Export Chat to PDF - In-House C++ Library

How to Build and Use an In-House Library(c++)

Building and using an in-house library in C++ involves creating reusable code that can be shared across multiple projects. This process typically includes designing the library, compiling it into a static (`.a` or `.lib`) or dynamic (`.so` or `.dll`) library, and linking it with other programs.

Steps to Build and Use an In-House Library in C++

1. Plan the Library Structure

Before writing any code, decide:

- The functionality of your library.
- Whether it will be a static or dynamic library.
- The public API (functions/classes available to users).
- The internal implementation (hidden from users).

---

2. Write the Library Code

Create the necessary source and header files.

Example: `mathlib.h` **(Header File)**

cpp

```cpp
#ifndef MATHLIB_H
#define MATHLIB_H

namespace MathLib {
    int add(int a, int b);
    int subtract(int a, int b);
}
```

```
#endif // MATHLIB_H
```

Example: `mathlib.cpp` (Implementation File)

cpp

```cpp
#include "mathlib.h"

namespace MathLib {
    int add(int a, int b) {
        return a + b;
    }

    int subtract(int a, int b) {
        return a - b;
    }
}
```

---

3. Compile the Library

You can create either a **static library** or a **shared (dynamic) library**.

a) Building a Static Library

Compile object files and archive them.

sh

```sh
g++ -c mathlib.cpp -o mathlib.o
ar rcs libmathlib.a mathlib.o
```

This creates a static library (`libmathlib.a`).

b) Building a Shared Library

Compile the source file as a position-independent shared object.

sh

```
g++ -fPIC -c mathlib.cpp -o mathlib.o
g++ -shared -o libmathlib.so mathlib.o
```

This creates a shared library (`libmathlib.so`).

---

4. Use the Library in Your Project

a) Using a Static Library

Create a main program (`main.cpp`):

cpp

```cpp
#include <iostream>
#include "mathlib.h"

int main() {
    std::cout << "Sum: " << MathLib::add(5, 3) << std::endl;
    return 0;
}
```

Compile and link with the static library:

sh

```sh
g++ main.cpp -L. -lmathlib -o main
```

b) Using a Shared Library

For a shared library, use:

sh

```sh
g++ main.cpp -L. -lmathlib -o main
export LD_LIBRARY_PATH=.:$LD_LIBRARY_PATH
./main
```

---

5. Install the Library (Optional)

To use the library across projects, install it globally:

```sh
sudo cp libmathlib.a /usr/local/lib/
sudo cp mathlib.h /usr/local/include/
```

For shared libraries:

```sh
sudo cp libmathlib.so /usr/local/lib/
sudo ldconfig
```

Now, you can use `#include <mathlib.h>` in any project without specifying library paths.

Conclusion

By following these steps, you can create and use an in-house C++ library effectively. Whether it's static or dynamic, structuring your code well and using proper linking methods will ensure a smooth integration across projects. 🚀