*作业对象*允许将进程组作为一个单元进行管理。 作业对象是可访问的、安全的、可共享的对象，用于控制与其关联的进程的属性。 针对某个作业对象执行的操作会影响与该作业对象关联的所有进程。 示例包括强制实施工作集大小和进程优先级等限制，或终止与作业关联的所有进程。

* [创建作业](https://learn.microsoft.com/zh-cn/windows/win32/procthread/job-objects#creating-jobs)
* [管理作业中的进程](https://learn.microsoft.com/zh-cn/windows/win32/procthread/job-objects#managing-processes-in-jobs)
* [作业限制和通知](https://learn.microsoft.com/zh-cn/windows/win32/procthread/job-objects#job-limits-and-notifications)
* [作业的资源核算](https://learn.microsoft.com/zh-cn/windows/win32/procthread/job-objects#resource-accounting-for-jobs)
* [管理作业对象](https://learn.microsoft.com/zh-cn/windows/win32/procthread/job-objects#managing-job-objects)
* [管理使用作业对象的进程树](https://learn.microsoft.com/zh-cn/windows/win32/procthread/job-objects#managing-a-process-tree-that-uses-job-objects)

**创建作业**

若要创建作业对象，请使用 [**CreateJobObject**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-createjobobjecta) 函数。 创建作业时，不会与作业关联任何进程。

若要将进程与作业相关联，请使用 [**AssignProcessToJobObject**](https://learn.microsoft.com/zh-cn/windows/win32/api/jobapi2/nf-jobapi2-assignprocesstojobobject) 函数。 进程与作业关联后，无法断开关联。 一个进程可以与嵌套作业层次结构中的多个作业相关联。 有关详细信息，请参阅 [嵌套作业](https://learn.microsoft.com/zh-cn/windows/win32/procthread/nested-jobs)。

**Windows 7、Windows Server 2008 R2、Windows XP SP3、Windows Server 2008、Windows Vista 和 Windows Server 2003：** 一个进程只能与一个作业相关联。 作业不能嵌套。 Windows 8 和 Windows Server 2012中添加了嵌套作业的功能。

调用 [**CreateJobObject**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-createjobobjecta) 函数时，可以为作业对象指定安全描述符。 有关详细信息，请参阅 [作业对象安全性和访问权限](https://learn.microsoft.com/zh-cn/windows/win32/procthread/job-object-security-and-access-rights)。

**管理作业中的进程**

将进程与作业关联后，默认情况下，它使用 [**CreateProcess**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-createprocessa) 创建的任何子进程也会与作业相关联。 (使用 [**Win32\_Process.Create**](https://learn.microsoft.com/zh-cn/windows/win32/cimwin32prov/create-method-in-class-win32-process) 创建的子进程不与 job.) 可以通过为作业设置扩展限制JOB\_OBJECT\_LIMIT\_BREAKAWAY\_OK或JOB\_OBJECT\_LIMIT\_SILENT\_BREAKAWAY\_OK来更改此默认行为。

* 如果作业具有扩展限制JOB\_OBJECT\_LIMIT\_BREAKAWAY\_OK并且父进程是使用 CREATE\_BREAKAWAY\_FROM\_JOB 标志创建的，则父进程的子进程不会与作业关联。
* 如果作业具有JOB\_OBJECT\_LIMIT\_SILENT\_BREAKAWAY\_OK扩展限制，则与作业关联的任何父进程的子进程都与作业不关联。 无需使用 CREATE\_BREAKAWAY\_FROM\_JOB 标志创建父进程。

如果作业是嵌套的，则层次结构中父作业的分离设置会影响子进程是否与层次结构中的另一个作业相关联。 有关详细信息，请参阅 [嵌套作业](https://learn.microsoft.com/zh-cn/windows/win32/procthread/nested-jobs)。

若要确定某个进程是否在作业中运行，请使用 [**IsProcessInJob**](https://learn.microsoft.com/zh-cn/windows/win32/api/jobapi/nf-jobapi-isprocessinjob) 函数。

若要终止当前与作业对象关联的所有进程，请使用 [**TerminateJobObject**](https://learn.microsoft.com/zh-cn/windows/win32/api/jobapi2/nf-jobapi2-terminatejobobject) 函数。

**作业限制和通知**

作业可以对与作业关联的每个进程强制实施工作集大小、进程优先级和作业结束时间限制等限制。 如果与作业关联的进程尝试从作业建立的限制中增加其工作集大小或进程优先级，则函数调用会成功，但会以无提示方式忽略。 作业还可以设置超出限制时触发通知，但允许作业继续运行。

若要为作业设置限制，请使用 [**SetInformationJobObject**](https://learn.microsoft.com/zh-cn/windows/win32/api/jobapi2/nf-jobapi2-setinformationjobobject) 函数。 有关可为作业设置的可能限制的列表，请参阅以下主题：

* [**JOBOBJECT\_BASIC\_LIMIT\_INFORMATION**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinNT/ns-winnt-jobobject_basic_limit_information)
* [**JOBOBJECT\_BASIC\_UI\_RESTRICTIONS**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinNT/ns-winnt-jobobject_basic_ui_restrictions)
* [**JOBOBJECT\_CPU\_RATE\_CONTROL\_INFORMATION**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winnt/ns-winnt-jobobject_cpu_rate_control_information)
* [**JOBOBJECT\_EXTENDED\_LIMIT\_INFORMATION**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinNT/ns-winnt-jobobject_extended_limit_information)
* [**JOBOBJECT\_NOTIFICATION\_LIMIT\_INFORMATION**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinNT/ns-winnt-jobobject_notification_limit_information)

必须为与作业对象关联的每个进程单独设置安全限制。 有关详细信息，请参阅 [进程安全性和访问权限](https://learn.microsoft.com/zh-cn/windows/win32/procthread/process-security-and-access-rights)。

**使用 SP3 和 Windows Server 2003 的 Windows XP：**[**SetInformationJobObject**](https://learn.microsoft.com/zh-cn/windows/win32/api/jobapi2/nf-jobapi2-setinformationjobobject) 函数可用于为与作业对象关联的所有进程设置安全限制。 从 Windows Vista 开始，必须为与作业对象关联的每个进程单独设置安全限制。

如果作业是嵌套的，则层次结构中的父作业会影响对作业强制实施的限制。 有关详细信息，请参阅 [嵌套作业](https://learn.microsoft.com/zh-cn/windows/win32/procthread/nested-jobs)。

如果作业具有关联的 I/O 完成端口，则可以在超出某些作业限制时收到通知。 当超出限制或发生某些其他事件时，系统会将消息发送到完成端口。 若要将完成端口与作业相关联，请将 [**SetInformationJobObject**](https://learn.microsoft.com/zh-cn/windows/win32/api/jobapi2/nf-jobapi2-setinformationjobobject) 函数与作业对象信息类 **JobObjectAssociateCompletionPortInformation** 和 [**指向JOBOBJECT\_ASSOCIATE\_COMPLETION\_PORT**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinNT/ns-winnt-jobobject_associate_completion_port) 结构的指针一起使用。 最好在作业处于非活动状态时执行此操作，以减少在关联完成端口期间状态发生更改的进程丢失通知的可能性。

所有消息都直接从作业发送，就像作业调用 [**PostQueuedCompletionStatus**](https://learn.microsoft.com/zh-cn/windows/win32/api/ioapiset/nf-ioapiset-postqueuedcompletionstatus) 函数一样。 线程必须使用 [**GetQueuedCompletionStatus**](https://learn.microsoft.com/zh-cn/windows/win32/api/ioapiset/nf-ioapiset-getqueuedcompletionstatus) 函数监视完成端口才能接收消息。 请注意，除了使用 **JobObjectNotificationLimitInformation** 信息类设置的限制外，无法保证将消息传递到完成端口;消息到达失败并不一定意味着事件未发生。 使用 **JobObjectNotificationLimitInformation** 设置的限制通知保证到达完成端口。 有关可能的消息列表，请参阅 [**JOBOBJECT\_ASSOCIATE\_COMPLETION\_PORT**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinNT/ns-winnt-jobobject_associate_completion_port)。

**作业的资源核算**

作业对象记录其所有关联进程（包括已终止的进程）的基本记帐信息。 若要检索此记帐信息，请使用 [**QueryInformationJobObject**](https://learn.microsoft.com/zh-cn/windows/win32/api/jobapi2/nf-jobapi2-queryinformationjobobject) 函数。 有关为作业维护的记帐信息列表，请参阅以下主题：

* [**JOBOBJECT\_BASIC\_ACCOUNTING\_INFORMATION**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinNT/ns-winnt-jobobject_basic_accounting_information)
* [**JOBOBJECT\_BASIC\_AND\_IO\_ACCOUNTING\_INFORMATION**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinNT/ns-winnt-jobobject_basic_and_io_accounting_information)

如果作业对象是嵌套的，则每个子作业的记帐信息将聚合在其父作业中。 有关详细信息，请参阅 [嵌套作业](https://learn.microsoft.com/zh-cn/windows/win32/procthread/nested-jobs)。

**管理作业对象**

作业对象的状态设置为在终止其所有进程时发出信号，因为已超出指定的作业结束时间限制。 使用 [**WaitForSingleObject**](https://learn.microsoft.com/zh-cn/windows/win32/api/synchapi/nf-synchapi-waitforsingleobject) 或 [**WaitForSingleObjectEx**](https://learn.microsoft.com/zh-cn/windows/win32/api/synchapi/nf-synchapi-waitforsingleobjectex) 监视此事件的作业对象。

若要获取现有作业对象的句柄，请使用 [**OpenJobObject**](https://learn.microsoft.com/zh-cn/windows/desktop/api/WinBase/nf-winbase-openjobobjecta) 函数，并指定在创建对象时为其指定的名称。 只能打开命名的作业对象。

若要关闭作业对象句柄，请使用 [**CloseHandle**](https://learn.microsoft.com/zh-cn/windows/win32/api/handleapi/nf-handleapi-closehandle) 函数。 当作业的最后一个句柄已关闭且所有关联的进程都已终止时，该作业将被销毁。 但是，如果作业指定了JOB\_OBJECT\_LIMIT\_KILL\_ON\_JOB\_CLOSE标志，则关闭最后一个作业对象句柄会终止所有关联的进程，然后销毁作业对象本身。 如果嵌套作业指定了 JOB\_OBJECT\_LIMIT\_KILL\_ON\_JOB\_CLOSE 标志，则关闭最后一个作业对象句柄将终止与该作业及其层次结构中的子作业关联的所有进程。

**管理使用作业对象的进程树**

从Windows 8和Windows Server 2012开始，应用程序可以使用[嵌套作业](https://learn.microsoft.com/zh-cn/windows/win32/procthread/nested-jobs)来管理使用多个作业对象的进程树。 但是，必须在 Windows 7、Windows Server 2008 R2 或不支持嵌套作业的 Windows 早期版本上运行的应用程序必须以其他方式管理进程树。

如果工具必须管理使用作业对象的进程树，并且无法使用嵌套作业，则该工具和进程树的成员必须合作。 使用以下选项之一：

* 使用JOB\_OBJECT\_LIMIT\_SILENT\_BREAKAWAY\_OK限制。 如果该工具使用此限制，则无法监视整个进程树。 该工具只能监视它添加到作业的进程。 如果这些进程创建子进程，则它们不与作业关联。 在此选项中，子进程可以与其他作业对象相关联。
* 使用JOB\_OBJECT\_LIMIT\_BREAKAWAY\_OK限制。 如果该工具使用此限制，它可以监视整个进程树，但树的任何成员显式脱离树的进程除外。 树的成员可以通过使用 CREATE\_BREAKAWAY\_FROM\_JOB 标志调用 [**CreateProcess**](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-createprocessa) 函数，然后调用 [**AssignProcessToJobObject**](https://learn.microsoft.com/zh-cn/windows/win32/api/jobapi2/nf-jobapi2-assignprocesstojobobject) 函数，在新作业对象中创建子进程。 否则，成员必须处理 **AssignProcessToJobObject** 失败的情况。

如果工具未监视树，则CREATE\_BREAKAWAY\_FROM\_JOB标志不起作用。 因此，这是首选选项，但需要事先了解所监视的进程。

* 通过不设置JOB\_OBJECT\_LIMIT\_BREAKAWAY\_OK和JOB\_OBJECT\_LIMIT\_SILENT\_BREAKAWAY\_OK限制来防止任何类型的分离。 在此选项中，该工具可以监视整个进程树。 但是，如果子进程尝试通过调用 [**AssignProcessToJobObject**](https://learn.microsoft.com/zh-cn/windows/win32/api/jobapi2/nf-jobapi2-assignprocesstojobobject) 将自身或另一个子进程与作业相关联，则调用将失败。 如果进程设计为与特定作业相关联，则此失败可能会阻止进程正常工作。