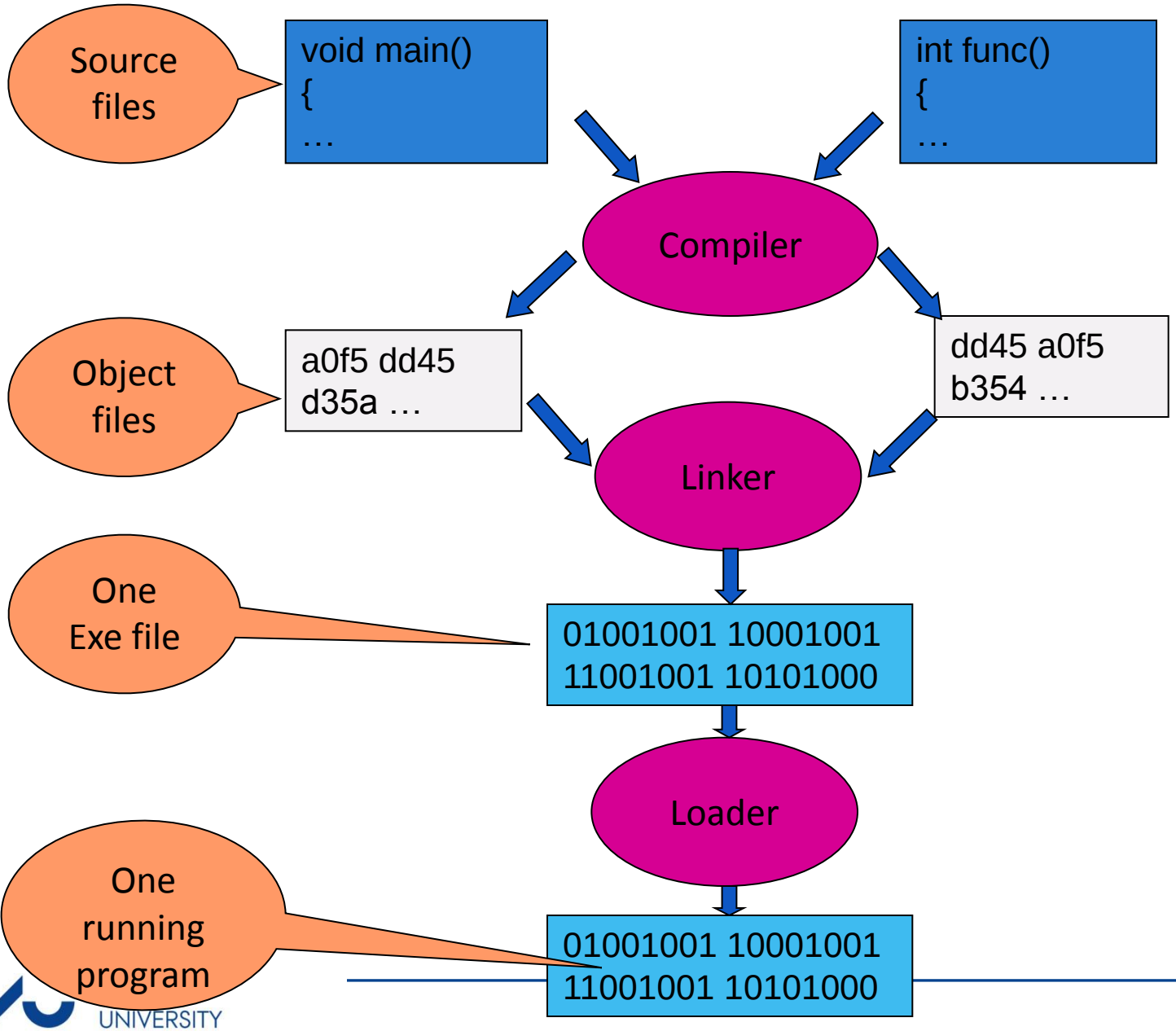
DLL files
in C++

Agenda

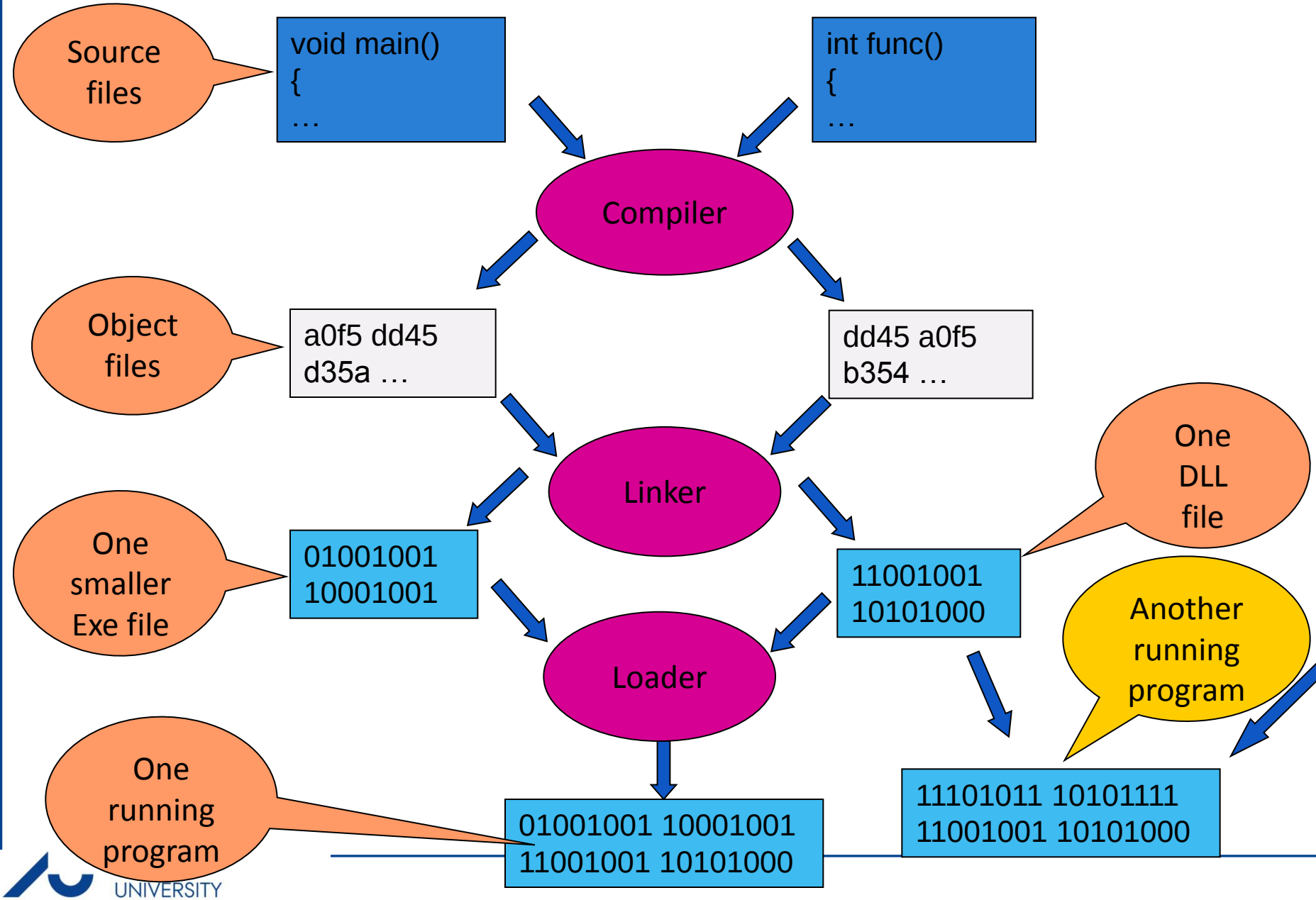
- What is a DLL-file?
- How to make a DLL-file?
- Load time use of DLL-files
- Run time use of DLL-files
- Name mangling

What is a DLL-file?

Static Linking



Dynamic Linking



Advantages of Dynamic Linking

- Saves system memory and reduces swapping.
  - Multiple processes, that load the same DLL, share a single copy of the DLL in physical memory.
- Less linking needed during development.
  - When the functions in a DLL change, the applications that use them do not need to be recompiled or relinked as long as the function arguments, calling conventions, and return values do not change.
- Easy upgrade.
  - For example, a display driver DLL can be modified to support a display that was not available when the application was initially shipped.
- Callable from different programming languages
  - Programs written in different programming languages can call the same DLL function as long as the programs follow the same calling convention that the function uses.

Typer af DLL-filer

- **Traditionelle C-style Win32 DLL'er**
  Som kan underopdeles i 2 grupper:
  - Standard DLL'er som Microsoft leverer med Windows – er en del af operativsystmet.
  - Custum DLL'er som laves af diverse programudviklere.

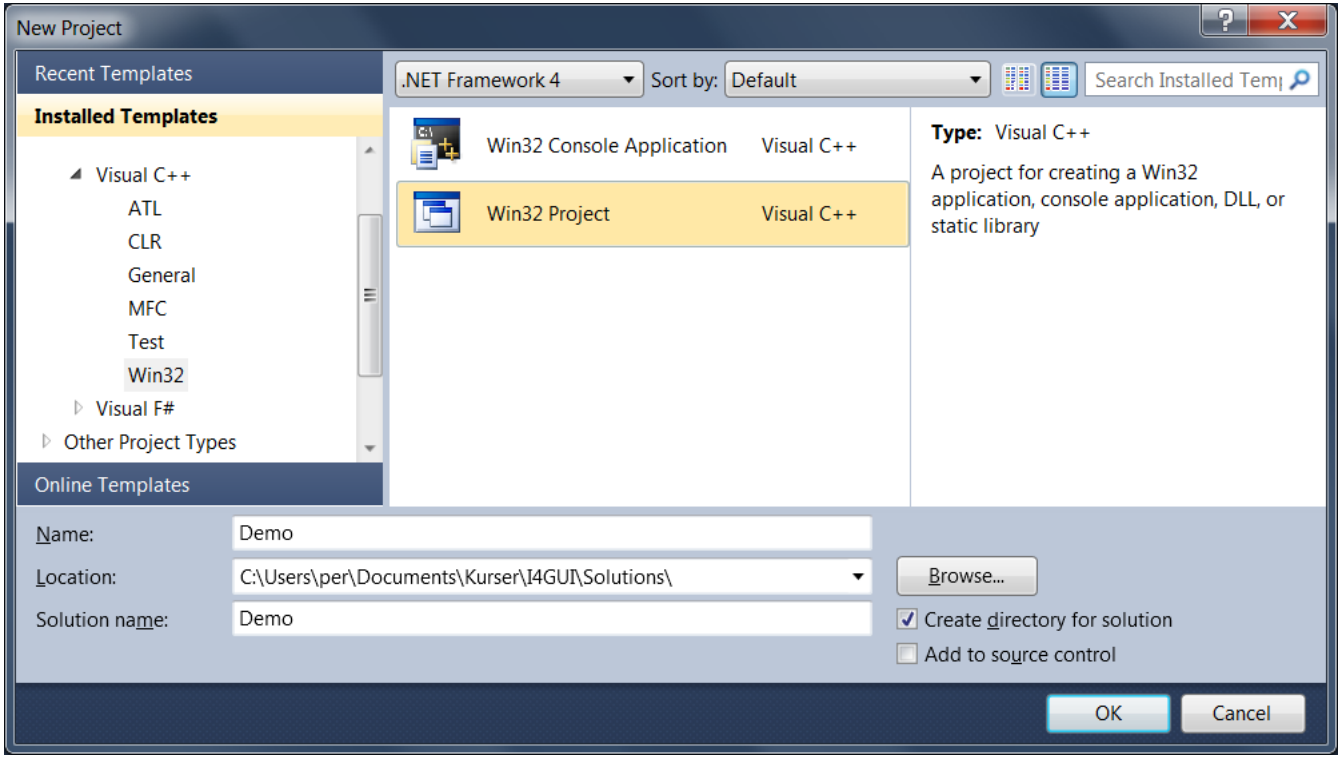
- **COM-baserede DLL'er**
  Er som en traditionel DLL – blot indeholder den altid 4 bestemte funktioner.

- **.Net DLL'er**
  Har internt en noget anderledes struktur. Indeholder f.eks. altid metadata, og koden er i IL-formatet og ikke maskinkode.

How to make a DLL-file?

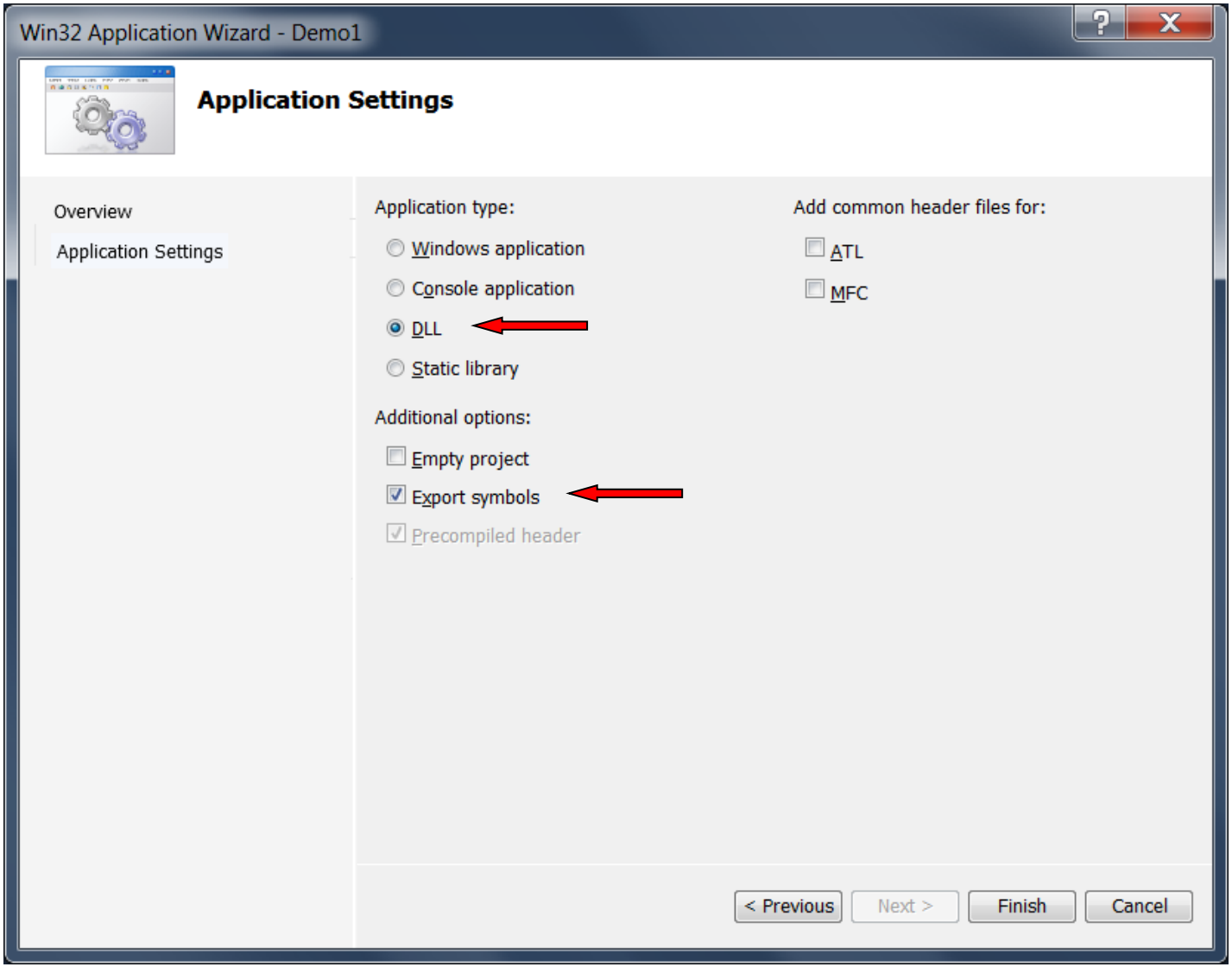
Use the Wizard to make a DLL-project

Step 1:



Use the Wizard to make a DLL-project

Step 2:



C-style DLL'ens struktur

Alle DLL'er indeholder en funktion med navnet DllMain.

DllMain kaldes af operativsystemet både når DLL'en loades og unloades, og når en tråd attaches eller detaches.

```
BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved)
```

DllMain

```
BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call, LPVOID lpReserved)
{
  switch (ul_reason_for_call)
  {
     case DLL_PROCESS_ATTACH:
       MessageBox(NULL, "Dll is loading!", "DllMain()
       says...", MB_OK);
       break;
    case DLL_PROCESS_DETACH:
       MessageBox(NULL, "Dll is UNloading!", "DllMain()
       says...", MB_OK);
       break;
    }
    return TRUE;
}
```

Types of Dynamic Linking

- There are two methods for calling a function in a C-style Win32 DLL:
  - **load-time dynamic linking**
    a module makes explicit calls to exported DLL functions as if they were local functions.
    This requires you to link the module with the ***import library*** for the DLL that contains the functions (aka a ***.lib***-file).

  - **run-time dynamic linking**
    a module uses the **LoadLibrary** or **LoadLibraryEx** function to load the DLL at run time.
    After the DLL is loaded, the module calls the **GetProcAddress** function to get the addresses of the exported DLL functions.
    The module calls the exported DLL functions using the *function pointers* returned by GetProcAddress.

DLLs and Memory Management

- Every process that loads the DLL maps it into its virtual address space. After the process loads the DLL into its virtual address, it can call the exported DLL functions.
- Like any other function, an exported DLL function runs in the context of the thread that calls it. Therefore, the following conditions apply:
  - The threads of the process that called the DLL can use handles opened by a DLL function. Similarly, handles opened by any thread of the calling process can be used in the DLL function.
  - The DLL uses the stack of the calling thread and the virtual address space of the calling process.
  - The DLL allocates memory from the virtual address space of the calling process.

Name Mangling in C++

```
MYDLL_API int AddNumbers(int a, int b);
```

Visual Studio .NET 2003 Command Prompt

```
C:\Demo\KPU\DLLDemo\MyDll\Debug>dumpbin /exports mydll.dll
Microsoft (R) COFF/PE Dumper Version 7.10.3077
Copyright (C) Microsoft Corporation.  All rights reserved.


Dump of file mydll.dll

File Type: DLL

  Section contains the following exports for MyDll.dll

    00000000 characteristics
    412DCFA5 time date stamp Thu Aug 26 13:55:17 2004
        0.00 version
           1 ordinal base
           3 number of functions
           3 number of names

    ordinal hint RVA      name

          1    0 00011479 ??0CMyDll@@QAE@XZ
          2    1 00011208 ??4CMyDll@@QAEAAV0@ABV0@@Z
          3    2 000110EB ?AddNumbers@@YAHHH@Z
```

The C++ compiler mangles function names to implement overloaded functions and type safety

Name Mangling in C++

```
extern "C" MYDLL_API int AddNumbers(int a, int b);
```

Use extern "C" to turn off name mangling!

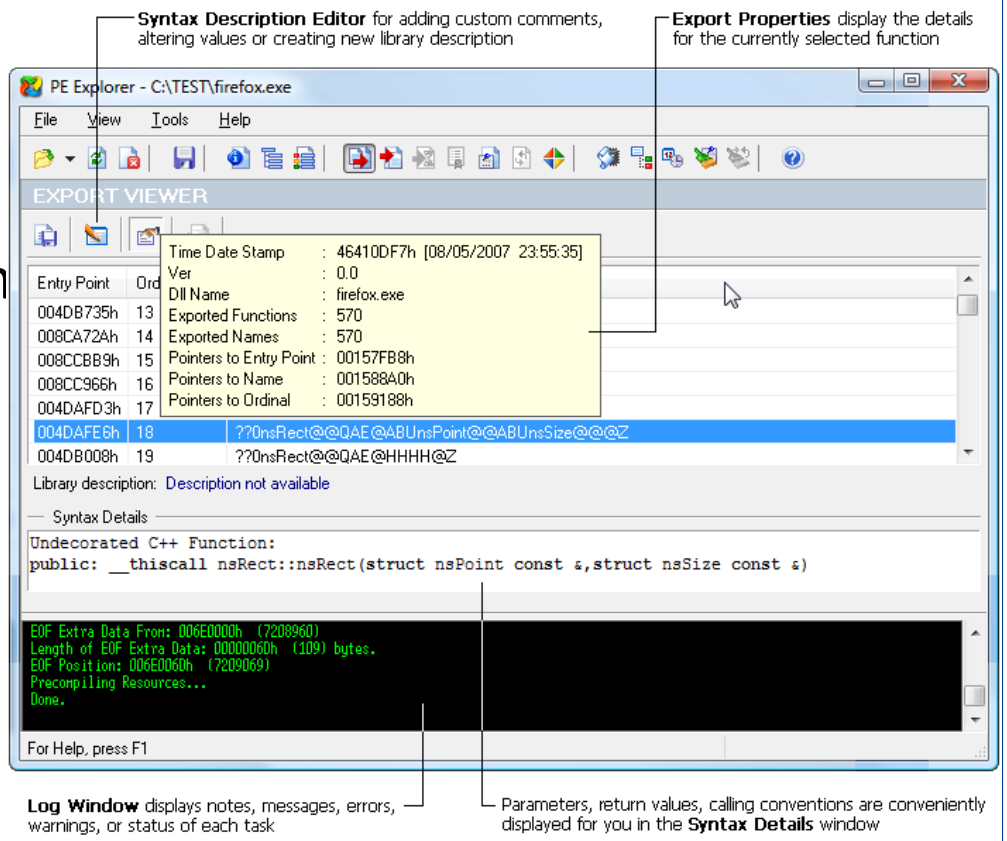
Visual Studio .NET 2003 Command Prompt

```
C:\Demo\KPU\DLLDemo\MyDll\Debug>dumpbin /exports mydll.dll
Microsoft (R) COFF/PE Dumper Version 7.10.3077
Copyright (C) Microsoft Corporation.  All rights reserved.


Dump of file mydll.dll

File Type: DLL

  Section contains the following exports for MyDll.dll

    00000000 characteristics
    4316D853 time date stamp Thu Sep 01 12:30:43 2005
        0.00 version
           1 ordinal base
           3 number of functions
           3 number of names

    ordinal hint RVA      name

          1    0 00011479 ??0CMyDll@@QAE@XZ
          2    1 00011208 ??4CMyDll@@QAEAAV0@ABV0@@Z
          3    2 00011163 AddNumbers
```

PE File Explorer

With PE Explorer You Can

- See what's inside an executable
- Customize the GUI elements of your favorite Windows programs
- Track down what a program accesses and which DLLs are called
- Understand the way a program works, behaves, and interacts with others
- http://www.pe-explorer.com/



References And Links

- MSDN:
  - http://msdn.microsoft.com/en-us/library/ms682589(v=VS.85).aspx
  - http://msdn.microsoft.com/en-us/library/1ez7dh12.aspx
- Wikipedia:
  - http://en.wikipedia.org/wiki/Dynamic-link_library
  - http://en.wikipedia.org/wiki/Dynamic_linker (linux and Apple)