**Assignment No : 06**

**TITLE :** Write a Dynamic Link Library for scientific calculator and Test it.

**PROBLEM STATEMENT :** Write a program to create Dynamic Link Library for any mathematical operation and write an application program to test it. (Java Native Interface / Use VB or VC++).

**OBJECTIVE :** To understand the use of DLL and utilize the dll by defining it.

**THEORY :**

A dynamic**-**linklibrary (DLL) is a module that contain functions and data that can be used by another module (application or DLL).A DLL can define two kinds of functions:

1. **Exported**
2. **Internal**

The **exported** functions are intended to be called by other modules, as well as from within the DLL where they are defined.

**Internal** functions are typically intended to be called only from within the DLL where they are defined.

Although a DLL can export data, its data is generally used only by its functions. However, there is nothing to prevent another module from reading or writing that address. DLLs provide a way to modularize applications so that their functionality can be updated and reused more easily. DLLs also help reduce memory overhead when several applications use the same functionality at the same time, because although each application receives its own copy of the DLL data, the applications share the DLL code.

The Windows application programming interface (API) is implemented as a set of dynamic-link libraries, so any process that uses the Windows API uses dynamic linking.

**About Dynamic-Link Libraries:**

Dynamic linking allows a module to include only the information needed to locate an exported DLL function at load time or run time. Dynamic linking differs from the more familiar static linking, in which the linker copies a library function's code into each module that calls it.

**Types of Dynamic Linking:**

There are two methods for calling a function in a DLL:

* In *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. An import library supplies the system with the information needed to load the DLL and locate the exported DLL functions when the application is loaded.
* In *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**. This eliminates the need for an import library.

**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. The system maintains a per-thread reference count for each DLL. When a thread loads the DLL, the reference count is incremented by one. When the process terminates, or when the reference count becomes zero (run-time dynamic linking only), the DLL is unloaded from the virtual address space of the process. 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.

**Advantages of Dynamic Linking:**

Dynamic linking has the following advantages over static linking:

* Multiple processes that load the same DLL at the same base address share a single copy of the DLL in physical memory. Doing this saves system memory and reduces swapping.
* 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. In contrast, statically linked object code requires that the application be relinked when the functions change.
* A DLL can provide after-market support. For example, a display driver DLL can be modified to support a display that was not available when the application was initially shipped.
* 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. The calling convention (such as C, Pascal, or standard call) controls the order in which the calling function must push the arguments onto the stack, whether the function or the calling function is responsible for cleaning up the stack, and whether any arguments are passed in registers. For more information, see the documentation included with your compiler.

A potential disadvantage to using DLLs is that the application is not self-contained; it depends on the existence of a separate DLL module. The system terminates processes using load-time dynamic linking if they require a DLL that is not found at process startup and gives an error message to the user. The system does not terminate a process using run-time dynamic linking in this situation, but functions exported by the missing DLL are not available to the program.

**Dynamic-Link Library Creation:**

To create a Dynamic-Link Library (DLL), you must create one or more source code files, and possibly a linker file for exporting the functions. If you plan to allow applications that use your DLL to use load-time dynamic linking, you must also create an import library.

**Creating Source Files:**

The source files for a DLL contain exported functions and data, internal functions and data, and an optional entry-point function for the DLL. You may use any development tools that support the creation of Windows-based DLLs.

If your DLL may be used by a multithreaded application, you should make your DLL "thread-safe". You must synchronize access to all of the DLL's global data to avoid data corruption. You must also ensure that you link only with libraries that are thread-safe as well. For example, Microsoft Visual C++ contains multiple versions of the C Run-time Library, one that is not thread-safe and two that are.

**Exporting Functions**

How you specify which functions in a DLL should be exported depends on the tools that you are using for development. Some compilers allow you to export a function directly in the source code by using a modifier in the function declaration. Other times, you must specify exports in a file that you pass to the linker.

For example, using Visual C++, there are two possible ways to export DLL functions: with \_declspec modifier or with a .def file. If you use the \_declspec modifier, it is not necessary to use a .def file.

**Creating an Import Library**

An import library (.lib) file contains information the linker needs to resolve external references to exported DLL functions, so the system can locate the specified DLL and exported DLL functions at run time.

**CONCLUSION:**

Thus, we have understood and implemented scientific calculator using DLL.

**FAQs:**

1)Explain advantages of using DLL?

2) Explain disadvantages of using DLL?

3) What is DLL? Why it is called as dynamic linking?

4)Explain types of Dynamic linking.

5) How DLL is created and called?

6) What you mean by dynamic linking with import?

7) What you mean by dynamic linking without import?

8) compare static linking with dynamic linking?

9) What do you mean by DDE?

10) What do you mean by call back functions in DLL?