1. 编译方式不同：

静态库是在编译时将库的代码打包到可执行程序中，因此生成的可执行程序包含了所有用到的库函数的代码。这样，当程序被调用时，需要使用哪些库函数就直接从可执行文件中取出来使用。因为代码打包进了可执行程序中，因此静态库的生成通常需要在代码的编译阶段进行。

动态库则是在运行时动态加载到程序中的，因此生成的可执行文件并不包含库函数的实现代码，而只是引用了动态库的接口。当程序调用到该库函数时，操作系统会将该函数从动态库文件中加载到内存中供程序运行使用。这样一来，程序的可执行文件会比静态库生成的可执行文件小很多。因为代码加载是在程序运行时进行的，所以动态库的链接通常是在程序运行之前进行。

2. 内存使用方式不同：

由于静态库的代码被打包进了可执行程序中，所以在程序运行时，静态库中的代码被复制到了程序使用的内存中，并一直驻留在内存中使用，因此不需要占用额外的内存空间。

而动态库的代码在程序运行时才会被加载到内存中，因此动态库的代码实现被复制进内存，会占用额外的内存空间。但是与静态库相比，动态库的内存使用方式具有更好的空间和性能优势，因为多个程序可以共享同一个动态库，而不需要重复加载相同的库文件，从而减少了系统的内存占用。

3. 更新和维护方式不同：

静态库的代码被打包成可执行程序的一部分，因此静态库的更新和维护需要重新进行编译和部署，才能让所有使用了该静态库的程序都能够得到更新的代码。

动态库可以独立于程序进行更新，因为动态库作为一个单独的文件存在于系统中，可以被多个程序共享。因此，当需要更新动态库时，只需要替换掉旧的动态库文件，不需要重新编译和部署所有使用了该动态库的程序。

因此，如果需要多个程序共享同一个库，或者需要较少的内存占用，则使用动态库可能更为合适。如果需要保持部署和更新的稳定性，则静态库可能更为适合。

————————————————

版权声明：本文为CSDN博主「m0\_56753665」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/m0\_56753665/article/details/130062999