

C++ Static and Dynamic Libraries

Outline

- Why bother?
- Static verses Dynamic Linking

Why Bother?

- How most team based projects are managed
- Given interface and expected behavior
- Produce a library that delivers requirements
- Also reusable
- Excellent encapsulation, I code to the API (Application Programming Interface) your library has, I don't care how its implemented
- Also how most 3rd party software is delivered for programmer use.

Whats an API?

- Public access to your library.
- Typically provided via header files.
- For instance here is the interface to a static library that I will discuss today for the library StaticLibrary;

```
#pragma once

namespace KP_StaticLib{
    int getint();
}
```

Static Verses Dynamic

Static linkage

LibraryStatic
(.a on eclipse .lib on MSVS)

```
Void myfunc(){  
std::cout<<"In library";  
}
```

MyApp
(uses LibraryStatic)

```
Void myfunc(){  
std::cout<<"In library";  
}
```

AnotherApp
(uses LibraryStatic)

```
Void myfunc(){  
std::cout<<"In library";  
}
```

Shared linkage

LibraryShared
(.dll)

```
Void myfunc(){  
std::cout<<"In library";  
}
```

MyApp
(uses LibraryShared)

```
Void myfunc();
```

MyApp 2
(uses LibraryShared)

```
Void myfunc();
```

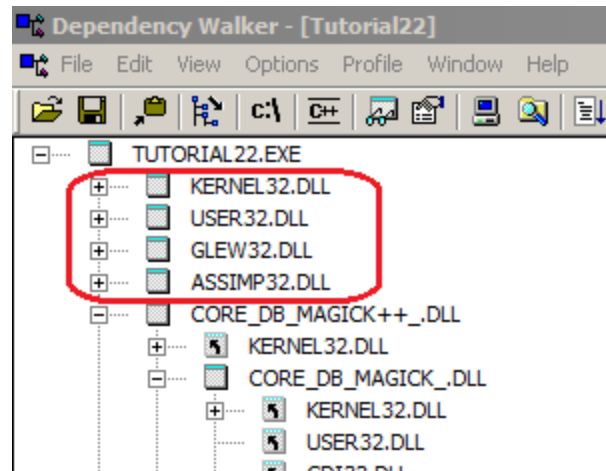
MyApp3
(uses LibraryShared)

```
Void myfunc();
```

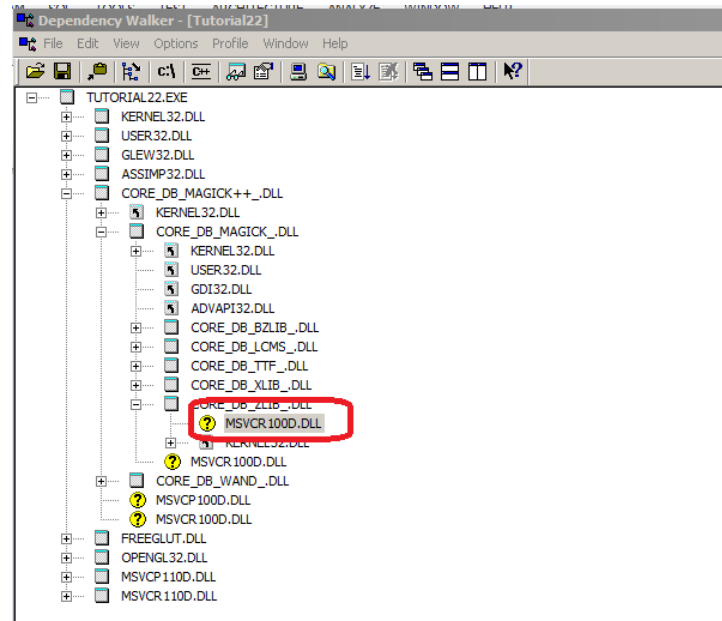


How it appears in Dependency Walker

Dynamic Link

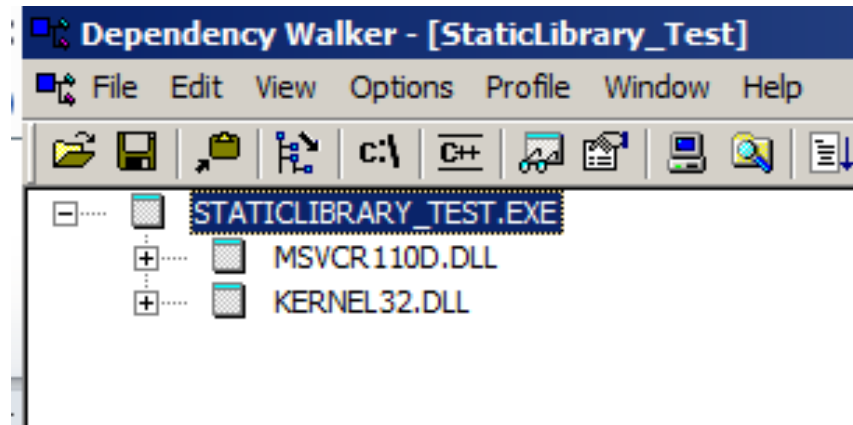


DLL – When things go wrong



Missing MSVCR100D.DLL

How it appears in Dependency Walker Static Link



You don't see it as its compiled in

Summary

Easier to use static linkage

Sacrifice flexibility and a little space

But, If you change the library need to recompile all apps that use it

Static linking eliminates a host of dynamic problems

- missing dynamic libraries

- libraries that are the wrong version

Build a Library

- Lets build a library
- Set up project (library and tester.exe)
- Putting all includes in one place for sharing
- Link the library to the tester
- Namespaces – cause you don't know what other libraries are being used and whether their function/class names are the same as yours
- Then dependencies (build the library first then the tester)
- See tutorial document on website