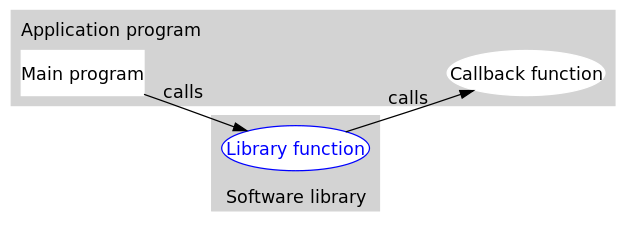
July 2nd, 2013

1. C stores all variables defined as static or global in the program’s heap space, and C stores all other variables on the stack.

A DLL has its own data segment but shares its stack with the calling program, this means that the DS and the SS do not point to the same location. The easiest solution to this problem is to build the DLL in the large memory model where all variables are referenced by a 32-bit value.

1. In Windows, you can dynamically allocate two types of memory, local and global. Local memory is limited to 64K, and in the case of a DLL, local memory is shared with the program that called the DLL. Global memory is all of the memory available to Windows after it has loaded.
2. If you try to store data in a DLL using global or static variables, don’t be surprised if these values have changed when you next call your DLL. The data stored in this way will be common to all applications that access this DLL. No matter how many applications use a DLL, there is only one instance of the DLL. The best way to get around this is to return structures from the DLL and pass them in again when they are needed.
3. It is not possible to share file handles between applications or DLLS. Each application has its own file-handle table. For two applications to use the same file using a DLL, they must both open the file individually.
4. Call Back Functions:

A callback is executable code that is passed as an argument to other code. It allows a lower-level software layer to call a function defined in a higher-level layer. Usually the higher-level code starts by calling a function within the lower-level code passing to it a pointer or handle to another function. While the lower-level function executes, it may call the passed-in function any number of times to perform some subtask. In another scenario, the lower-level function registers the padded-in function as a handler that is to be called asynchronously by the lower-level function. This can be a very powerful technique for code reuse. Callback functions separate the caller from the callee, the caller doesn’t care who the callee is.

[](http://upload.wikimedia.org/wikipedia/commons/d/d4/Callback-notitle.svg)

There are two types of callbacks: blocking callbacks(also known as synchronous callbacks) and deferred callbacks(also known as asynchronous callbacks). These two design choices differ in how they control data flow at runtime. While blocking callbacks are invoked before a function returns, deferred callbacks may be invoked after a function returns. Deferred callbacks are often used in the context of I/O operations or event handling. While deferred callbacks imply the existence of multithreads, blocking callbacks often rely on a single thread. This means that blocking callbacks are not commonly used for synchronization or delegating work to another thread.