**Library**

Library is a binary file which does not execute itself but provides services to other executable files.

There are two types of libraries :

1. Static Library
2. Dynamic Library / Shared Library

**Static Library**

A static library (or **archive**) contains code that is linked to users’ programs **at compile time**. The executable file generated keeps its own copy of the library code. When using static libraries, every executable must load the library code to the memory.

Windows : .lib (library file)

Linux : .a (archive file) / .sa (static archive)

Mac : .a (archive file) / .sa (static archive)

Advantages of Static Library:

1. Static Libraries are more robust and never has compatibility issues, since all the code is one in executable module.
2. The functions or libraries used very frequently in our program must be linked statically.

Ex- Standard Libraries.

Using static libraries lead to two obvious drawbacks:

1. Increasing the size of the application. The problem gets worse if the application contains multiple executables. You may end up keeping several copies of the same library.
2. Modifying the library code requires recompiling/linking of other parts of the application. This can be a pain for deploying purpose.

**Dynamic Library**

A **dynamic library** (also called a **shared library**) consists of routines that are loaded into your application at run time. When you compile a program that uses a dynamic library, the library does not become part of your executable -- it remains as a separate unit.

Windows : .dll (dynamic linked library)

Linux : .so (shared object)

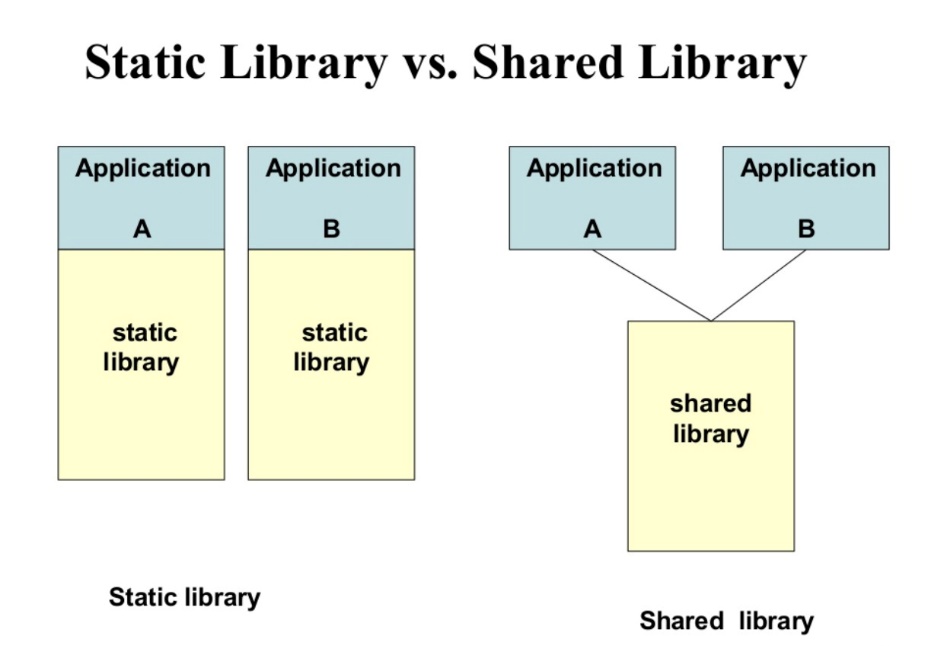
Mac : .dylib (dynamic library)

Advantages of dynamic libraries are:

1. Many programs can share one copy, which saves space.
2. Dynamic libraries can be upgraded to a newer version without replacing all the executables that use it.

Disadvantages of dynamic libraries are:

1. Dynamic Libraries are not as robust as Static Libraries. They may cause some compatibility issues at runtime.



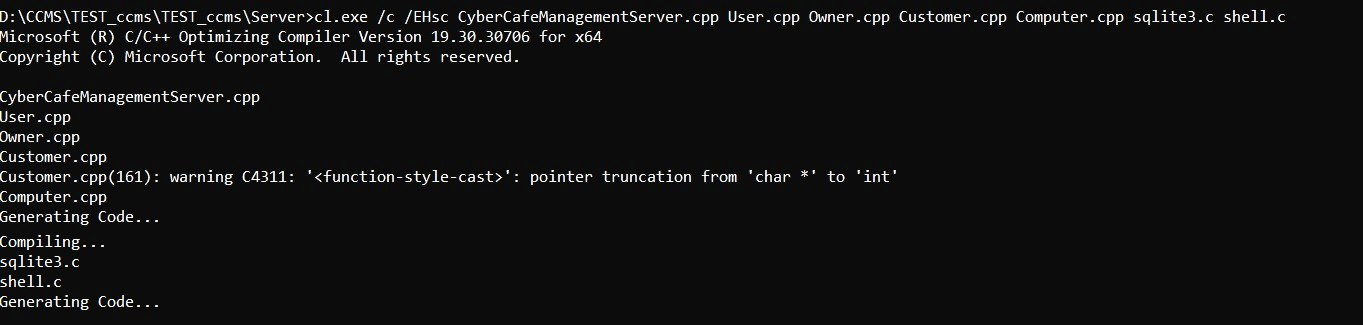
* **Steps of Execution to create libraries**

1. The cl.exe compiler generates an .obj file that contains the compiled code, and then runs the linker to create an executable file of .exe type. To compile a program that has additional source code files, enter them all on the command line, like:

cl.exe /c /EHsc file1.cpp file2.cpp file3.cpp

The /c command-line option instructs the compiler to compile and create .obj (object files).

The /EHsc command-line option instructs the compiler to enable standard C++ exception handling behaviour and synchronization. Without it, thrown exceptions can result in undestroyed objects and resource leaks.



1. The LINK.EXE program works primarily with object files as input, to produce an executable file (.exe) or dynamic link library(.dll) and static linked library (.lib) as output.

link.exe /DLL /DEF:file1.def file1.obj file2.obj user32.lib

1. Module definition file (.def) is a file which shows the list of functions which are library is exporting. Using .def file is the most standard way to create a static or dynamic library because it is “C” compliant. Libraries created using .def file can be used on multiple systems without any error.

