Math 157

DLL's What

DLL's Why

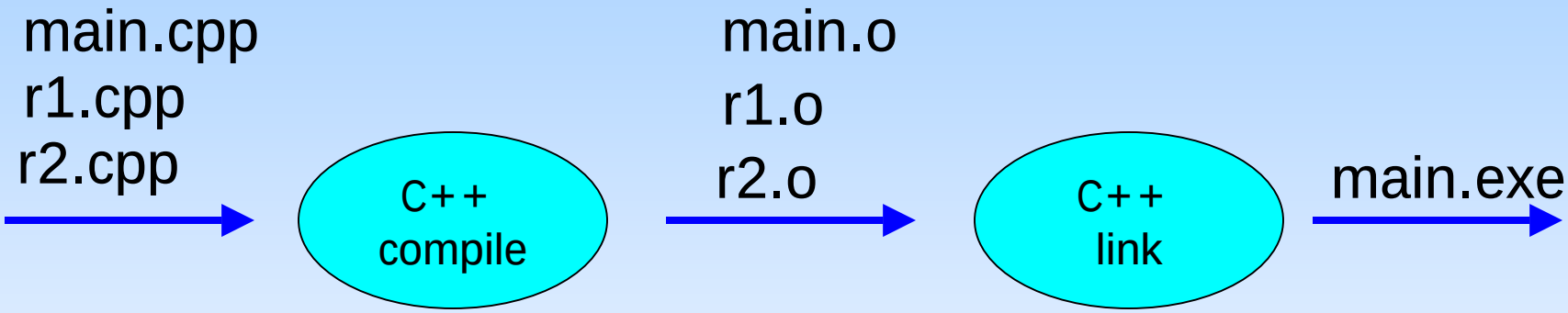
DLL's How

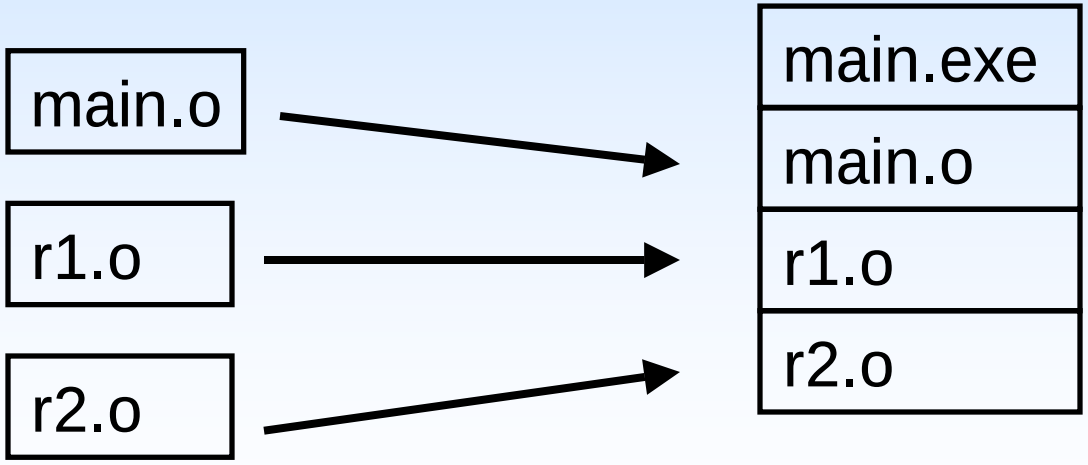
March 2, 2009

Ways to create an executable (1)

Main routine and two files with supporting routines.

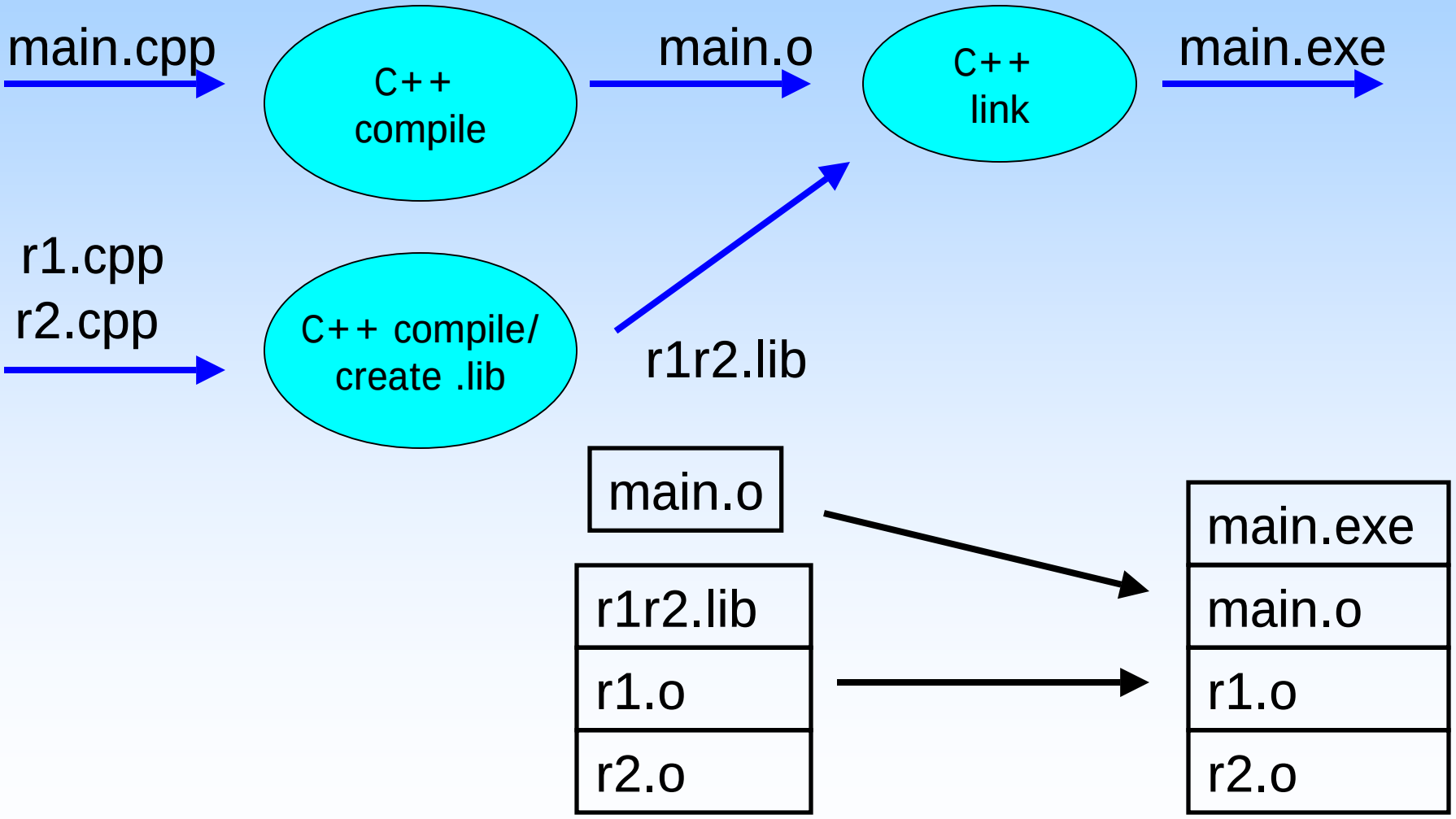
No libraries, all in one:





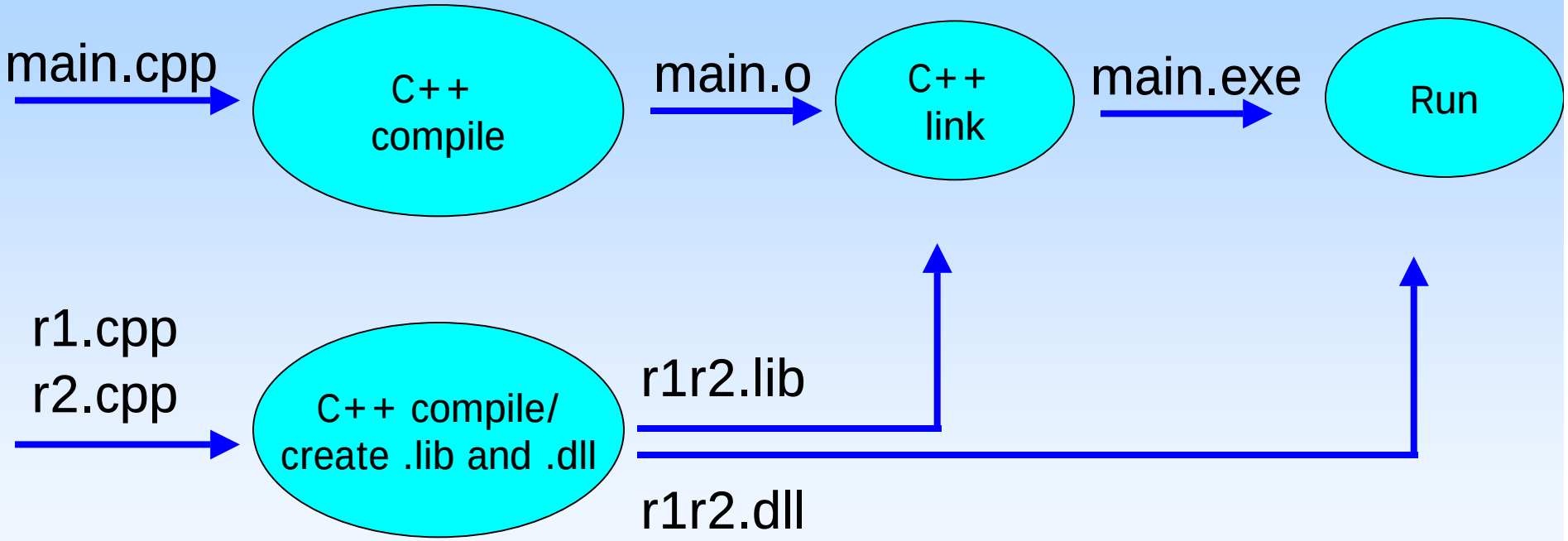
Ways to create an executable (2)

Static libraries



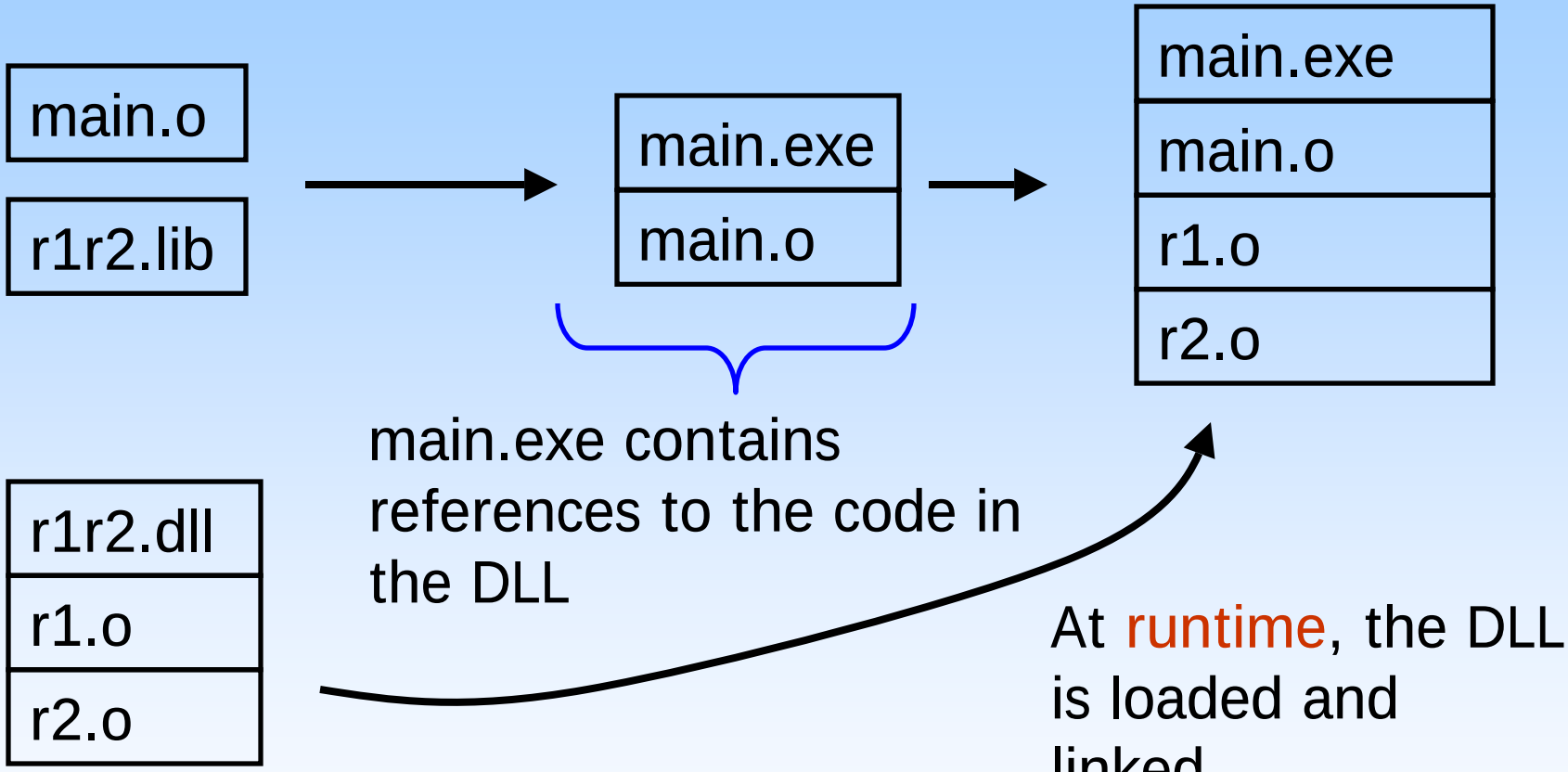
Ways to create an executable (3)

Dynamically Linked Libraries (DLL's)



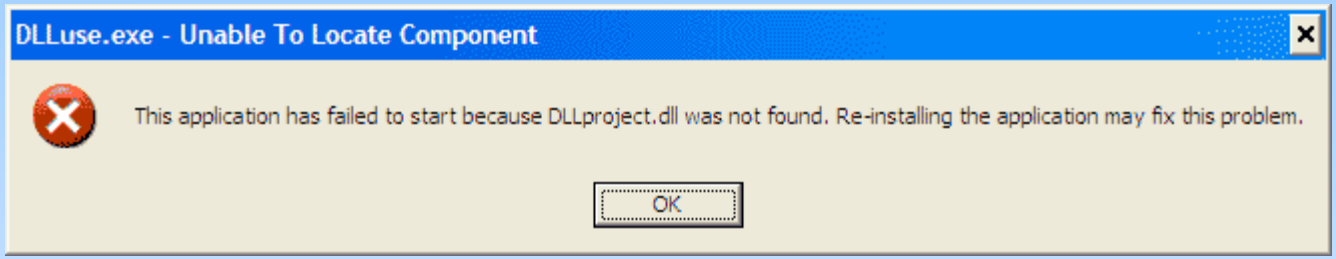
r1r2.lib ? *r1r2.lib is an "import" library, essentially a table of contents for the DLL.*

DLL inclusion



You need to have the DLL's in a place where the system can find them!!!!

What happens when a DLL can't be located?



- Find and place the DLL in your current search path

- Add the directory of the location of the DLL to your search path.

My Computer /
System Properties /
Advanced /
Environment Variables /
System Variables

Why use libraries (static or DLL)?

- Users of r1.cpp and r2.cpp do not require the source - only the header (.h) files and the library files (.lib, or .lib and .dll).

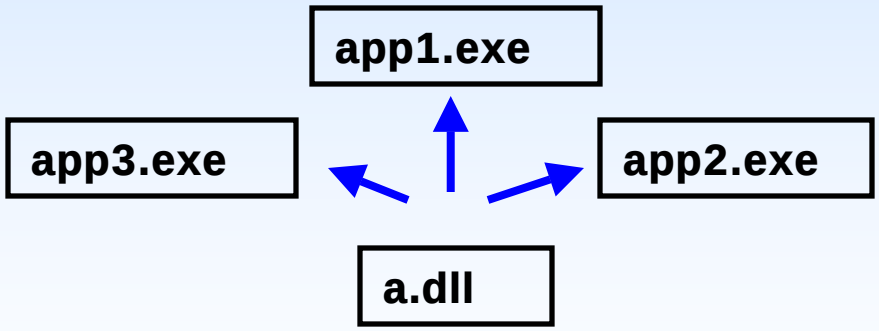
- Avoids recompilation of source.

Why use static libraries?

- Code self contained. No need to distribute .dll's and .exe's
- Execution does not require a "path" setting

Why use DLL libraries?

- The code in a DLL is shared among all .exe's that use it … so executables are smaller.
- Facilitates code maintenance/updates.



Static vs. dll ? Think about code maintenance …

Code maintenance and library structure

What is required to update codes that use a library?

Suppose r1.cpp has a bug, the bug gets fixed, how is the change incorporated?

Static Library

- recompile r1.cpp
- recreate the static library (r1r2.lib)
- relink and create a new .exe

Every application that depends upon the library must be recreated

DLL Library

- recompile r1.cpp
- recreate the DLL library (r1r2.dll)
- redistribute the new DLL

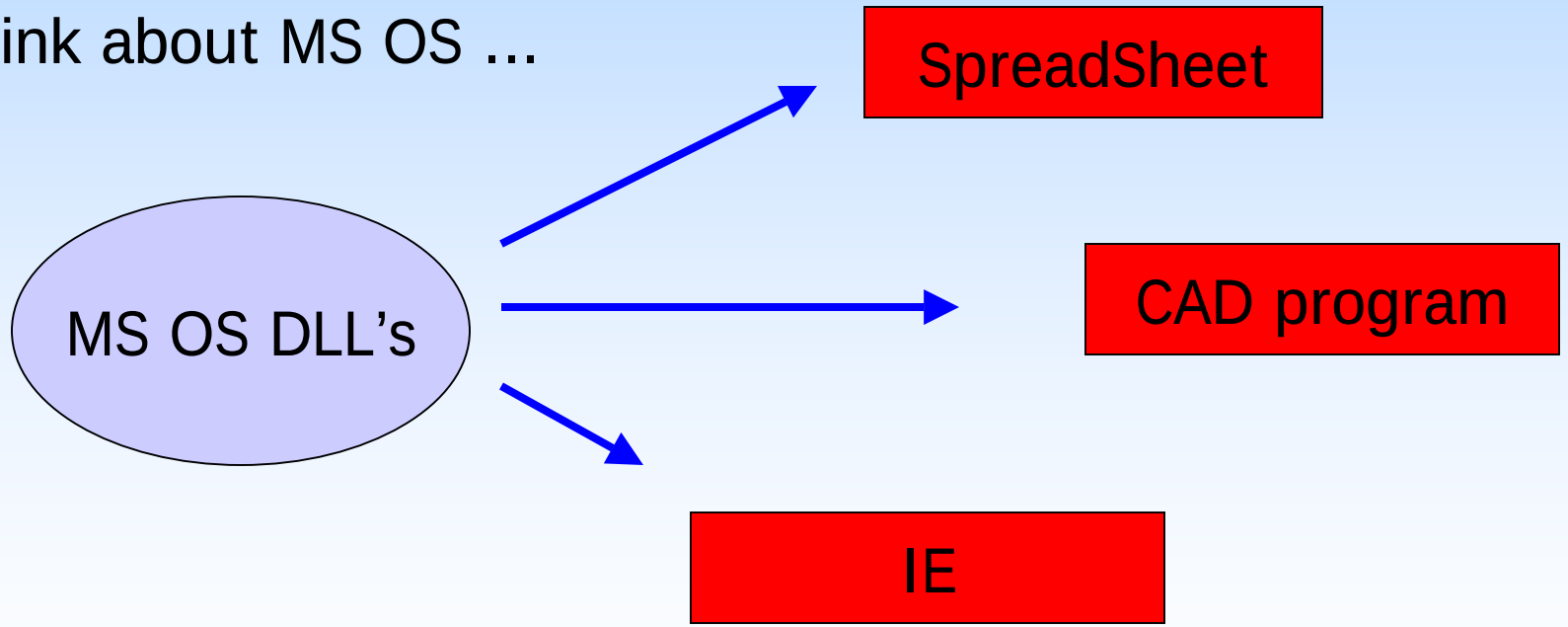
No need to change the application that uses the DLL

Code maintenance and library structure

!!! Important aspect about DLL libraries

Dependent applications do not need to be re-compiled if the .dll code is updated.

Think about MS OS ...



Code maintenance and library structure

All is not perfect ...

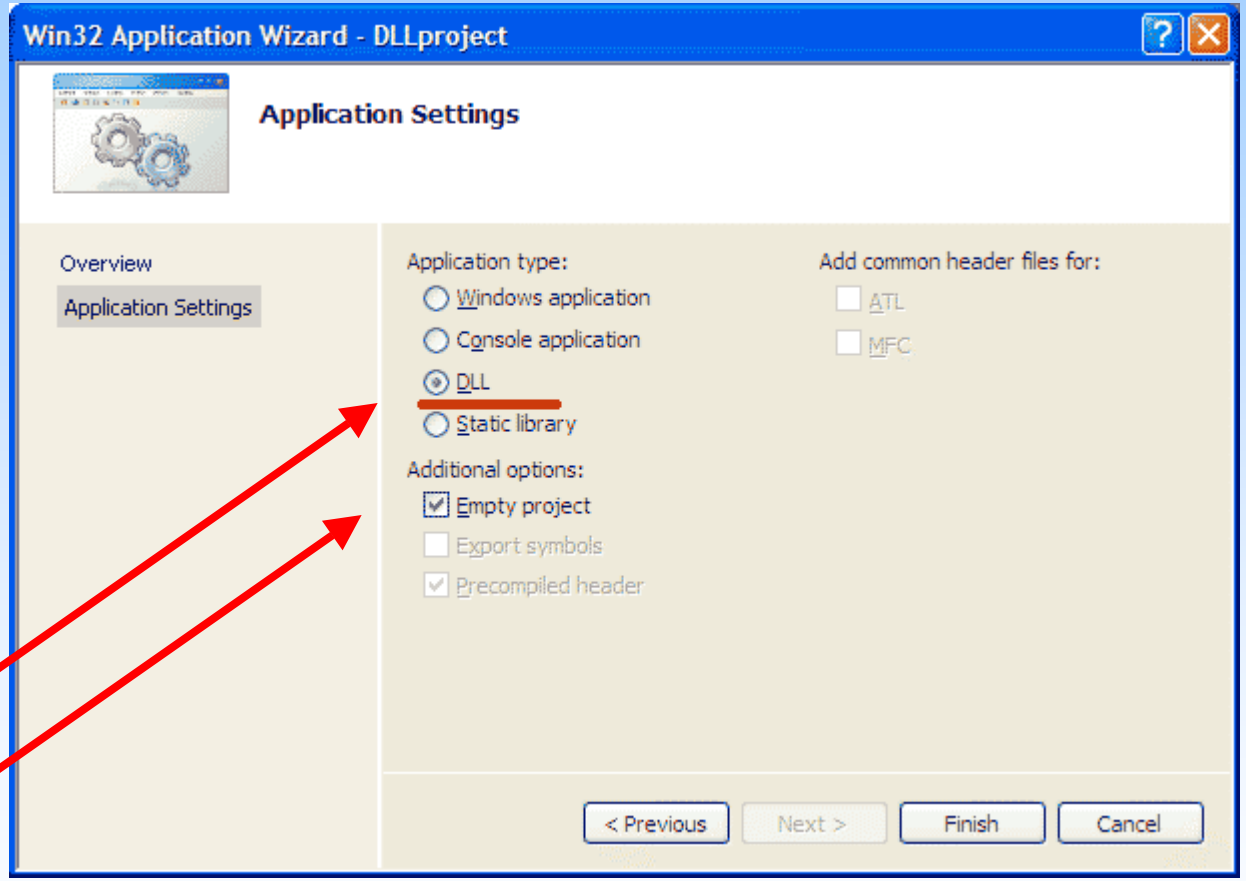
- “Updated” DLL’s can often cause working programs to cease to function

- Applications are often distributed with “updated” DLL’s that overwrite existing DLL’s. This can cause problems if the “updated” DLL is actually older than the one it is replacing.

- DLL’s are a vector for computer viruses

DLL library creation -- project settings.

Add New/Win32 Console Application project then...



Select DLL

Empty Project

DLL library creation ...

You have to specify classes you want “exported” in the DLL

- When *creating* the DLL use the __declspec(dllexport) macro

```
class __declspec(dllexport) SupportRoutines
{
        public:
        SupportRoutines();
        ~SupportRoutines();

        double timesTwo(double x);
};
```

Build the project → DLLproject.lib

DLLproject.dll

Creating a program that uses classes implemented in a .dll...

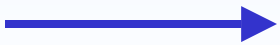
When *using* a class in a DLL, the header of the class must indicate that you will be "importing" it.

- Use the __declspec(dllimport) macro

```
class __declspec(dllimport) SupportRoutines
{
        public:
        SupportRoutines();
        ~SupportRoutines();

        double timesTwo(double x);
};
```

Create your program …
distribute the .dll with it



DLLtest.exe

DLLproject.dll

Use compiler directives to avoid changing headers

Header for class contained in a DLL

```
#ifdef  __MAKEDLL__                          // If __MAKEDLL__ is specified in
#define __IMPEXP__  __declspec(dllexport)    // (C/C++)/General/Preprocessor Definitions
#endif                                       // then __IMPEXP__ = __declspec(dllexport)

#ifndef  __MAKEDLL__                         // If __MAKEDLL__ is not specified
#define __IMPEXP__  __declspec(dllimport)    // then __IMPEXP__ = __declspec(dllimport)
#endif

#ifndef  _MSC_VER                            // For non-windows machines, define
#define __IMPEXP__                           // __IMPEXP__ = as a null, as
#endif                                       // __declspec is not a Microsoft specific

class __IMPEXP__ SupportRoutines
{
public:
        double timesTwo(double x);
        SupportRoutines();
        virtual ~SupportRoutines();
};
```

Demonstration ...

- Creating the DLL DLLproject.dll
- Creating a test program to use the DLL.