**创建进程：**

**BOOL WINAPI CreateProcess(**

**\_\_in\_opt LPCTSTR lpApplicationName,**

**\_\_inout\_opt LPTSTR lpCommandLine,**

**\_\_in\_opt LPSECURITY\_ATTRIBUTES lpProcessAttributes,**

**\_\_in\_opt LPSECURITY\_ATTRIBUTES lpThreadAttributes,**

**\_\_in BOOL bInheritHandles,**

**\_\_in DWORD dwCreationFlags,**

**\_\_in\_opt LPVOID lpEnvironment,**

**\_\_in\_opt LPCTSTR lpCurrentDirectory,**

**\_\_in LPSTARTUPINFO lpStartupInfo,**

**\_\_out LPPROCESS\_INFORMATION lpProcessInformation**

**);**

创建一个新的进程以及进程的主线程。新的进程运行在调用该函数的进程的安全环境中。

如果调用进程模拟了另外一个用户，那么新进程使用调用进程的令牌，而不是这个进程的模拟令牌。如果想让新建进程运行在模拟令牌表示的安全环境中，使用CreateProcessAsUser和CreateProcessWithLongW函数。

**lpApplicationName：**

被执行的模块的名称。这个模块可以是一个windows应用程序。也可以是其他类型的模块（例如MS-DOS或者OS/2）。当然，如果要运行这些类型的模块，操作系统必须有支持这些模块运行的子系统。

这个字符串，可以指定可执行文件的全路径名和文件名，也可以指定一个相对路径文件名。如果指定的是相对路径和文件名，那么函数使用当前的磁盘号和当前目录，来将这个相对路径补充成全路径。这个函数不使用查找路径。这个参数必须包括文件的扩展名，没有默认的文件扩展名！

这个lpApplicationName参数可以使NULL。在这种情况下，lpCommandLine参数的第一个字符，必须是一个空白字符。如果可执行文件的文件名中，包含有空格，那么使用引用字符串，来表示文件名的结束和参数的开始；否则，文件名就不明确。例如，考虑这个文件名字符串：”c:\program files\sub dir\program name”。这个字符串可以被解释成好多种字符串。

c:\program.exe files\sub dir\program name

c:\program files\sub.exe dir\program name

c:\program files\sub dir\program.exe name

c:\program ifles\sub dir\program name.exe

如果可执行模块是16为程序，那么lpApplicationName应该是NULL，lpCommandLine应该指向可执行模块和其参数。

为了运行一个批处理文件，你必须启动命令行解释器；将lpApplicationName设置成cmd.exe，将lpCommandLine参数设置成如下形式：/c加上批处理文件的文件名。

**lpCommandLine：**

被执行的命令行参数。这个字符串的最大长度可以达到32768个字符，包括null结尾符。如果lpApplicationName是NULL，那么lpCommandLine参数中的可执行文件的名字被限定在MAX\_PATH个字符之内。

这个函数的UNICODE版本，CreateProcessW，会修改这个字符串的内容。因此，这个参数不能指向只读内存（例如一个字符串常量）。如果这个参数是一个常量字符串，那么这个函数将引发一个违规访问。

lpCommandLine参数可以是NULL。此时，函数使用lpApplicationName指向的字符串，作为命令行字符串。

如果lpApplicationName和lpCommandLine都是no-NULL，那么lpApplicationName就是可执行文件的文件名，而lpCommandLine指向的就是命令行参数。新进程可以使用GetCommandLine函数，来获取完整的命令行。控制台进程使用argc和argv参数，来分析命令行。此时argv[0]代表可执行文件的名称，作为命令行的第一个参数。

如果lpApplicationName是NULL，那么第一个空白符前面的字符串，被看做是可执行文件的文件名。如果使用的文件名或者路径名中包含了空格，那么需要使用一个引号，将文件名字符串包围起来。如果文件名没有扩展名，那么系统会添加exe作为其文件的扩展名。如果文件名的后缀扩展名为.com，那么就必须在文件名后，添加.com作为其后缀。如果文件名不包含路径，那么系统会按按照下面的顺序查找文件：

1. 进程可执行文件所在目录
2. 父进程的当前目录
3. GetSystemDirectory函数获取的系统目录。
4. 16位windows系统目录。没有函数可以获得这个系统目录，但这个目录确实会被搜索。这个系统目录是System。
5. windows目录。也就是GetWindowsDirectory函数获得的目录。
6. 在PATH环境变量中列出的目录。注意，这个函数并不搜索App Paths注册表键定义的路径。如果想搜索这个目录下的目录，使用ShellExecute函数。

**lpProcessAttributes：**

一个指向SECURITY\_ATTRIBUTES结构的指针，这个结构中，最重要的数据结构是一个安全描述符，他决定了新产生的进程对象，是否能被其他子进程继承，这个进程对象，可以被那些用户访问灯。如果这个参数是NULL，那么产生的进程的进程对象不能被继承。

SECURITY\_ATTRIBUTES结构体的成员变量lpSecurityDescriptor，指向一个安全描述符，这个安全描述符将做为新进程的安全描述符。如果lpProcessAttributes=NULL，或者lpSecurityDescriptor=NULL，那么新进程将使用默认的安全描述符。在默认的安全描述符中，ACLs来自于创建新进程的原始令牌。

**lpThreadAttributes:**

一个指向SECURITY\_ATTRIBUTES结构体的指针。如果lpThreadAttributes=NULL，那么新线程的句柄不能够被继承。

SECURITY\_ATTRIBUTES结构的lpSecurityDescriptor成员变量指向一个安全描述符，这个描述符将作为进程主线程的安全描述符。如果lpThreadAttributes=NULL，那么线程的描述符使用缺省描述符，其描述符中的ACLs来自于进程令牌。

**dwCreationFlags：**

这个标志控制了进程的创建和优先级。进程的创建标志如下：

|  |  |
| --- | --- |
| **Constant/value** | **Description** |
| CREATE\_BREAKAWAY\_FROM\_JOB  0x01000000 | The child processes of a process associated with a job are not associated with the job.  If the calling process is not associated with a job, this constant has no effect. If the calling process is associated with a job, the job must set the JOB\_OBJECT\_LIMIT\_BREAKAWAY\_OK limit. |
| CREATE\_DEFAULT\_ERROR\_MODE  0x04000000 | The new process does not inherit the error mode of the calling process. Instead, the new process gets the default error mode.  This feature is particularly useful for multi-threaded shell applications that run with hard errors disabled.  The default behavior is for the new process to inherit the error mode of the caller. Setting this flag changes that default behavior. |
| CREATE\_NEW\_CONSOLE  0x00000010 | The new process has a new console, instead of inheriting its parent's console (the default). For more information, see [Creation of a Console](http://127.0.0.1:47873/help/1-4792/ms.help?method=page&id=84EC2559-CADE-447E-8594-5B824D3D3E81&product=VS&productVersion=100&topicVersion=85&locale=ZH-CN&topicLocale=EN-US).  This flag cannot be used with DETACHED\_PROCESS. |
| CREATE\_NEW\_PROCESS\_GROUP  0x00000200 | The new process is the root process of a new process group. The process group includes all processes that are descendants of this root process. The process identifier of the new process group is the same as the process identifier, which is returned in the lpProcessInformation parameter. Process groups are used by the [GenerateConsoleCtrlEvent](http://127.0.0.1:47873/help/1-4792/ms.help?method=page&id=ED392D97-6FD0-4256-A783-BC7D27D01A10&product=VS&productVersion=100&topicVersion=85&locale=ZH-CN&topicLocale=EN-US) function to enable sending a CTRL+BREAK signal to a group of console processes.  If this flag is specified, CTRL+C signals will be disabled for all processes within the new process group.  This flag is ignored if specified with CREATE\_NEW\_CONSOLE. |
| CREATE\_NO\_WINDOW  0x08000000 | The process is a console application that is being run without a console window. Therefore, the console handle for the application is not set.  This flag is ignored if the application is not a console application, or if it is used with either CREATE\_NEW\_CONSOLE or DETACHED\_PROCESS. |
| CREATE\_PROTECTED\_PROCESS  0x00040000 | The process is to be run as a protected process. The system restricts access to protected processes and the threads of protected processes. For more information on how processes can interact with protected processes, see [Process Security and Access Rights](http://127.0.0.1:47873/help/1-4792/ms.help?method=page&id=508A17C4-88CD-431A-A102-00180A7F7AB5&product=VS&productVersion=100&topicVersion=85&locale=ZH-CN&topicLocale=EN-US).  To activate a protected process, the binary must have a special signature. This signature is provided by Microsoft but not currently available for non-Microsoft binaries. There are currently four protected processes: media foundation, audio engine, Windows error reporting, and system. Components that load into these binaries must also be signed. Multimedia companies can leverage the first two protected processes. For more information, see 联机[Overview of the Protected Media Path](http://go.microsoft.com/FWLink/?LinkId=83957).  Windows Server 2003 and Windows XP/2000: This value is not supported. |
| CREATE\_PRESERVE\_CODE\_AUTHZ\_LEVEL  0x02000000 | Allows the caller to execute a child process that bypasses the process restrictions that would normally be applied automatically to the process.  Windows 2000: This value is not supported. |
| CREATE\_SEPARATE\_WOW\_VDM  0x00000800 | This flag is valid only when starting a 16-bit Windows-based application. If set, the new process runs in a private Virtual DOS Machine (VDM). By default, all 16-bit Windows-based applications run as threads in a single, shared VDM. The advantage of running separately is that a crash only terminates the single VDM; any other programs running in distinct VDMs continue to function normally. Also, 16-bit Windows-based applications that are run in separate VDMs have separate input queues. That means that if one application stops responding momentarily, applications in separate VDMs continue to receive input. The disadvantage of running separately is that it takes significantly more memory to do so. You should use this flag only if the user requests that 16-bit applications should run in their own VDM. |
| CREATE\_SHARED\_WOW\_VDM  0x00001000 | The flag is valid only when starting a 16-bit Windows-based application. If the DefaultSeparateVDM switch in the Windows section of WIN.INI is TRUE, this flag overrides the switch. The new process is run in the shared Virtual DOS Machine. |
| CREATE\_SUSPENDED  0x00000004 | The primary thread of the new process is created in a suspended state, and does not run until the [ResumeThread](http://127.0.0.1:47873/help/1-4792/ms.help?method=page&id=FFC4E474-635B-4BF7-A68F-073899FB3FDE&product=VS&productVersion=100&topicVersion=85&locale=ZH-CN&topicLocale=EN-US) function is called. |
| CREATE\_UNICODE\_ENVIRONMENT  0x00000400 | If this flag is set, the environment block pointed to by lpEnvironment uses Unicode characters. Otherwise, the environment block uses ANSI characters. |
| DEBUG\_ONLY\_THIS\_PROCESS  0x00000002 | The calling thread starts and debugs the new process. It can receive all related debug events using the [WaitForDebugEvent](http://127.0.0.1:47873/help/1-4792/ms.help?method=page&id=0D81A4AC-DD66-4648-9F3F-1F54ACA84243&product=VS&productVersion=100&topicVersion=85&locale=ZH-CN&topicLocale=EN-US) function. |
| DEBUG\_PROCESS  0x00000001 | The calling thread starts and debugs the new process and all child processes created by the new process. It can receive all related debug events using the [WaitForDebugEvent](http://127.0.0.1:47873/help/1-4792/ms.help?method=page&id=0D81A4AC-DD66-4648-9F3F-1F54ACA84243&product=VS&productVersion=100&topicVersion=85&locale=ZH-CN&topicLocale=EN-US) function.  A process that uses DEBUG\_PROCESS becomes the root of a debugging chain. This continues until another process in the chain is created with DEBUG\_PROCESS.  If this flag is combined with DEBUG\_ONLY\_THIS\_PROCESS, the caller debugs only the new process, not any child processes. |
| DETACHED\_PROCESS  0x00000008 | For console processes, the new process does not inherit its parent's console (the default). The new process can call the [AllocConsole](http://127.0.0.1:47873/help/1-4792/ms.help?method=page&id=BDF3BF2F-8EB8-4BA6-BF3A-A67B29DFFDA2&product=VS&productVersion=100&topicVersion=85&locale=ZH-CN&topicLocale=EN-US) function at a later time to create a console. For more information, see [Creation of a Console](http://127.0.0.1:47873/help/1-4792/ms.help?method=page&id=84EC2559-CADE-447E-8594-5B824D3D3E81&product=VS&productVersion=100&topicVersion=85&locale=ZH-CN&topicLocale=EN-US).  This value cannot be used with CREATE\_NEW\_CONSOLE. |
| EXTENDED\_STARTUPINFO\_PRESENT  0x00080000 | The process is created with extended startup information; the lpStartupInfo parameter specifies a [STARTUPINFOEX](http://127.0.0.1:47873/help/1-4792/ms.help?method=page&id=61203F57-292D-4EA1-88F4-A3B05012D7A3&product=VS&productVersion=100&topicVersion=85&locale=ZH-CN&topicLocale=EN-US) structure.  Windows Server 2003 and Windows XP/2000: This value is not supported. |
| INHERIT\_PARENT\_AFFINITY  0x00010000 | The process inherits its parent's affinity. If the parent process has threads in more than one [processor group](http://127.0.0.1:47873/help/1-4792/ms.help?method=page&id=C627AC0F-96E8-48B5-9103-4316F487E173&product=VS&productVersion=100&topicVersion=85&locale=ZH-CN&topicLocale=EN-US), the new process inherits the group-relative affinity of an arbitrary group in use by the parent.  Windows Server 2008, Windows Vista, Windows Server 2003, and Windows XP/2000: This value is not supported. |

Remarks

On 32-bit Windows, 16-bit applications are simulated by ntvdm.exe, not run as individual processes. Therefore, the process creation flags apply to ntvdm.exe. Because ntvdm.exe persists after you run the first 16-bit application, when you launch another 16-bit application, the new creation flags are not applied, except for CREATE\_NEW\_CONSOLE and CREATE\_SEPARATE\_WOW\_VDM, which create a new ntvdm.exe.

这个参数还可以控制进程的优先级，这个优先级决定了进程中的线程的优先级。如果没有优先级标志给出，那么新进程的优先级为NORMAL\_PRIORITY\_CLASS，如果父进程的优先级是IDLE\_PRIORITY\_CLASS或者BELOW\_NORMAL\_PRIORITY\_CLASS，那么被创建的子进程的优先级则继承父进程的优先级。

|  |  |
| --- | --- |
| Return code/value | Description |
| ABOVE\_NORMAL\_PRIORITY\_CLASS  0x00008000（第三高） | Process that has priority above NORMAL\_PRIORITY\_CLASS but below HIGH\_PRIORITY\_CLASS. |
| BELOW\_NORMAL\_PRIORITY\_CLASS  0x00004000（第五高） | Process that has priority above IDLE\_PRIORITY\_CLASS but below NORMAL\_PRIORITY\_CLASS. |
| HIGH\_PRIORITY\_CLASS  0x00000080（第二高） | Process that performs time-critical tasks that must be executed immediately for it to run correctly. The threads of a high-priority class process preempt the threads of normal or idle priority class processes. An example is the Task List, which must respond quickly when called by the user, regardless of the load on the operating system. Use extreme care when using the high-priority class, because a high-priority class CPU-bound application can use nearly all available cycles. |
| IDLE\_PRIORITY\_CLASS  0x00000040（最低） | Process whose threads run only when the system is idle and are preempted by the threads of any process running in a higher priority class. An example is a screen saver. The idle priority class is inherited by child processes. |
| NORMAL\_PRIORITY\_CLASS  0x00000020（第四高） | Process with no special scheduling needs. |
| REALTIME\_PRIORITY\_CLASS  0x00000100(最高) | Process that has the highest possible priority. The threads of a real-time priority class process preempt the threads of all other processes, including operating system processes performing important tasks. For example, a real-time process that executes for more than a very brief interval can cause disk caches not to flush or cause the mouse to be unresponsive. |

**lpEnvironment：**

一个指向环境变量内存块的指针。如果这个参数是NULL，那么新进程使用父进程的环境变量。环境变量块保存的是一个NULL结尾的字符串，每个环境变量在该字符串中的形式为：name=value\0

因为等号是分隔符，所以，在环境变量的名称中，千万不要出现等号。

环境变量块可以包含UNICODE或者ASCII字符。如果lpEnvironment指针指向的内存，包含UNICODE字符，那么请确保dwCreationFlags标志包含CREATE\_UNICODE\_ENVIRONMENT标志。如果这个lpEnvironment=NULL，那么子进程将继承父进程的环境变量块，同时你必须保证，dwCreationFlages包含CREATE\_UNICODE\_ENVIRONMENT标志。

这个函数的的ACSII版本，也就是CreateProcessA，在环境变量块的大小超过32767时，会失败。

注意，注意，这个环境变量块以2个零结尾。

**lpCurrentDirectory:**

进程的当前目录。这个字符产还可以定义一个UNC路径。如果lpCurrentDirectory=NULL，那么子进程的当前目录继承父进程的当前目录。

**lpStartupInfo：**

一个指向STARTUPINFO或者STARTUPINFOEX结构的指针。如果要设置扩展属性，那么dwCreateFlags标志中，应该包含EXTENDED\_STARTUPINFO\_PRESENT标志。

typedef struct \_STARTUPINFO {

DWORD cb; //startupinfo结构体的大小

LPTSTR lpReserved;

LPTSTR lpDesktop;//次进程归那个桌面。

LPTSTR lpTitle;

DWORD dwX;

DWORD dwY;

DWORD dwXSize;

DWORD dwYSize;

DWORD dwXCountChars;//对于控制台程序来说，一行有几个字符

DWORD dwYCountChars;// 对于控制台程序来说，有多少行。

DWORD dwFillAttribute;//对控制台程序来说，背景色和字体颜色

DWORD dwFlags;

WORD wShowWindow;//表示窗口是否显式。

WORD cbReserved2;

LPBYTE lpReserved2;

HANDLE hStdInput;

HANDLE hStdOutput;

HANDLE hStdError;

} STARTUPINFO, \*LPSTARTUPINFO;

typedef struct \_STARTUPINFOEX {

STARTUPINFO StartupInfo;

PPROC\_THREAD\_ATTRIBUTE\_LIST lpAttributeList;

} STARTUPINFOEX, \*LPSTARTUPINFOEX;

BOOL WINAPI InitializeProcThreadAttributeList(

\_\_out\_opt LPPROC\_THREAD\_ATTRIBUTE\_LIST lpAttributeList,

\_\_in DWORD dwAttributeCount,

\_\_reserved DWORD dwFlags,

\_\_inout PSIZE\_T lpSize

);

如果lpAttributeList不是NULL，那么lpSize参数输入lpAttributeList内存块的字节大小。如果lpAttributeList=NULL，那么这个参数接收buffer所需大小。

BOOL WINAPI UpdateProcThreadAttribute(

\_\_inout LPPROC\_THREAD\_ATTRIBUTE\_LIST lpAttributeList,

\_\_in DWORD dwFlags,

\_\_in DWORD\_PTR Attribute,

\_\_in PVOID lpValue,

\_\_in SIZE\_T cbSize,

\_\_out\_opt PVOID lpPreviousValue,

\_\_out\_opt PSIZE\_T lpReturnSize);

VOID WINAPI DeleteProcThreadAttributeList(

\_\_inout LPPROC\_THREAD\_ATTRIBUTE\_LIST lpAttributeList);

**lpProcessInformation：**

一个指向PROCESS\_INFORMATION结构的指针。

typedef struct \_PROCESS\_INFORMATION {

HANDLE hProcess;//进程句柄

HANDLE hThread;//主线程句柄

DWORD dwProcessId;//进程ID

DWORD dwThreadId;//主线程ID

} PROCESS\_INFORMATION, \*LPPROCESS\_INFORMATION;

相关的函数：CreateProcessAsUser和CreateProcessWithLogonW。

**进程内核对象的生命周期，一定比进程生命周期长，进程没有退出，代表进程的进程内核对象一定不会被销毁！！！！当进程退出以后，也就是说，进程消失以后，进程内核对象才可能被销毁。那么什么时候销毁这个进程内核对象呢？当进程内核对象的引用计数为零时，销毁进程内核对象。当进程退出的时候，进程内核对象的状态会发生变化，我们把进程退出后，代表该进程的内核对象的状态教激发态！如果进程在运行，那么代表进程的内核对象处于非激发态，此时，如果有类似WaitForSingleObject函数等待这个进程内核对象的话，这个WaitForSingleObject函数会被阻塞，直到进程退出，进程内核对象处于激发态。**