

[...](#) / [MSVC linker reference](#) / [MSVC linker options](#) /

/BASE (Base Address)

Article • 08/03/2021 • 2 minutes to read • [7 contributors](#)

In this article

[Syntax](#)[Remarks](#)[See also](#)

Specifies the base address for a program.

Syntax

```
/BASE:{address[,size] | @filename,key}
```

Remarks

ⓘ Note

For security reasons, Microsoft recommends you use the **/DYNAMICBASE** option instead of specifying base addresses for your executables. This generates an executable image that can be randomly rebased at load time by using the address space layout randomization (ASLR) feature of Windows. The **/DYNAMICBASE** option is on by default.


The **/BASE** option sets a base address for the program, overriding the default location for an .exe or DLL file. The default base address for an .exe file is 0x400000 for 32-bit images or 0x140000000 for 64-bit images. For a DLL, the default base address is 0x10000000 for 32-bit images or 0x180000000 for 64-bit images. On operating systems that do not support address space layout randomization (ASLR), or when the **/DYNAMICBASE:NO** option was set, the operating system first attempts to load a program at its specified or default base address. If sufficient space is not available there, the system relocates the program. To prevent relocation, use the **/FIXED** option.

The linker issues an error if *address* is not a multiple of 64K. You can optionally specify the size of the program; the linker issues a warning if the program can't fit in the size you specified.


On the command line, another way to specify the base address is by using a base address response file. A base address response file is a text file that contains the base addresses and optional sizes of all the DLLs your program will use, and a unique text key for each base address. To specify a base address by using a response file, use an at sign (@) followed by the name of the response file, *filename*, followed by a comma, then the *key* value for the base address to use in the file. The linker looks for *filename* in either the specified path, or if no path is specified, in the directories specified in the LIB environment variable. Each line in *filename* represents one DLL and has the following syntax:

```
key address [size] ; comment
```

The *key* is a string of alphanumeric characters and is not case sensitive. It is usually the name of a DLL, but it need not be. The *key* is followed by a base *address* in C-language, hexadecimal, or decimal notation and an optional maximum *size*. All three arguments are separated by spaces or tabs. The linker issues a warning if the specified *size* is less than the virtual address space required by the program. A *comment* is specified by a semicolon (;) and can be on the same or a separate line. The linker ignores all text from the semicolon to the end of the line. This example shows part of such a file:

			 Copy
main	0x00010000	0x08000000	; for PROJECT.exe
one	0x28000000	0x00100000	; for DLLONE.DLL
two	0x28100000	0x00300000	; for DLLTWO.DLL

If the file that contains these lines is called DLLS.txt, the following example command applies this information:

		 Copy
<pre>link dlltwo.obj /dll /base:@dlls.txt,two</pre>		

Another way to set the base address is by using the *BASE* argument in a [NAME](#) or [LIBRARY](#) statement. The /BASE and /DLL options together are equivalent to the **LIBRARY** statement.

To set this linker option in the Visual Studio development environment

1. Open the project's **Property Pages** dialog box. For details, see [Set C++ compiler and build properties in Visual Studio](#).
2. Select the **Configuration Properties > Linker > Advanced** property page.
3. Modify the **Base Address** property.

To set this linker option programmatically

- See [BaseAddress](#).

See also

[MSVC linker reference](#)

[MSVC Linker Options](#)

Recommended content

[/SECTION \(Specify Section Attributes\)](#)

Learn more about: [/SECTION \(Specify Section Attributes\)](#)

[Using Thread Local Storage in a Dynamic-Link Library - Win32 apps](#)

This section shows the use of a DLL entry-point function to set up a thread local storage (TLS) index to provide private storage for each thread of a multithreaded process.

[Using wmain](#)

Learn more about: [Using wmain](#)

[DllMain entry point \(Process.h\) - Win32 apps](#)

An optional entry point into a dynamic-link library (DLL). When the system starts or terminates a process or thread, it calls the entry-point function for each loaded DLL using the first thread of the process.

[GetProcAddress](#)

Learn more about: [GetProcAddress](#)

[_ReturnAddress](#)

Learn more about: [_ReturnAddress](#)

[stricmp, wcsicmp](#)

Learn more about: [stricmp](#), [wcsicmp](#)

[__fastcall](#)

Learn more about: [__fastcall](#)

Show more 