**Dispatch Queue 函数**

以下函数创建 [DispatcherQueueController](https://learn.microsoft.com/zh-cn/uwp/api/windows.system.dispatcherqueuecontroller)。

展开表

| **函数** | **说明** |
| --- | --- |
| [**CreateDispatcherQueueController**](https://learn.microsoft.com/zh-cn/windows/desktop/api/DispatcherQueue/nf-dispatcherqueue-createdispatcherqueuecontroller) | 创建 [DispatcherQueueController](https://learn.microsoft.com/zh-cn/uwp/api/windows.system.dispatcherqueuecontroller) ，用于管理在另一个线程上按优先级顺序运行排队任务的 [DispatcherQueue](https://learn.microsoft.com/zh-cn/uwp/api/windows.system.dispatcherqueue) 的生存期。 |

**处理函数**

以下函数用于 [进程](https://learn.microsoft.com/zh-cn/windows/win32/procthread/child-processes)。

展开表

| **函数** | **说明** |
| --- | --- |
| [**CreateProcess**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-createprocessa) | 创建新进程及其主线程。 |
| [**CreateProcessAsUser**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-createprocessasusera) | 创建新进程及其主线程。 新进程在指定令牌表示的用户的安全上下文中运行。 |
| [**CreateProcessWithLogonW**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-createprocesswithlogonw) | 创建新进程及其主线程。 然后，新进程在指定凭据的安全上下文中运行指定的可执行文件， (用户、域和密码) 。 |
| [**CreateProcessWithTokenW**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-createprocesswithtokenw) | 创建新进程及其主线程。 新进程在指定令牌的安全上下文中运行。 |
| [**ExitProcess**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-exitprocess) | 结束调用进程及其所有线程。 |
| [**FlushProcessWriteBuffers**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-flushprocesswritebuffers) | 刷新运行当前进程的线程的每个处理器的写入队列。 |
| [**FreeEnvironmentStrings**](https://learn.microsoft.com/zh-cn/windows/win32/api/processenv/nf-processenv-freeenvironmentstringsa) | 释放环境字符串块。 |
| [**GetCommandLine**](https://learn.microsoft.com/zh-cn/windows/win32/api/processenv/nf-processenv-getcommandlinea) | 检索当前进程的命令行字符串。 |
| [**GetCurrentProcess**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-getcurrentprocess) | 检索当前进程的伪句柄。 |
| [**GetCurrentProcessId**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-getcurrentprocessid) | 检索调用进程的进程标识符。 |
| [**GetCurrentProcessorNumber**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-getcurrentprocessornumber) | 检索调用此函数期间运行当前线程的处理器数。 |
| [**GetEnvironmentStrings**](https://learn.microsoft.com/zh-cn/windows/win32/api/processenv/nf-processenv-getenvironmentstrings) | 检索当前进程的环境块。 |
| [**GetEnvironmentVariable**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-getenvironmentvariable) | 从调用进程的环境块中检索指定变量的值。 |
| [**GetExitCodeProcess**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-getexitcodeprocess) | 检索指定进程的终止状态。 |
| [**GetGuiResources**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winuser/nf-winuser-getguiresources) | 检索图形用户界面 (GUI) 指定进程正在使用的对象的句柄计数。 |
| [**GetLogicalProcessorInformation**](https://learn.microsoft.com/zh-cn/windows/win32/api/sysinfoapi/nf-sysinfoapi-getlogicalprocessorinformation) | 检索有关逻辑处理器和相关硬件的信息。 |
| [**GetPriorityClass**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-getpriorityclass) | 检索指定进程的优先级类。 |
| [**GetProcessAffinityMask**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-getprocessaffinitymask) | 检索指定进程的进程关联掩码和系统的系统关联掩码。 |
| [**GetProcessGroupAffinity**](https://learn.microsoft.com/zh-cn/windows/win32/api/processtopologyapi/nf-processtopologyapi-getprocessgroupaffinity) | 检索指定进程的处理器组相关性。 |
| [**GetProcessHandleCount**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-getprocesshandlecount) | 检索属于指定进程的打开句柄数。 |
| [**GetProcessId**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-getprocessid) | 检索指定进程的进程标识符。 |
| [**GetProcessIoCounters**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-getprocessiocounters) | 检索指定进程执行的所有 I/O 操作的计帐信息。 |
| [**GetProcessMitigationPolicy**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Processthreadsapi/nf-processthreadsapi-getprocessmitigationpolicy) | 检索调用进程的缓解策略设置。 |
| [**GetProcessPriorityBoost**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-getprocesspriorityboost) | 检索指定进程的优先级提升控制状态。 |
| [**GetProcessShutdownParameters**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-getprocessshutdownparameters) | 检索当前调用进程的关闭参数。 |
| [**GetProcessTimes**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-getprocesstimes) | 检索指定进程的 计时信息。 |
| [**GetProcessVersion**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-getprocessversion) | 检索预期运行指定进程的系统的主版本号和次版本号。 |
| [**GetProcessWorkingSetSize**](https://learn.microsoft.com/zh-cn/windows/desktop/api/memoryapi/nf-memoryapi-getprocessworkingsetsize) | 检索指定进程的最小和最大工作集大小。 |
| [**GetProcessWorkingSetSizeEx**](https://learn.microsoft.com/zh-cn/windows/win32/api/memoryapi/nf-memoryapi-getprocessworkingsetsizeex) | 检索指定进程的最小和最大工作集大小。 |
| [**GetProcessorSystemCycleTime**](https://learn.microsoft.com/zh-cn/windows/win32/api/sysinfoapi/nf-sysinfoapi-getprocessorsystemcycletime) | 检索指定组中每个处理器执行延迟过程调用所用的周期时间， (DPC) 和中断服务例程 (ISR) 。 |
| [**GetStartupInfo**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-getstartupinfow) | 检索创建调用进程时指定的 [**STARTUPINFO**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/ns-processthreadsapi-startupinfoa) 结构的内容。 |
| [**IsImmersiveProcess**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winuser/nf-winuser-isimmersiveprocess) | 确定进程是否属于 Windows 应用商店应用。 |
| [**NeedCurrentDirectoryForExePath**](https://learn.microsoft.com/zh-cn/windows/win32/api/processenv/nf-processenv-needcurrentdirectoryforexepatha) | 确定是否应将当前目录包含在指定可执行文件的搜索路径中。 |
| [**OpenProcess**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-openprocess) | 打开现有的本地进程对象。 |
| [**QueryFullProcessImageName**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-queryfullprocessimagenamea) | 检索指定进程的可执行映像的完整名称。 |
| [**QueryProcessAffinityUpdateMode**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-queryprocessaffinityupdatemode) | 检索指定进程的相关性更新模式。 |
| [**QueryProcessCycleTime**](https://learn.microsoft.com/zh-cn/windows/win32/api/realtimeapiset/nf-realtimeapiset-queryprocesscycletime) | 检索指定进程的所有线程的周期时间之和。 |
| [**SetEnvironmentVariable**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-setenvironmentvariable) | 设置当前进程的环境变量的值。 |
| [**SetPriorityClass**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-setpriorityclass) | 设置指定进程的优先级类。 |
| [**SetProcessAffinityMask**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-setprocessaffinitymask) | 为指定进程的线程设置处理器关联掩码。 |
| [**SetProcessAffinityUpdateMode**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-setprocessaffinityupdatemode) | 设置指定进程的相关性更新模式。 |
| [**SetProcessInformation**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-setprocessinformation) | 设置指定进程的信息。 |
| [**SetProcessMitigationPolicy**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Processthreadsapi/nf-processthreadsapi-setprocessmitigationpolicy) | 设置调用进程的缓解策略。 |
| [**SetProcessPriorityBoost**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-setprocesspriorityboost) | 禁用系统暂时提升指定进程的线程优先级的功能。 |
| [**SetProcessRestrictionExemption**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winuser/nf-winuser-setprocessrestrictionexemption) | 使调用进程免受阻止桌面进程与 Windows 应用商店应用环境交互的限制。 开发和调试工具使用此函数。 |
| [**SetProcessShutdownParameters**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-setprocessshutdownparameters) | 设置当前调用进程的关闭参数。 |
| [**SetProcessWorkingSetSize**](https://learn.microsoft.com/zh-cn/windows/desktop/api/memoryapi/nf-memoryapi-setprocessworkingsetsize) | 设置指定进程的最小和最大工作集大小。 |
| [**SetProcessWorkingSetSizeEx**](https://learn.microsoft.com/zh-cn/windows/win32/api/memoryapi/nf-memoryapi-setprocessworkingsetsizeex) | 设置指定进程的最小和最大工作集大小。 |
| [**TerminateProcess**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-terminateprocess) | 终止指定的进程及其所有线程。 |

**进程枚举函数**

以下函数用于枚举进程。

展开表

| **函数** | **说明** |
| --- | --- |
| [**EnumProcesses**](https://learn.microsoft.com/zh-cn/windows/win32/api/psapi/nf-psapi-enumprocesses) | 检索系统中每个进程对象的进程标识符。 |
| [**Process32First**](https://learn.microsoft.com/zh-cn/windows/win32/api/tlhelp32/nf-tlhelp32-process32first) | 检索有关系统快照中遇到的第一个进程的信息。 |
| [**Process32Next**](https://learn.microsoft.com/zh-cn/windows/win32/api/tlhelp32/nf-tlhelp32-process32next) | 检索系统快照中记录的下一个进程的相关信息。 |
| [**WTSEnumerateProcesses**](https://learn.microsoft.com/zh-cn/windows/win32/api/wtsapi32/nf-wtsapi32-wtsenumerateprocessesa) | 检索有关指定终端服务器上的活动进程的信息。 |

**策略函数**

以下函数用于进程范围的策略。

展开表

| **函数** | **说明** |
| --- | --- |
| [**QueryProtectedPolicy**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-queryprotectedpolicy) | 查询与受保护策略关联的值。 |
| [**SetProtectedPolicy**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-setprotectedpolicy) | 设置受保护的策略。 |

**线程函数**

以下函数用于 [线程](https://learn.microsoft.com/zh-cn/windows/win32/procthread/multiple-threads)。

展开表

| **函数** | **说明** |
| --- | --- |
| [**AttachThreadInput**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winuser/nf-winuser-attachthreadinput) | 将一个线程的输入处理机制附加到另一个线程的输入处理机制。 |
| [**CreateRemoteThread**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-createremotethread) | 创建在另一个进程的虚拟地址空间中运行的线程。 |
| [**CreateRemoteThreadEx**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-createremotethreadex) | 创建一个线程，该线程在另一个进程的虚拟地址空间中运行，并选择性地指定扩展属性，例如处理器组相关性。 |
| [**CreateThread**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-createthread) | 创建在调用进程的虚拟地址空间内执行的线程。 |
| [**ExitThread**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-exitthread) | 结束调用线程。 |
| [**GetCurrentThread**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-getcurrentthread) | 检索当前线程的伪句柄。 |
| [**GetCurrentThreadId**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-getcurrentthreadid) | 检索调用线程的线程标识符。 |
| [**GetExitCodeThread**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-getexitcodethread) | 检索指定线程的终止状态。 |
| [**GetProcessIdOfThread**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-getprocessidofthread) | 检索与指定线程关联的进程的进程标识符。 |
| [**GetThreadDescription**](https://learn.microsoft.com/zh-cn/windows/desktop/api/ProcessThreadsApi/nf-processthreadsapi-getthreaddescription) | 通过调用 [**SetThreadDescription**](https://learn.microsoft.com/zh-cn/windows/desktop/api/ProcessThreadsApi/nf-processthreadsapi-setthreaddescription) 检索分配给线程的说明。 |
| [**GetThreadGroupAffinity**](https://learn.microsoft.com/zh-cn/windows/win32/api/processtopologyapi/nf-processtopologyapi-getthreadgroupaffinity) | 检索指定线程的处理器组相关性。 |
| [**GetThreadId**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-getthreadid) | 检索指定线程的线程标识符。 |
| [**GetThreadIdealProcessorEx**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-getthreadidealprocessorex) | 检索指定线程的理想处理器的处理器编号。 |
| [**GetThreadInformation**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-getthreadinformation) | 检索有关指定线程的信息。 |
| [**GetThreadIOPendingFlag**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-getthreadiopendingflag) | 确定指定线程是否有任何 I/O 请求挂起。 |
| [**GetThreadPriority**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-getthreadpriority) | 检索指定线程的优先级值。 |
| [**GetThreadPriorityBoost**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-getthreadpriorityboost) | 检索指定线程的优先级提升控制状态。 |
| [**GetThreadTimes**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-getthreadtimes) | 检索指定线程的计时信息。 |
| [**OpenThread**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-openthread) | 打开现有线程对象。 |
| [**QueryIdleProcessorCycleTime**](https://learn.microsoft.com/zh-cn/windows/win32/api/realtimeapiset/nf-realtimeapiset-queryidleprocessorcycletime) | 检索系统中每个处理器的空闲线程的周期时间。 |
| [**QueryThreadCycleTime**](https://learn.microsoft.com/zh-cn/windows/win32/api/realtimeapiset/nf-realtimeapiset-querythreadcycletime) | 检索指定线程的周期时间。 |
| [**ResumeThread**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-resumethread) | 递减线程的挂起计数。 |
| [**SetThreadAffinityMask**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-setthreadaffinitymask) | 为指定线程设置处理器关联掩码。 |
| [**SetThreadDescription**](https://learn.microsoft.com/zh-cn/windows/desktop/api/ProcessThreadsApi/nf-processthreadsapi-setthreaddescription) | 为线程分配说明。 |
| [**SetThreadGroupAffinity**](https://learn.microsoft.com/zh-cn/windows/win32/api/processtopologyapi/nf-processtopologyapi-setthreadgroupaffinity) | 设置指定线程的处理器组相关性。 |
| [**SetThreadIdealProcessor**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-setthreadidealprocessor) | 指定线程的首选处理器。 |
| [**SetThreadIdealProcessorEx**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-setthreadidealprocessorex) | 设置指定线程的理想处理器，并选择性地检索上一个理想处理器。 |
| [**SetThreadInformation**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-setthreadinformation) | 设置指定线程的信息。 |
| [**SetThreadPriority**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-setthreadpriority) | 设置指定线程的优先级值。 |
| [**SetThreadPriorityBoost**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-setthreadpriorityboost) | 禁用系统暂时提升线程优先级的功能。 |
| [**SetThreadStackGuarantee**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-setthreadstackguarantee) | 设置调用线程的堆栈保证。 |
| [**睡眠**](https://learn.microsoft.com/zh-cn/windows/win32/api/synchapi/nf-synchapi-sleep) | 在指定的时间间隔内暂停当前线程的执行。 |
| [**SleepEx**](https://learn.microsoft.com/zh-cn/windows/win32/api/synchapi/nf-synchapi-sleepex) | 挂起当前线程，直到满足指定的条件。 |
| [**SuspendThread**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-suspendthread) | 挂起指定的线程。 |
| [**SwitchToThread**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-switchtothread) | 导致调用线程执行准备好在当前处理器上运行的另一个线程。 |
| [**TerminateThread**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-terminatethread) | 终止线程。 |
| [**ThreadProc**](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/legacy/ms686736(v=vs.85)) | 一个应用程序定义的函数，用作线程的起始地址。 |
| [**TlsAlloc**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-tlsalloc) | (TLS) 索引分配线程本地存储。 |
| [**TlsFree**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-tlsfree) | 释放 TLS 索引。 |
| [**TlsGetValue**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-tlsgetvalue) | 检索指定 TLS 索引的调用线程的 TLS 槽中的值。 |
| [**TlsSetValue**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-tlssetvalue) | 将值存储在指定 TLS 索引的调用线程的 TLS 槽中。 |
| [**WaitForInputIdle**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winuser/nf-winuser-waitforinputidle) | 等待，直到指定进程等待用户输入且没有输入挂起，或直到超时间隔已过。 |

**进程和线程扩展属性函数**

以下函数用于设置用于创建进程和线程的扩展属性。

展开表

| **函数** | **说明** |
| --- | --- |
| [**DeleteProcThreadAttributeList**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-deleteprocthreadattributelist) | 删除用于创建进程和线程的指定属性列表。 |
| [**InitializeProcThreadAttributeList**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-initializeprocthreadattributelist) | 初始化用于创建进程和线程的指定属性列表。 |
| [**UpdateProcThreadAttribute**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-updateprocthreadattribute) | 汇报指定属性列表中的指定属性，以便创建进程和线程。 |

**WOW64 函数**

以下函数与 [WOW64](https://learn.microsoft.com/zh-cn/windows/win32/winprog64/running-32-bit-applications) 一起使用。

展开表

| **函数** | **说明** |
| --- | --- |
| [**IsWow64Message**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winuser/nf-winuser-iswow64message) | 确定从当前线程队列中读取的最后一条消息是否源自 WOW64 进程。 |
| [**IsWow64Process**](https://learn.microsoft.com/zh-cn/windows/win32/api/wow64apiset/nf-wow64apiset-iswow64process) | 确定指定的进程是否在 WOW64 下运行。 |
| [**IsWow64Process2**](https://learn.microsoft.com/zh-cn/windows/desktop/api/wow64apiset/nf-wow64apiset-iswow64process2) | 确定指定的进程是否在 WOW64 下运行;还返回其他计算机进程和体系结构信息。 |
| [**Wow64SuspendThread**](https://learn.microsoft.com/zh-cn/windows/win32/api/wow64apiset/nf-wow64apiset-wow64suspendthread) | 挂起指定的 WOW64 线程。 |

**作业对象函数**

以下函数用于 [作业对象](https://learn.microsoft.com/zh-cn/windows/win32/procthread/job-objects)。

展开表

| **函数** | **说明** |
| --- | --- |
| [**AssignProcessToJobObject**](https://learn.microsoft.com/zh-cn/windows/win32/api/jobapi2/nf-jobapi2-assignprocesstojobobject) | 将进程与现有作业对象相关联。 |
| [**CreateJobObject**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-createjobobjecta) | 创建或打开作业对象。 |
| [**IsProcessInJob**](https://learn.microsoft.com/zh-cn/windows/win32/api/jobapi/nf-jobapi-isprocessinjob) | 确定进程是否在指定的作业中运行。 |
| [**OpenJobObject**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-openjobobjecta) | 打开现有作业对象。 |
| [**QueryInformationJobObject**](https://learn.microsoft.com/zh-cn/windows/win32/api/jobapi2/nf-jobapi2-queryinformationjobobject) | 从作业对象检索限制和作业状态信息。 |
| [**SetInformationJobObject**](https://learn.microsoft.com/zh-cn/windows/win32/api/jobapi2/nf-jobapi2-setinformationjobobject) | 为作业对象设置限制。 |
| [**TerminateJobObject**](https://learn.microsoft.com/zh-cn/windows/win32/api/jobapi2/nf-jobapi2-terminatejobobject) | 终止当前与作业关联的所有进程。 |
| [**UserHandleGrantAccess**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winuser/nf-winuser-userhandlegrantaccess) | 向具有用户界面限制的作业授予或拒绝对 User 对象的句柄的访问权限。 |

**线程池函数**

以下函数用于 [线程池](https://learn.microsoft.com/zh-cn/windows/win32/procthread/thread-pools)。

展开表

| **函数** | **说明** |
| --- | --- |
| [**CallbackMayRunLong**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-callbackmayrunlong) | 指示回调可能不会快速返回。 |
| [**CancelThreadpoolIo**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-cancelthreadpoolio) | 从 [**StartThreadpoolIo**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-startthreadpoolio) 函数取消通知。 |
| [**CloseThreadpool**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-closethreadpool) | 关闭指定的线程池。 |
| [**CloseThreadpoolCleanupGroup**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-closethreadpoolcleanupgroup) | 关闭指定的清理组。 |
| [**CloseThreadpoolCleanupGroupMembers**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-closethreadpoolcleanupgroupmembers) | 释放指定清理组的成员，等待所有回调函数完成，并选择性地取消任何未完成的回调函数。 |
| [**CloseThreadpoolIo**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-closethreadpoolio) | 释放指定的 I/O 完成对象。 |
| [**CloseThreadpoolTimer**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-closethreadpooltimer) | 释放指定的计时器对象。 |
| [**CloseThreadpoolWait**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-closethreadpoolwait) | 释放指定的等待对象。 |
| [**CloseThreadpoolWork**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-closethreadpoolwork) | 释放指定的工作对象。 |
| [**CreateThreadpool**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-createthreadpool) | 分配要执行回调的新线程池。 |
| [**CreateThreadpoolCleanupGroup**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-createthreadpoolcleanupgroup) | 创建应用程序可用于跟踪一个或多个线程池回调的清理组。 |
| [**CreateThreadpoolIo**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-createthreadpoolio) | 创建新的 I/O 完成对象。 |
| [**CreateThreadpoolTimer**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-createthreadpooltimer) | 创建新的计时器对象。 |
| [**CreateThreadpoolWait**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-createthreadpoolwait) | 创建新的等待对象。 |
| [**CreateThreadpoolWork**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-createthreadpoolwork) | 创建新的工作对象。 |
| [**DestroyThreadpoolEnvironment**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-destroythreadpoolenvironment) | 删除指定的回调环境。 当不再需要回调环境来创建新的线程池对象时，调用此函数。 |
| [**DisassociateCurrentThreadFromCallback**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-disassociatecurrentthreadfromcallback) | 删除当前正在执行的回调函数与启动回调的对象之间的关联。 当前线程将不再被视为代表对象执行回调。 |
| [**FreeLibraryWhenCallbackReturns**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-freelibrarywhencallbackreturns) | 指定在当前回调完成时线程池将卸载的 DLL。 |
| [**InitializeThreadpoolEnvironment**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-initializethreadpoolenvironment) | 初始化回调环境。 |
| [**IsThreadpoolTimerSet**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-isthreadpooltimerset) | 确定当前是否设置了指定的计时器对象。 |
| [**LeaveCriticalSectionWhenCallbackReturns**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-leavecriticalsectionwhencallbackreturns) | 指定当前回调完成时线程池将释放的关键部分。 |
| [**QueryThreadpoolStackInformation**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-querythreadpoolstackinformation) | 检索指定线程池中线程的堆栈保留和提交大小。 |
| [**ReleaseMutexWhenCallbackReturns**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-releasemutexwhencallbackreturns) | 指定在当前回调完成时线程池将释放的互斥体。 |
| [**ReleaseSemaphoreWhenCallbackReturns**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-releasesemaphorewhencallbackreturns) | 指定在当前回调完成时线程池将释放的信号灯。 |
| [**SetEventWhenCallbackReturns**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-seteventwhencallbackreturns) | 指定当前回调完成时线程池将设置的事件。 |
| [**SetThreadpoolCallbackCleanupGroup**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-setthreadpoolcallbackcleanupgroup) | 将指定的清理组与指定的回调环境相关联。 |
| [**SetThreadpoolCallbackLibrary**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-setthreadpoolcallbacklibrary) | 确保只要有未完成的回调，指定的 DLL 就保持加载状态。 |
| [**SetThreadpoolCallbackPersistent**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-setthreadpoolcallbackpersistent) | 指定回调应在持久线程上运行。 |
| [**SetThreadpoolCallbackPool**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-setthreadpoolcallbackpool) | 设置生成回调时要使用的线程池。 |
| [**SetThreadpoolCallbackPriority**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-setthreadpoolcallbackpriority) | 指定回调函数相对于同一线程池中其他工作项的优先级。 |
| [**SetThreadpoolCallbackRunsLong**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-setthreadpoolcallbackrunslong) | 指示与此回调环境关联的回调可能不会快速返回。 |
| [**SetThreadpoolStackInformation**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-setthreadpoolstackinformation) | 设置指定线程池中新线程的堆栈保留和提交大小。 |
| [**SetThreadpoolThreadMaximum**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-setthreadpoolthreadmaximum) | 设置指定线程池可以分配给进程回调的最大线程数。 |
| [**SetThreadpoolThreadMinimum**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-setthreadpoolthreadminimum) | 设置指定线程池必须提供给进程回调的最小线程数。 |
| [**SetThreadpoolTimerEx**](https://learn.microsoft.com/zh-cn/windows/desktop/api/threadpoolapiset/nf-threadpoolapiset-setthreadpooltimerex) | 设置计时器对象。 工作线程在指定的超时到期后调用计时器对象的回调。 |
| [**SetThreadpoolTimer**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-setthreadpooltimer) | 设置计时器对象。 工作线程在指定的超时到期后调用计时器对象的回调。 |
| [**SetThreadpoolWait**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-setthreadpoolwait) | 设置 wait 对象。 工作线程在句柄发出信号后或在指定的超时到期后调用等待对象的回调函数。 |
| [**SetThreadpoolWaitEx**](https://learn.microsoft.com/zh-cn/windows/desktop/api/threadpoolapiset/nf-threadpoolapiset-setthreadpoolwaitex) | 设置 wait 对象。 工作线程在句柄发出信号后或在指定的超时到期后调用等待对象的回调函数。 |
| [**StartThreadpoolIo**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-startthreadpoolio) | 通知线程池，I/O 操作可能针对指定的 I/O 完成对象开始。 在绑定到此对象的文件句柄上的操作完成之后，工作线程调用 I/O 完成对象的回调函数。 |
| [**SubmitThreadpoolWork**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-submitthreadpoolwork) | 将工作对象发布到线程池。 工作线程调用工作对象的回调函数。 |
| [**TpInitializeCallbackEnviron**](https://learn.microsoft.com/zh-cn/windows/desktop/api/winnt/nf-winnt-tpinitializecallbackenviron) | 初始化线程池的回调环境。 |
| [**TpDestroyCallbackEnviron**](https://learn.microsoft.com/zh-cn/windows/desktop/api/winnt/nf-winnt-tpdestroycallbackenviron) | 删除指定的回调环境。 当不再需要回调环境来创建新的线程池对象时，调用此函数。 |
| [**TpSetCallbackActivationContext**](https://learn.microsoft.com/zh-cn/windows/desktop/api/winnt/nf-winnt-tpsetcallbackactivationcontext) | 将激活上下文分配给回调环境。 |
| [**TpSetCallbackCleanupGroup**](https://learn.microsoft.com/zh-cn/windows/desktop/api/winnt/nf-winnt-tpsetcallbackcleanupgroup) | 将指定的清理组与指定的回调环境相关联。 |
| [**TpSetCallbackFinalizationCallback**](https://learn.microsoft.com/zh-cn/windows/desktop/api/winnt/nf-winnt-tpsetcallbackfinalizationcallback) | 指示在回调环境完成时要调用的函数。 |
| [**TpSetCallbackLongFunction**](https://learn.microsoft.com/zh-cn/windows/desktop/api/winnt/nf-winnt-tpsetcallbacklongfunction) | 指示与此回调环境关联的回调可能不会快速返回。 |
| [**TpSetCallbackNoActivationContext**](https://learn.microsoft.com/zh-cn/windows/desktop/api/winnt/nf-winnt-tpsetcallbacknoactivationcontext) | 指示回调环境没有激活上下文。 |
| [**TpSetCallbackPersistent**](https://learn.microsoft.com/zh-cn/windows/desktop/api/winnt/nf-winnt-tpsetcallbackpersistent) | 指定回调应在持久线程上运行。 |
| [**TpSetCallbackPriority**](https://learn.microsoft.com/zh-cn/windows/desktop/api/winnt/nf-winnt-tpsetcallbackpriority) | 指定回调函数相对于同一线程池中其他工作项的优先级。 |
| [**TpSetCallbackRaceWithDll**](https://learn.microsoft.com/zh-cn/windows/desktop/api/winnt/nf-winnt-tpsetcallbackracewithdll) | 确保只要有未完成的回调，指定的 DLL 就保持加载状态。 |
| [**TpSetCallbackThreadpool**](https://learn.microsoft.com/zh-cn/windows/desktop/api/winnt/nf-winnt-tpsetcallbackthreadpool) | 将线程池分配给回调环境。 |
| [**TrySubmitThreadpoolCallback**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-trysubmitthreadpoolcallback) | 请求线程池工作线程调用指定的回调函数。 |
| [**WaitForThreadpoolIoCallbacks**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-waitforthreadpooliocallbacks) | 等待未完成的 I/O 完成回调完成，并选择性地取消尚未开始执行的挂起回调。 |
| [**WaitForThreadpoolTimerCallbacks**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-waitforthreadpooltimercallbacks) | 等待未完成的计时器回调完成，并选择性地取消尚未开始执行的挂起回调。 |
| [**WaitForThreadpoolWaitCallbacks**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-waitforthreadpoolwaitcallbacks) | 等待未完成的等待回调完成，并选择性地取消尚未开始执行的挂起回调。 |
| [**WaitForThreadpoolWorkCallbacks**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoolapiset/nf-threadpoolapiset-waitforthreadpoolworkcallbacks) | 等待未完成的工作回调完成，并选择性地取消尚未开始执行的挂起回调。 |

以下函数是原始 [线程池](https://learn.microsoft.com/zh-cn/windows/win32/procthread/thread-pooling) API 的一部分。

展开表

| **函数** | **说明** |
| --- | --- |
| [**BindIoCompletionCallback**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-bindiocompletioncallback) | 将线程池拥有的 I/O 完成端口与指定的文件句柄相关联。 完成涉及此文件的 I/O 请求后，非 I/O 工作线程将执行指定的回调函数。 |
| [**QueueUserWorkItem**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoollegacyapiset/nf-threadpoollegacyapiset-queueuserworkitem) | 将工作项排队到线程池中的工作线程。 |
| [**RegisterWaitForSingleObject**](https://learn.microsoft.com/zh-cn/windows/win32/api/winbase/nf-winbase-registerwaitforsingleobject) | 指示线程池中的等待线程等待对象。 |
| [**UnregisterWaitEx**](https://learn.microsoft.com/zh-cn/windows/win32/api/threadpoollegacyapiset/nf-threadpoollegacyapiset-unregisterwaitex) | 等待，直到一个或所有指定对象处于信号状态或超时间隔已过。 |

**线程排序服务函数**

以下函数与 [线程排序服务](https://learn.microsoft.com/zh-cn/windows/win32/procthread/thread-ordering-service)一起使用。

展开表

| **函数** | **说明** |
| --- | --- |
| [**AvQuerySystemResponsiveness**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Avrt/nf-avrt-avquerysystemresponsiveness) | 检索多媒体类计划程序服务使用的系统响应能力设置。 |
| [**AvRtCreateThreadOrderingGroup**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Avrt/nf-avrt-avrtcreatethreadorderinggroup) | 创建线程排序组。 |
| [**AvRtCreateThreadOrderingGroupEx**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Avrt/nf-avrt-avrtcreatethreadorderinggroupexa) | 创建线程排序组并将服务器线程与任务相关联。 |
| [**AvRtDeleteThreadOrderingGroup**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Avrt/nf-avrt-avrtdeletethreadorderinggroup) | 删除调用方创建的指定线程排序组。 |
| [**AvRtJoinThreadOrderingGroup**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Avrt/nf-avrt-avrtjointhreadorderinggroup) | 将客户端线程加入线程排序组。 |
| [**AvRtLeaveThreadOrderingGroup**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Avrt/nf-avrt-avrtleavethreadorderinggroup) | 使客户端线程能够离开线程排序组。 |
| [**AvRtWaitOnThreadOrderingGroup**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Avrt/nf-avrt-avrtwaitonthreadorderinggroup) | 允许线程排序组的客户端线程等待，直到它们应执行。 |

**多媒体类计划程序服务函数**

以下函数与 [多媒体类计划程序服务](https://learn.microsoft.com/zh-cn/windows/win32/procthread/multimedia-class-scheduler-service)一起使用。

展开表

| **函数** | **说明** |
| --- | --- |
| [**AvRevertMmThreadCharacteristics**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Avrt/nf-avrt-avrevertmmthreadcharacteristics) | 指示线程不再执行与指定任务关联的工作。 |
| [**AvSetMmMaxThreadCharacteristics**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Avrt/nf-avrt-avsetmmmaxthreadcharacteristicsa) | 将调用线程与指定任务相关联。 |
| [**AvSetMmThreadCharacteristics**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Avrt/nf-avrt-avsetmmthreadcharacteristicsa) | 将调用线程与指定的任务相关联。 |
| [**AvSetMmThreadPriority**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Avrt/nf-avrt-avsetmmthreadpriority) | 调整调用线程相对于执行相同任务的其他线程的线程优先级。 |

**光纤函数**

以下函数用于 [光纤](https://learn.microsoft.com/zh-cn/windows/win32/procthread/fibers)。

展开表

| **函数** | **说明** |
| --- | --- |
| [**ConvertFiberToThread**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-convertfibertothread) | 将当前光纤转换为线程。 |
| [**ConvertThreadToFiber**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-convertthreadtofiber) | 将当前线程转换为光纤。 |
| [**ConvertThreadToFiberEx**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-convertthreadtofiberex) | 将当前线程转换为光纤。 |
| [**CreateFiber**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-createfiber) | 分配一个光纤对象，为其分配一个堆栈，并将执行设置为从指定的起始地址开始。 |
| [**CreateFiberEx**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-createfiberex) | 分配一个光纤对象，为其分配一个堆栈，并将执行设置为从指定的起始地址开始。 |
| [**DeleteFiber**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-deletefiber) | 删除现有光纤。 |
| [**FiberProc**](https://learn.microsoft.com/zh-cn/windows/win32/api/winbase/nc-winbase-pfiber_start_routine) | 与 [**CreateFiber**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-createfiber) 函数一起使用的应用程序定义的函数。 |
| [**FlsAlloc**](https://learn.microsoft.com/zh-cn/windows/win32/api/fibersapi/nf-fibersapi-flsalloc) | (FLS) 索引分配光纤本地存储。 |
| [**FlsFree**](https://learn.microsoft.com/zh-cn/windows/win32/api/fibersapi/nf-fibersapi-flsfree) | 释放 FLS 索引。 |
| [**FlsGetValue**](https://learn.microsoft.com/zh-cn/windows/win32/api/fibersapi/nf-fibersapi-flsgetvalue) | 检索指定 FLS 索引的调用光纤 FLS 槽中的值。 |
| [**FlsSetValue**](https://learn.microsoft.com/zh-cn/windows/win32/api/fibersapi/nf-fibersapi-flssetvalue) | 将值存储在指定 FLS 索引的调用光纤的 FLS 槽中。 |
| [**IsThreadAFiber**](https://learn.microsoft.com/zh-cn/windows/win32/api/fibersapi/nf-fibersapi-isthreadafiber) | 确定当前线程是否为光纤。 |
| [**SwitchToFiber**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-switchtofiber) | 计划光纤。 |

**NUMA 支持函数**

以下函数提供 [NUMA 支持](https://learn.microsoft.com/zh-cn/windows/win32/procthread/numa-support)。

展开表

| **函数** | **说明** |
| --- | --- |
| [**AllocateUserPhysicalPagesNuma**](https://learn.microsoft.com/zh-cn/windows/win32/api/memoryapi/nf-memoryapi-allocateuserphysicalpagesnuma) | 在指定进程的虚拟地址空间中保留或提交内存区域，并为物理内存指定 NUMA 节点。 |
| [**GetLogicalProcessorInformation**](https://learn.microsoft.com/zh-cn/windows/win32/api/sysinfoapi/nf-sysinfoapi-getlogicalprocessorinformation) | 检索有关逻辑处理器和相关硬件的信息。 |
| [**GetNumaAvailableMemoryNode**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-getnumaavailablememorynode) | 检索指定节点中的可用内存量。 |
| [**GetNumaAvailableMemoryNodeEx**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-getnumaavailablememorynodeex) | 检索指定节点中可用内存量作为 USHORT 值。 |
| [**GetNumaHighestNodeNumber**](https://learn.microsoft.com/zh-cn/windows/win32/api/systemtopologyapi/nf-systemtopologyapi-getnumahighestnodenumber) | 检索当前具有最大数目的节点。 |
| [**GetNumaNodeNumberFromHandle**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-getnumanodenumberfromhandle) | 检索与基础设备关联的 NUMA 节点，以获取文件句柄。 |
| [**GetNumaNodeProcessorMask**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-getnumanodeprocessormask) | 检索指定节点的处理器掩码。 |
| [**GetNumaNodeProcessorMaskEx**](https://learn.microsoft.com/zh-cn/windows/win32/api/systemtopologyapi/nf-systemtopologyapi-getnumanodeprocessormaskex) | 检索指定 NUMA 节点的处理器掩码作为 USHORT 值。 |
| [**GetNumaProcessorNode**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-getnumaprocessornode) | 检索指定处理器的节点号。 |
| [**GetNumaProcessorNodeEx**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-getnumaprocessornodeex) | 检索指定逻辑处理器的节点号作为 USHORT 值。 |
| [**GetNumaProximityNode**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-getnumaproximitynode) | 检索指定邻近感应标识符的节点号。 |
| [**GetNumaProximityNodeEx**](https://learn.microsoft.com/zh-cn/windows/win32/api/systemtopologyapi/nf-systemtopologyapi-getnumaproximitynodeex) | 检索节点号作为指定邻近感应标识符的 USHORT 值。 |
| [**VirtualAllocExNuma**](https://learn.microsoft.com/zh-cn/windows/win32/api/memoryapi/nf-memoryapi-virtualallocexnuma) | 在指定进程的虚拟地址空间中保留或提交内存区域，并为物理内存指定 NUMA 节点。 |

**处理器函数**

以下函数用于逻辑处理器和 [处理器组](https://learn.microsoft.com/zh-cn/windows/win32/procthread/processor-groups)。

展开表

| **函数** | **说明** |
| --- | --- |
| [**GetActiveProcessorCount**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-getactiveprocessorcount) | 返回处理器组或系统中的活动处理器数。 |
| [**GetActiveProcessorGroupCount**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-getactiveprocessorgroupcount) | 返回系统中活动处理器组的数目。 |
| [**GetCurrentProcessorNumber**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-getcurrentprocessornumber) | 检索调用此函数期间运行当前线程的处理器数。 |
| [**GetCurrentProcessorNumberEx**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-getcurrentprocessornumberex) | 检索运行调用线程的处理器组和逻辑处理器编号。 |
| [**GetLogicalProcessorInformation**](https://learn.microsoft.com/zh-cn/windows/win32/api/sysinfoapi/nf-sysinfoapi-getlogicalprocessorinformation) | 检索有关逻辑处理器和相关硬件的信息。 |
| [**GetLogicalProcessorInformationEx**](https://learn.microsoft.com/zh-cn/windows/win32/api/sysinfoapi/nf-sysinfoapi-getlogicalprocessorinformationex) | 检索有关逻辑处理器和相关硬件关系的信息。 |
| [**GetMaximumProcessorCount**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-getmaximumprocessorcount) | 返回处理器组或系统可以具有的最大逻辑处理器数。 |
| [**GetMaximumProcessorGroupCount**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-getmaximumprocessorgroupcount) | 返回系统可以具有的最大处理器组数。 |
| [**QueryIdleProcessorCycleTime**](https://learn.microsoft.com/zh-cn/windows/win32/api/realtimeapiset/nf-realtimeapiset-queryidleprocessorcycletime) | 检索系统中每个处理器的空闲线程的周期时间。 |
| [**QueryIdleProcessorCycleTimeEx**](https://learn.microsoft.com/zh-cn/windows/win32/api/realtimeapiset/nf-realtimeapiset-queryidleprocessorcycletimeex) | 检索指定处理器组中每个逻辑处理器上的空闲线程的累积周期时间。 |

**User-Mode计划函数**

以下函数用于用户模式计划 (UMS) 。

展开表

| **函数** | **说明** |
| --- | --- |
| [**CreateUmsCompletionList**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-createumscompletionlist) | 创建 UMS 完成列表。 |
| [**CreateUmsThreadContext**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-createumsthreadcontext) | 创建一个 UMS 线程上下文来表示 UMS 工作线程。 |
| [**DeleteUmsCompletionList**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-deleteumscompletionlist) | 删除指定的 UMS 完成列表。 列表必须为空。 |
| [**DeleteUmsThreadContext**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-deleteumsthreadcontext) | 删除指定的 UMS 线程上下文。 线程必须终止。 |
| [**DequeueUmsCompletionListItems**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-dequeueumscompletionlistitems) | 从指定的 UMS 完成列表中检索 UMS 工作线程。 |
| [**EnterUmsSchedulingMode**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-enterumsschedulingmode) | 将调用线程转换为 UMS 计划程序线程。 |
| [**ExecuteUmsThread**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-executeumsthread) | 运行指定的 UMS 工作线程。 |
| [**GetCurrentUmsThread**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-getcurrentumsthread) | 返回调用 UMS 线程的 UMS 线程上下文。 |
| [**GetNextUmsListItem**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-getnextumslistitem) | 返回 UMS 线程上下文列表中的下一个 UMS 线程上下文。 |
| [**GetUmsCompletionListEvent**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-getumscompletionlistevent) | 检索与指定 UMS 完成列表关联的事件的句柄。 |
| [**GetUmsSystemThreadInformation**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-getumssystemthreadinformation) | 查询指定的线程是 UMS 计划程序线程、UMS 工作线程还是非 UMS 线程。 |
| [**QueryUmsThreadInformation**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-queryumsthreadinformation) | 检索有关指定 UMS 工作线程的信息。 |
| [**SetUmsThreadInformation**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-setumsthreadinformation) | 为指定的 UMS 工作线程设置特定于应用程序的上下文信息。 |
| [*UmsSchedulerProc*](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinNT/nc-winnt-rtl_ums_scheduler_entry_point) | 与 UMS 完成列表关联的应用程序定义的 UMS 计划程序入口点函数。 |
| [**UmsThreadYield**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-umsthreadyield) | 生成对运行调用 UMS 工作线程的 UMS 计划程序线程的控制。 |

**已过时的函数**

* [**NtGetCurrentProcessorNumber**](https://learn.microsoft.com/zh-cn/windows/win32/procthread/ntgetcurrentprocessornumber)
* [**NtQueryInformationProcess**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winternl/nf-winternl-ntqueryinformationprocess)
* [**NtQueryInformationThread**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winternl/nf-winternl-ntqueryinformationthread)
* [**WinExec**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-winexec)
* [**ZwQueryInformationProcess**](https://learn.microsoft.com/zh-cn/windows/win32/procthread/zwqueryinformationprocess)