clang 怎样生成动态库,以及怎样使用?

用 Clang 生成动态库（.so 在 Linux/macOS，.dylib 在 macOS，.dll 在 Windows）和使用它的完整指南如下（以 Linux 为例最常见，其他平台稍后说明）。

1. 生成动态库（共享库）

```bash
# 假设你有以下源文件
# math_lib.c
int add(int a, int b) { return a + b; }
int sub(int a, int b) { return a - b; }
```

编译生成动态库的正确命令（推荐方式）：

```bash
clang -fPIC -c math_lib.c -o math_lib.o          # 生成位置无关代码
clang -shared -o libmath.so math_lib.o           # 链接成动态库
```

或者一步完成（最常用）：

```bash
clang -fPIC -shared -o libmath.so math_lib.c
```

关键参数说明：

- `-fPIC`：生成位置无关代码（Position Independent Code），动态库必须加
- `-shared`：告诉链接器生成共享库
- `-o libmath.so`：输出的动态库名字，Linux 约定以 `lib` 开头
- 多个源文件时：

  ```bash
  clang -fPIC -shared -o libmath.so file1.c file2.c file3.c
  ```

如果有头文件（推荐做法）

```c
// math_lib.h
#ifndef MATH_LIB_H
#define MATH_LIB_H

#ifdef __cplusplus
extern "C" {
#endif

int add(int a, int b);
int sub(int a, int b);

#ifdef __cplusplus
}
#endif
#endif
```

编译时需要导出符号（Linux 默认都导出，macOS/Windows 需要特别处理）：

Linux（默认全部导出）：

```bash
clang -fPIC -shared -o libmath.so math_lib.c
```

如果想只导出特定函数（推荐，更安全）：

```bash
# 创建版本脚本或使用 -fvisibility=hidden + __attribute__((visibility("default")))
```

方法一：使用 `__attribute__((visibility("default")))`（推荐）

```c
// math_lib.c
__attribute__((visibility("default"))) int add(int a, int b) { return a + b; }
__attribute__((visibility("default"))) int sub(int a, int b) { return a - b; }
```

然后编译加参数：

```bash
clang -fPIC -shared -fvisibility=hidden -o libmath.so math_lib.c
```

2. 使用动态库

方法一：编译时直接链接（最简单）

```bash
clang main.c -L. -lmath -o app
# -L. 表示在当前目录找库
# -lmath 表示链接 libmath.so（自动加 lib 前缀和 .so 后缀）
```

运行前需要让系统能找到动态库：

```bash
# 方法1：临时设置
export LD_LIBRARY_PATH=.:$LD_LIBRARY_PATH
./app

# 方法2：把库复制到标准路径（不推荐）
sudo cp libmath.so /usr/lib/

# 方法3：安装时指定 rpath（推荐）
clang main.c -L. -lmath -Wl,-rpath,. -o app
# -Wl,-rpath,. 表示运行时也在当前目录搜索动态库
```

方法二：运行时动态加载（dlopen）

```c
// main.c
#include <stdio.h>
#include <dlfcn.h>

int main() {
    void* handle = dlopen("./libmath.so", RTLD_LAZY);
    if (!handle) {
        printf("加载失败: %s\n", dlerror());
        return 1;
    }

    // 获取函数指针
    int (*add)(int, int) = dlsym(handle, "add");
    int (*sub)(int, int) = dlsym(handle, "sub");

    printf("3 + 5 = %d\n", add(3, 5));
    printf("10 - 7 = %d\n", sub(10, 7));

    dlclose(handle);
    return 0;
}
```

编译：

```bash
clang main.c -ldl -o app_dl   # 需要链接 -ldl
./app_dl
```

3. 不同平台差异总结

平台	动态库后缀	生成命令示例	运行时环境变量
Linux	.so	`clang -fPIC -shared -o libxxx.so xxx.c`	`LD_LIBRARY_PATH`
macOS	.dylib	`clang -dynamiclib -o libxxx.dylib xxx.c`	`DYLD_LIBRARY_PATH`
Windows (MinGW/Clang)	.dll	`clang -shared -o xxx.dll xxx.c`	PATH

macOS 示例：

```bash
clang -fPIC -dynamiclib -o libmath.dylib math_lib.c
clang main.c -L. -lmath -o app
# macOS 默认会在当前目录找，基本不需要设置环境变量
```

Windows (clang + MinGW 或 clang-cl)：

```bash
clang --target=x86_64-w64-mingw32 -shared -o math.dll math_lib.c
# 使用时需要生成 import library (.lib) 或用 LoadLibrary
```

4. 完整推荐示例（Linux）

```bash
# 生成动态库（推荐方式）
clang -fPIC -shared -fvisibility=hidden -o libcalc.so calc.c

# 编译主程序，嵌入 rpath
clang main.c -L. -lcalc -Wl,-rpath,. -o app

# 直接运行
./app
```

这样生成的动态库和程序最干净、可移植性最好。

有任何其他需求（比如 CMake 集成、版本号、符号导出控制等）也可以继续问！

CMake集成动态库

静态库生成方法

Think Harder

Ask anything