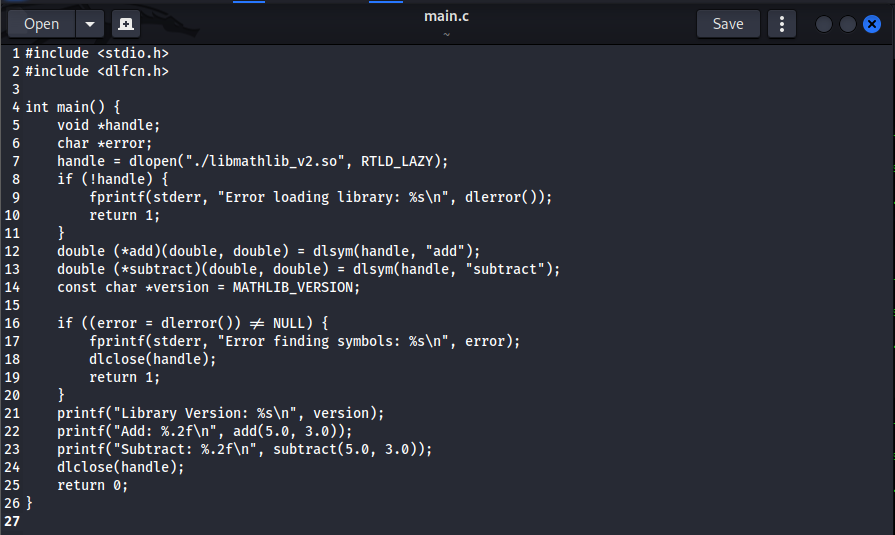
b.) Create multiple versions of the dynamic library with different implementations

A screenshot of a computer

Description automatically generated

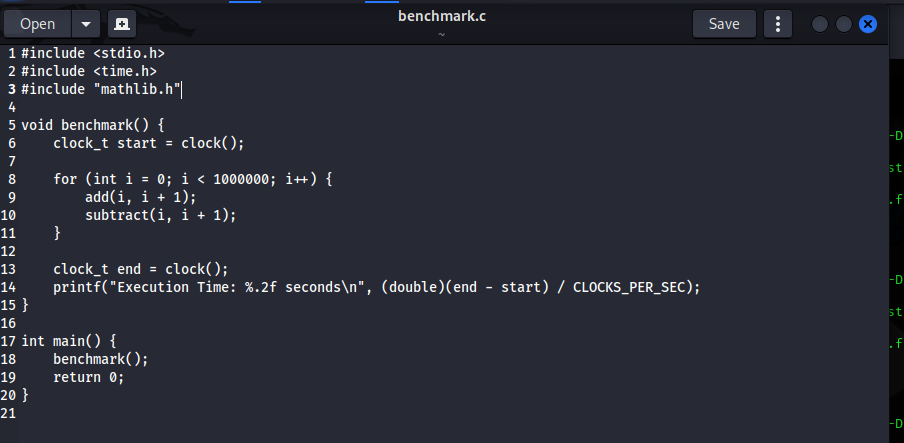


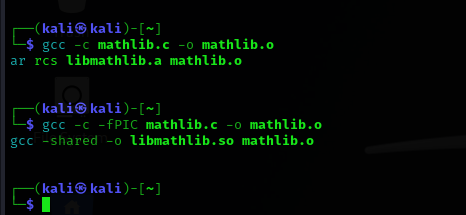
c.) Write a program that can work with different versions of the library

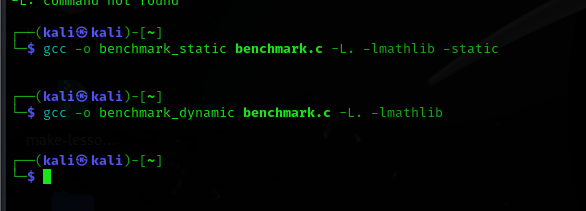


2. Performance Analysis:

a. Create a benchmark program that compares the performance of static vs. dynamic linking







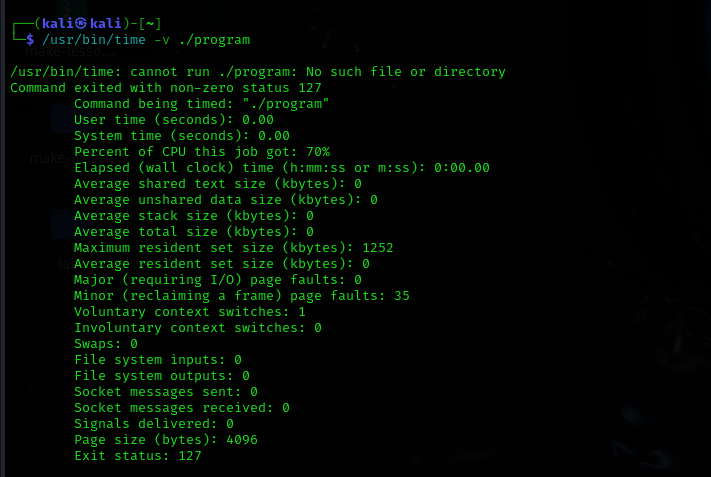
b. Measure and analyze:

i. Load time differences

A screen shot of a computer

Description automatically generated

ii. Memory usage



3. Complex Library Dependencies:

a. Create a library that depends on another library

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A computer code on a black background

Description automatically generated

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

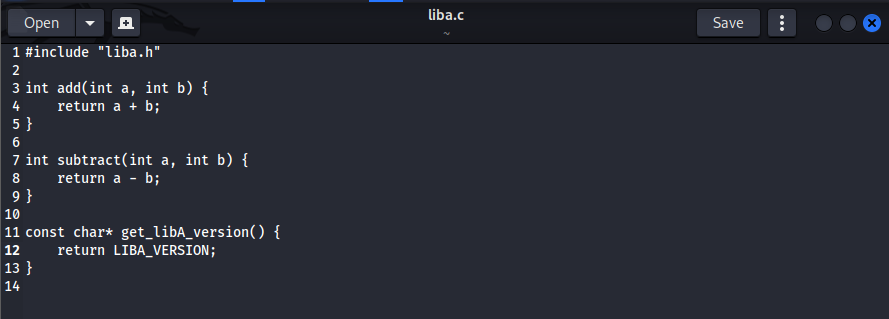
A computer code on a black background

Description automatically generated

b. Implement proper version checking and dependency management

A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated

c. Handle circular dependencies and resolve them