**命名管道-官网介绍**

A *named pipe* is a named, one-way or duplex pipe for communication between the pipe server and one or more pipe clients. All instances of a named pipe share the same pipe name, but each instance has its own buffers and handles, and provides a separate conduit for client/server communication. The use of instances enables multiple pipe clients to use the same named pipe simultaneously.

命名管道是用于管道服务器和一个或多个管道客户机之间通信的命名、单向或双工管道。命名管道的所有实例共享相同的管道名称，但每个实例都有自己的缓冲区和句柄，并为客户机/服务器通信提供一个独立的管道。实例的使用允许多个管道客户端同时使用相同的命名管道。

Any process can access named pipes, subject to security checks, making named pipes an easy form of communication between related or unrelated processes.

任何进程都可以通过安全检查访问命名管道，这使得命名管道成为相关或不相关进程之间的一种简单通信形式。

Any process can act as both a server and a client, making peer-to-peer communication possible. As used here, the term pipe server refers to a process that creates a named pipe, and the term pipe client refers to a process that connects to an instance of a named pipe. The server-side function for instantiating a named pipe is [**CreateNamedPipe**](https://docs.microsoft.com/en-us/windows/desktop/api/Winbase/nf-winbase-createnamedpipea). The server-side function for accepting a connection is [**ConnectNamedPipe**](https://msdn.microsoft.com/en-us/library/Aa365146(v=VS.85).aspx). A client process connects to a named pipe by using the [**CreateFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-createfilea) or [**CallNamedPipe**](https://docs.microsoft.com/en-us/windows/desktop/api/Winbase/nf-winbase-callnamedpipea) function.

任何进程都可以同时充当服务器和客户机，这使得点对点通信成为可能。这里使用的术语pipe server指的是创建命名管道的进程，而术语pipe client指的是连接到命名管道实例的进程。用于实例化命名管道的服务器端函数是CreateNamedPipe。用于接受连接的服务器端函数是ConnectNamedPipe。客户端进程通过使用CreateFile或CallNamedPipe函数连接到指定的管道。

Named pipes can be used to provide communication between processes on the same computer or between processes on different computers across a network. If the server service is running, all named pipes are accessible remotely. If you intend to use a named pipe locally only, deny access to NT AUTHORITY\NETWORK or switch to local RPC.

命名管道可用于在同一计算机上的进程之间或在网络上不同计算机上的进程之间提供通信。如果服务器服务正在运行，则可以远程访问所有命名管道。如果您打算仅在本地使用命名管道，请拒绝对NT AUTHORITY\NETWORK的访问或切换到本地RPC。

For more information, see the following topics: 有关更多信息，请参阅以下主题:

* [Pipe Names](https://docs.microsoft.com/zh-cn/windows/win32/ipc/pipe-names)

* [Named Pipe Open Modes](https://docs.microsoft.com/zh-cn/windows/win32/ipc/named-pipe-open-modes)
* [Named Pipe Type, Read, and Wait Modes](https://docs.microsoft.com/zh-cn/windows/win32/ipc/named-pipe-type-read-and-wait-modes)
* [Named Pipe Instances](https://docs.microsoft.com/zh-cn/windows/win32/ipc/named-pipe-instances)
* [Named Pipe Operations](https://docs.microsoft.com/zh-cn/windows/win32/ipc/named-pipe-operations)
* [Synchronous and Overlapped Input and Output](https://docs.microsoft.com/zh-cn/windows/win32/ipc/synchronous-and-overlapped-input-and-output)
* [Named Pipe Security and Access Rights](https://docs.microsoft.com/zh-cn/windows/win32/ipc/named-pipe-security-and-access-rights)
* [Impersonating a Named Pipe Client](https://docs.microsoft.com/zh-cn/windows/win32/ipc/impersonating-a-named-pipe-client)
* [Using Pipes](https://docs.microsoft.com/zh-cn/windows/win32/ipc/using-pipes)

# Pipe Names

Each named pipe has a unique name that distinguishes it from other named pipes in the system's list of named objects. A pipe server specifies a name for the pipe when it calls the [**CreateNamedPipe**](https://docs.microsoft.com/en-us/windows/desktop/api/Winbase/nf-winbase-createnamedpipea) function to create one or more instances of a named pipe. Pipe clients specify the pipe name when they call the [**CreateFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-createfilea) or [**CallNamedPipe**](https://docs.microsoft.com/en-us/windows/desktop/api/Winbase/nf-winbase-callnamedpipea) function to connect to an instance of the named pipe.

每个命名管道都有一个惟一的名称，该名称将其与系统的命名对象列表中的其他命名管道区分开来。管道服务器在调用CreateNamedPipe函数来创建命名管道的一个或多个实例时，为管道指定一个名称。管道客户端在调用CreateFile或CallNamedPipe函数连接到指定管道的实例时指定管道名称。

Use the following form when specifying the name of a pipe in the [**CreateFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-createfilea), [**WaitNamedPipe**](https://docs.microsoft.com/en-us/windows/desktop/api/Winbase/nf-winbase-waitnamedpipea), or [**CallNamedPipe**](https://docs.microsoft.com/en-us/windows/desktop/api/Winbase/nf-winbase-callnamedpipea) function:

\\ServerName\pipe\PipeName

在CreateFile、WaitNamedPipe或CallNamedPipe函数中指定管道名称时，请使用以下形式 : \\ServerName\pipe\PipeName

where ServerName is either the name of a remote computer or a period, to specify the local computer. The pipe name string specified by PipeName can include any character other than a backslash, including numbers and special characters. The entire pipe name string can be up to 256 characters long. Pipe names are not case-sensitive.

其中ServerName是远程计算机的名称或句点，用于指定本地计算机。由PipeName指定的管道名称字符串可以包含除反斜杠以外的任何字符，包括数字和特殊字符。整个管道名称字符串最长可达256个字符。管道名称不区分大小写。

The pipe server cannot create a pipe on another computer, so [**CreateNamedPipe**](https://docs.microsoft.com/en-us/windows/desktop/api/Winbase/nf-winbase-createnamedpipea) must use a period for the server name, as shown in the following example.

[\\.\pipe\PipeName](file:///\\.\pipe\PipeName)

管道服务器不能在另一台计算机上创建管道，因此CreateNamedPipe必须为服务器名使用句点，如下面的示例所示 : [\\.\pipe\PipeName](file:///\\.\pipe\PipeName)

A pipe server can provide the pipe name to its pipe clients, so they can connect to the pipe. The pipe client discovers the pipe name from some persistent source, such as a registry entry, a file, or another application. Otherwise, the clients must know the pipe name at compile time.

管道服务器可以将管道名称提供给它的管道客户机，这样它们就可以连接到管道。管道客户端从一些持久源(如注册表项、文件或其他应用程序)发现管道名称。否则，客户机必须在编译时知道管道名称。

# [Named Pipe Open Modes](https://docs.microsoft.com/zh-cn/windows/win32/ipc/named-pipe-open-modes)

The pipe server specifies the pipe access, overlap, and write-through modes in the dwOpenMode parameter of the [**CreateNamedPipe**](https://docs.microsoft.com/en-us/windows/desktop/api/Winbase/nf-winbase-createnamedpipea) function. The pipe clients can specify these open modes for their pipe handles using the [**CreateFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-createfilea) function.

管道服务器在CreateNamedPipe函数的dwOpenMode参数中指定管道访问、重叠和写入通过模式。管道客户机可以使用CreateFile函数为它们的管道句柄指定这些打开模式

## 2.1 Access Mode

Setting the pipe access mode is equivalent to specifying read or write access associated with the pipe server's handles. The following table shows the equivalent generic access right for each access mode you can specify with [**CreateNamedPipe**](https://docs.microsoft.com/en-us/windows/desktop/api/Winbase/nf-winbase-createnamedpipea).

设置管道访问模式相当于指定与管道服务器句柄关联的读或写访问。下表显示了您可以使用CreateNamedPipe指定的每个访问模式的等效通用访问权限。

| **Access mode** | **Equivalent generic access right** |
| --- | --- |
| PIPE\_ACCESS\_INBOUND | GENERIC\_READ |
| PIPE\_ACCESS\_OUTBOUND | GENERIC\_WRITE |
| PIPE\_ACCESS\_DUPLEX | GENERIC\_READ | GENERIC\_WRITE |

If the pipe server creates a pipe with PIPE\_ACCESS\_INBOUND, the pipe is read-only for the pipe server and write-only for the pipe client. If the pipe server creates a pipe with PIPE\_ACCESS\_OUTBOUND, the pipe is write-only for the pipe server and read-only for the pipe client. A pipe created with PIPE\_ACCESS\_DUPLEX is read/write for both the pipe server and the pipe client.

如果管道服务器使用PIPE\_ACCESS\_INBOUND创建管道，则管道服务器的管道是只读的，管道客户机的管道是只读的。如果管道服务器使用PIPE\_ACCESS\_OUTBOUND创建管道，则管道服务器只写管道，管道客户端是只读的。使用PIPE\_ACCESS\_DUPLEX创建的管道用于管道服务器和管道客户机的读/写。

Pipe clients using [**CreateFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-createfilea) to connect to a named pipe must specify an access right in the dwDesiredAccess parameter that is compatible with the access mode specified by the pipe server. For example, a client must specify GENERIC\_READ access to open a handle for a pipe that the pipe server created with PIPE\_ACCESS\_OUTBOUND. The access modes must be the same for all instances of a pipe.

使用CreateFile连接到指定管道的管道客户端必须在dwDesiredAccess参数中指定与管道服务器指定的访问模式兼容的访问权限。例如，客户端必须指定GENERIC\_READ访问权限来打开管道服务器使用PIPE\_ACCESS\_OUTBOUND创建的管道句柄。对于管道的所有实例，访问模式必须相同

To read pipe attributes such as the read mode or blocking mode, the pipe handle must have the FILE\_READ\_ATTRIBUTES access right; to write pipe attributes, the pipe handle must have the FILE\_WRITE\_ATTRIBUTES access right. These access rights can be combined with the generic access right that is appropriate for the pipe: GENERIC\_READ with FILE\_WRITE\_ATTRIBUTES for a read-only pipe, or GENERIC\_WRITE with FILE\_READ\_ATTRIBUTES for a write-only pipe. Restricting access rights in this way provides better security for the pipe.

若要读取管道属性，如读取模式或阻塞模式，管道句柄必须具有FILE\_READ\_ATTRIBUTES访问权限;要写入管道属性，管道句柄必须具有FILE\_WRITE\_ATTRIBUTES访问权限。这些访问权限可以与适用于管道的通用访问权限相结合:GENERIC\_READ与FILE\_WRITE\_ATTRIBUTES用于只读管道，GENERIC\_WRITE与FILE\_READ\_ATTRIBUTES用于只写管道。以这种方式限制访问权限可以为管道提供更好的安全性。

## 2.2 Overlapped Mode

In overlapped mode, functions performing lengthy read, write, and connect operations can return immediately. This enables the thread to perform other operations while a time-consuming operation is executing in the background. To specify overlapped mode, use the FILE\_FLAG\_OVERLAPPED flag. For more information, see [Synchronous and Overlapped Input and Output](https://docs.microsoft.com/zh-cn/windows/win32/ipc/synchronous-and-overlapped-input-and-output).

在重叠模式下，执行冗长的读、写和连接操作的函数可以立即返回。这使线程能够在后台执行耗时的操作时执行其他操作。要指定重叠模式，请使用FILE\_FLAG\_OVERLAPPED标志。有关更多信息，请参见[Synchronous and Overlapped Input and Output](https://docs.microsoft.com/zh-cn/windows/win32/ipc/synchronous-and-overlapped-input-and-output)

The [**CreateFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-createfilea) function allows the pipe client to set overlapped mode (FILE\_FLAG\_OVERLAPPED) for its pipe handles using the dwFlagsAndAttributes parameter.

CreateFile函数允许管道客户端使用dwFlagsAndAttributes参数为其管道句柄设置重叠模式(FILE\_FLAG\_OVERLAPPED)。

## 2.3 Write-Through Mode

Specify write-through mode with FILE\_FLAG\_WRITE\_THROUGH. This mode affects only write operations to byte-type pipes between pipe clients and pipe servers on different computers. In write-through mode, the functions that write to a named pipe do not return until the data is transmitted across the network and into the pipe's buffer on the remote computer. Write-through mode is useful for applications that require synchronization for every write operation.

使用FILE\_FLAG\_WRITE\_THROUGH指定写入模式。此模式仅影响对不同计算机上的管道客户端和管道服务器之间的字节类型管道的写操作。在write-through模式下，在数据通过网络传输到远程计算机上的管道缓冲区之前，对指定管道进行写操作的函数不会返回。write -through模式对于每个写操作都需要同步的应用程序非常有用。

If write-through mode is not enabled, the system enhances the efficiency of network operations by buffering data until a minimum number of bytes have accumulated or until a maximum time period has elapsed. Buffering enables the system to combine multiple write operations into a single network transmission. This means that a write operation can be successfully completed after the system puts the data in the outbound buffer, but before the system transmits it across the network.

如果未启用write-through模式，则系统将通过缓冲数据来提高网络操作的效率，直到积累的字节数达到最小，或者直到达到最大的时间段为止。缓冲使系统能够将多个写操作合并到单个网络传输中。这意味着在系统将数据放入出站缓冲区之后，但在系统跨网络传输之前，可以成功地完成写操作。

The [**CreateFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-createfilea) function allows the pipe client to set write-through mode (FILE\_FLAG\_WRITE\_THROUGH) for its pipe handles using the dwFlagsAndAttributes parameter. The write-through mode of a pipe handle cannot be changed after the pipe handle has been created. The write-through mode can be different for server and client handles to the same pipe instance.

CreateFile函数允许管道客户端使用dwFlagsAndAttributes参数为其管道句柄设置write-through模式(FILE\_FLAG\_WRITE\_THROUGH)。在创建管道句柄之后，无法更改管道句柄的写通模式。对于相同管道实例的服务器和客户端句柄，直通模式可能不同。

A pipe client can use the [**SetNamedPipeHandleState**](https://msdn.microsoft.com/en-us/library/Aa365787(v=VS.85).aspx) function to control the number of bytes and the time-out period before transmission for a pipe on which write-through mode is disabled. For a read-only pipe, the pipe handle must be opened with the GENERIC\_READ and FILE\_WRITE\_ATTRIBUTES access rights.

管道客户端可以使用SetNamedPipeHandleState函数来控制传输之前的字节数和超时时间，以便禁用管道上的写通模式。对于只读管道，必须使用GENERIC\_READ和FILE\_WRITE\_ATTRIBUTES访问权限打开管道句柄。

# [Named Pipe Type, Read, and Wait Modes](https://docs.microsoft.com/zh-cn/windows/win32/ipc/named-pipe-type-read-and-wait-modes)

The pipe server specifies the pipe type mode, read mode, and wait mode in the dwPipeMode parameter of the [**CreateNamedPipe**](https://docs.microsoft.com/en-us/windows/desktop/api/Winbase/nf-winbase-createnamedpipea) function. Pipe clients can specify these pipe modes for their pipe handles using the [**CreateFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-createfilea) function.

管道服务器在CreateNamedPipe函数的dwPipeMode参数中指定管道类型模式、读取模式和等待模式。管道客户机可以使用CreateFile函数为它们的管道句柄指定这些管道模式。

## 3.1 Type Mode

The type mode of a pipe determines how data is written to a named pipe. Data can be transmitted through a named pipe as either a stream of bytes or as a stream of messages. The pipe server specifies the pipe type when calling [**CreateNamedPipe**](https://docs.microsoft.com/en-us/windows/desktop/api/Winbase/nf-winbase-createnamedpipea) to create an instance of a named pipe. The type modes must be the same for all instances of a pipe.

管道的类型模式决定如何将数据写入指定管道。数据可以通过指定管道以字节流或消息流的形式传输。管道服务器在调用CreateNamedPipe创建命名管道实例时指定管道类型。对于管道的所有实例，类型模式必须相同。

To create a byte-type pipe, specify PIPE\_TYPE\_BYTE or use the default value. The data is written to the pipe as a stream of bytes, and the system does not differentiate between the bytes written in different write operations.

要创建字节类型的管道，请指定PIPE\_TYPE\_BYTE或使用默认值。数据以字节流的形式写入管道，系统不区分在不同的写入操作中写入的字节。

To create a message-type pipe, specify PIPE\_TYPE\_MESSAGE. The system treats the bytes written in each write operation to the pipe as a message unit. The system always performs write operations on message-type pipes as if write-through mode were enabled.

要创建消息类型管道，请指定PIPE\_TYPE\_MESSAGE。系统将每个写入操作中写入管道的字节视为一个消息单元。系统总是在消息类型管道上执行写操作，就像启用了写通过模式一样。

## 3.2 Read Mode

The read mode of a pipe determines how data is read from a named pipe. The pipe server specifies the initial read mode for a pipe handle when calling [**CreateNamedPipe**](https://docs.microsoft.com/en-us/windows/desktop/api/Winbase/nf-winbase-createnamedpipea). Data can be read in byte-read mode or message-read mode. A handle to a byte-type pipe can be in byte-read mode only. A handle to a message-type pipe can be in either byte-read or message-read mode. For a message-type pipe, the read mode can be different for server and client handles to the same pipe instance.

管道的读取模式决定如何从指定管道读取数据。管道服务器在调用CreateNamedPipe时为管道句柄指定初始读取模式。可以以字节读取模式或消息读取模式读取数据。字节类型管道的句柄只能是字节读模式。消息类型管道的句柄可以是字节读取模式，也可以是消息读取模式。对于消息类型管道，相同管道实例的服务器和客户端句柄的读模式可能不同。

To create the pipe handle in byte-read mode, specify PIPE\_READMODE\_BYTE or use the default value. Data is read from the pipe as a stream of bytes. A read operation is completed successfully when all available bytes in the pipe are read or when the specified number of bytes is read.

要以字节读取模式创建管道句柄，请指定PIPE\_READMODE\_BYTE或使用默认值。数据以字节流的形式从管道中读取。当读取管道中的所有可用字节或读取指定的字节数时，读取操作将成功完成。

To create the pipe handle in message-read mode, specify PIPE\_READMODE\_MESSAGE. Data is read from the pipe as a stream of messages. A read operation is completed successfully only when the entire message is read. If the specified number of bytes to read is less than the size of the next message, the function reads as much of the message as possible before returning zero (the [**GetLastError**](https://docs.microsoft.com/windows/desktop/api/errhandlingapi/nf-errhandlingapi-getlasterror) function returns ERROR\_MORE\_DATA). The remainder of the message can be read using another read operation.

要在消息读取模式下创建管道句柄，请指定PIPE\_READMODE\_MESSAGE。数据作为消息流从管道中读取。只有在读取整个消息时，才会成功地完成读取操作。如果要读取的指定字节数小于下一条消息的大小，则该函数将在返回0之前尽可能多地读取消息(GetLastError函数将返回ERROR\_MORE\_DATA)。可以使用另一个读操作读取消息的其余部分。

For a pipe client, a pipe handle returned by [**CreateFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-createfilea) is always in byte-read mode initially. Both pipe clients and pipe servers can use the [**SetNamedPipeHandleState**](https://msdn.microsoft.com/en-us/library/Aa365787(v=VS.85).aspx) function to change the read mode of a pipe handle. The pipe handle must have the FILE\_WRITE\_ATTRIBUTES access right.

对于管道客户机，CreateFile返回的管道句柄最初始终处于字节读取模式。管道客户机和管道服务器都可以使用SetNamedPipeHandleState函数来更改管道句柄的读取模式。管道句柄必须具有FILE\_WRITE\_ATTRIBUTES访问权限。

## 3.3 Wait Mode

The wait mode of a pipe handle determines how the [**ReadFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-readfile), [**WriteFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-writefile), and [**ConnectNamedPipe**](https://msdn.microsoft.com/en-us/library/Aa365146(v=VS.85).aspx) functions handle lengthy operations. In blocking-wait mode, the functions wait indefinitely for a process on the other end of the pipe to complete an operation. In nonblocking-wait mode, the functions return immediately in situations that would otherwise require an indefinite wait.

管道句柄的等待模式决定ReadFile、WriteFile和ConnectNamedPipe函数如何处理冗长的操作。在阻塞等待模式下，函数无限期地等待管道另一端的进程完成操作。在非阻塞等待模式下，函数在需要不确定等待的情况下立即返回。

A [**ReadFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-readfile) operation is affected by the wait mode of a pipe handle when the pipe is empty. With a blocking-wait handle, the operation is not completed successfully until data is available from a thread writing to the other end of the pipe. Using a nonblocking-wait handle, the function returns zero immediately, and the [**GetLastError**](https://docs.microsoft.com/windows/desktop/api/errhandlingapi/nf-errhandlingapi-getlasterror) function returns ERROR\_NO\_DATA.

当管道为空时，管道句柄的等待模式会影响ReadFile操作。使用阻塞等待句柄，只有当从线程写入管道另一端的数据可用时，操作才会成功完成。使用非阻塞等待句柄，函数立即返回零，GetLastError函数返回ERROR\_NO\_DATA。

A [**WriteFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-writefile) operation is affected by the wait mode of a pipe handle when there is insufficient space in the pipe's buffer. With a blocking-wait handle, the write operation cannot succeed until sufficient space is created in the buffer by a thread reading from the other end of the pipe. With a nonblocking-wait handle, the write operation returns a nonzero value immediately, without writing any bytes (for a message-type pipe) or after writing as many bytes as the buffer holds (for a byte-type pipe).

当管道缓冲区中空间不足时，管道句柄的等待模式会影响写文件操作。使用阻塞等待句柄，只有通过从管道另一端读取的线程在缓冲区中创建足够的空间，写操作才能成功。使用非阻塞等待句柄，write操作立即返回一个非零值，不需要写入任何字节(对于消息类型管道)，也不需要写入缓冲区持有的所有字节(对于字节类型管道)。

A [**ConnectNamedPipe**](https://msdn.microsoft.com/en-us/library/Aa365146(v=VS.85).aspx) operation is affected by the wait mode of a pipe handle when there is no client connected or waiting to connect to the pipe instance. With a blocking-wait handle, the connect operation does not succeed until a pipe client connects to the pipe instance by calling either the [**CreateFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-createfilea) or [**CallNamedPipe**](https://docs.microsoft.com/en-us/windows/desktop/api/Winbase/nf-winbase-callnamedpipea) function. With a nonblocking-wait handle, the connect operation returns zero immediately, and the [**GetLastError**](https://docs.microsoft.com/windows/desktop/api/errhandlingapi/nf-errhandlingapi-getlasterror) function returns ERROR\_PIPE\_LISTENING.

当没有连接客户端或等待连接到管道实例时，ConnectNamedPipe操作受到管道句柄的等待模式的影响。使用阻塞等待句柄，在管道客户机通过调用CreateFile或CallNamedPipe函数连接到管道实例之前，连接操作不会成功。使用非阻塞等待句柄，connect操作立即返回零，GetLastError函数返回ERROR\_PIPE\_LISTENING。

By default, all named pipe handles returned by the [**CreateNamedPipe**](https://docs.microsoft.com/en-us/windows/desktop/api/Winbase/nf-winbase-createnamedpipea) or [**CreateFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-createfilea) function are created with blocking-wait mode enabled. To create the pipe in nonblocking-wait mode, the pipe server specifies PIPE\_NOWAIT when calling **CreateNamedPipe**.

默认情况下，CreateNamedPipe或CreateFile函数返回的所有命名管道句柄都是在启用阻塞等待模式的情况下创建的。要以非阻塞等待模式创建管道，管道服务器在调用CreateNamedPipe时指定PIPE\_NOWAIT。

Both pipe clients and pipe servers can change a pipe handle's wait mode by specifying either PIPE\_WAIT or PIPE\_NOWAIT in a call to the [**SetNamedPipeHandleState**](https://msdn.microsoft.com/en-us/library/Aa365787(v=VS.85).aspx) function.

管道客户机和管道服务器都可以通过在调用SetNamedPipeHandleState函数时指定PIPE\_WAIT或PIPE\_NOWAIT来更改管道句柄的等待模式。

Note:备注

The nonblocking-wait mode is supported for compatibility with Microsoft LAN Manager version 2.0. This mode should not be used to achieve overlapped input and output (I/O) with named pipes. Overlapped I/O should be used instead, because it enables time-consuming operations to run in the background after the function returns. For more information about overlapped I/O, see [Synchronous and Overlapped Input and Output](https://docs.microsoft.com/zh-cn/windows/win32/ipc/synchronous-and-overlapped-input-and-output).

为了与Microsoft LAN Manager 2.0版本兼容，支持非阻塞等待模式。此模式不应用于通过指定管道实现重叠的输入和输出(I/O)。应该使用Overlapped I/O，因为它使耗时的操作能够在函数返回后在后台运行。有关重叠I/O的更多信息，请参见[Synchronous and Overlapped Input and Output](https://docs.microsoft.com/zh-cn/windows/win32/ipc/synchronous-and-overlapped-input-and-output).

# [Named Pipe Instances](https://docs.microsoft.com/zh-cn/windows/win32/ipc/named-pipe-instances)

The simplest pipe server creates a single instance of a pipe, connects to a single client, communicates with the client, disconnects from the client, closes the pipe handle, and terminates. However, it is more common for a pipe server to communicate with multiple pipe clients. A pipe server could use a single pipe instance to connect with multiple pipe clients by connecting to and disconnecting from each client in sequence, but performance would be poor. The pipe server must create multiple pipe instances to efficiently handle multiple clients simultaneously.

最简单的管道服务器创建管道的单个实例，连接到单个客户机，与客户机通信，断开与客户机的连接，关闭管道句柄，并终止。但是，管道服务器更常见的做法是与多个管道客户机通信。管道服务器可以通过依次连接和断开与每个客户机的连接来使用单个管道实例连接多个管道客户机，但是性能会很差。管道服务器必须创建多个管道实例来有效地同时处理多个客户端。

There are three basic strategies for servicing multiple pipe instances.

服务多个管道实例有三种基本策略。

* Create a separate thread for each instance of the pipe. For an example of a multithreaded pipe server, see [Multithreaded Pipe Server](https://docs.microsoft.com/zh-cn/windows/win32/ipc/multithreaded-pipe-server).

为管道的每个实例创建单独的线程。有关多线程管道服务器的示例，请参见[Multithreaded Pipe Server](https://docs.microsoft.com/zh-cn/windows/win32/ipc/multithreaded-pipe-server)

* Use overlapped operations by specifying an [**OVERLAPPED**](https://docs.microsoft.com/windows/desktop/api/minwinbase/ns-minwinbase-overlapped) structure in the [**ReadFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-readfile), [**WriteFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-writefile), and [**ConnectNamedPipe**](https://msdn.microsoft.com/en-us/library/Aa365146(v=VS.85).aspx) functions. For an example, see [Named Pipe Server Using Overlapped I/O](https://docs.microsoft.com/zh-cn/windows/win32/ipc/named-pipe-server-using-overlapped-i-o).

通过在ReadFile、WriteFile和ConnectNamedPipe函数中指定重叠结构来使用重叠操作。有关示例，请参见[Named Pipe Server Using Overlapped I/O](https://docs.microsoft.com/zh-cn/windows/win32/ipc/named-pipe-server-using-overlapped-i-o).

* Use overlapped operations by using the [**ReadFileEx**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-readfileex) and [**WriteFileEx**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-writefileex) functions, which specify a completion routine to be executed when the operation is completed. For an example, see [Named Pipe Server Using Completion Routines](https://docs.microsoft.com/zh-cn/windows/win32/ipc/named-pipe-server-using-completion-routines).

使用ReadFileEx和WriteFileEx函数来使用重叠操作，这两个函数指定在操作完成时执行的完成例程。有关示例，请参阅[Named Pipe Server Using Completion Routines](https://docs.microsoft.com/zh-cn/windows/win32/ipc/named-pipe-server-using-completion-routines).

The multithreaded pipe server is easiest to write, because the thread for each instance handles communications for a single pipe client. The system allocates processor time to each thread as needed. But each thread uses system resources, which is a disadvantage for a pipe server that handles a large number of clients.

多线程管道服务器是最容易编写的，因为每个实例的线程处理单个管道客户机的通信。系统根据需要为每个线程分配处理器时间。但是每个线程都使用系统资源，这对于处理大量客户机的管道服务器来说是一个缺点。

With a single-threaded server, it is easier to coordinate operations that affect multiple clients, and it is easier to protect shared resources from simultaneous access by multiple clients. The challenge of a single-threaded server is that it requires coordination of overlapped operations to allocate processor time for handling the simultaneous needs of clients.

使用单线程服务器，更容易协调影响多个客户机的操作，也更容易保护共享资源不受多个客户机同时访问。单线程服务器的挑战在于，它需要协调重叠的操作来分配处理器时间来处理客户机的同步需求。

# [Named Pipe Operations](https://docs.microsoft.com/zh-cn/windows/win32/ipc/named-pipe-operations)

The first time the pipe server calls the [**CreateNamedPipe**](https://docs.microsoft.com/en-us/windows/desktop/api/Winbase/nf-winbase-createnamedpipea) function, it uses the *nMaxInstances* parameter to specify the maximum number of instances of the pipe that can exist simultaneously. The server can call **CreateNamedPipe** repeatedly to create additional instances of the pipe, as long as it does not exceed the maximum number of instances. If the function succeeds, each call returns a handle to the server end of a named pipe instance.

As soon as the pipe server creates a pipe instance, a pipe client can connect to it by calling the [**CreateFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-createfilea) or [**CallNamedPipe**](https://docs.microsoft.com/en-us/windows/desktop/api/Winbase/nf-winbase-callnamedpipea) function. If a pipe instance is available, **CreateFile** returns a handle to the client end of the pipe instance. If no instances of the pipe are available, a pipe client can use the [**WaitNamedPipe**](https://docs.microsoft.com/en-us/windows/desktop/api/Winbase/nf-winbase-waitnamedpipea) function to wait until a pipe becomes available.

A pipe server can determine when a pipe client is connected to a pipe instance by calling the [**ConnectNamedPipe**](https://msdn.microsoft.com/en-us/library/Aa365146(v=VS.85).aspx) function. If the pipe handle is in blocking-wait mode, **ConnectNamedPipe** does not return until a client is connected.

Pipe clients and servers can call one of several functions — in addition to [**CallNamedPipe**](https://docs.microsoft.com/en-us/windows/desktop/api/Winbase/nf-winbase-callnamedpipea) — to read from and write to a named pipe. The behavior of these functions depends on the type of pipe and the modes in effect for the specified pipe handle, as follows:

* The [**ReadFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-readfile) and [**WriteFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-writefile) functions can be used with either byte-type or message-type pipes.
* The [**ReadFileEx**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-readfileex) and [**WriteFileEx**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-writefileex) functions can be used with either byte-type or message-type pipes if the pipe handle was opened for overlapped operations.
* The [**PeekNamedPipe**](https://msdn.microsoft.com/en-us/library/Aa365779(v=VS.85).aspx) function can be used to read without removing the contents of either a byte-type pipe or a message-type pipe. **PeekNamedPipe** can also return additional information about the pipe instance.
* The [**TransactNamedPipe**](https://msdn.microsoft.com/en-us/library/Aa365790(v=VS.85).aspx) function can be used with message-type duplex pipes if the pipe handle to the calling process is set to message-read mode. The function writes a request message and reads a reply message in a single operation, enhancing network performance.

The pipe server should not perform a blocking read operation until the pipe client has started. Otherwise, a race condition can occur. This typically occurs when initialization code, such as that of the C run-time library, needs to lock and examine inherited handles.

When a client and server finish using a pipe instance, the server should first call the [**FlushFileBuffers**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-flushfilebuffers) function, to ensure that all bytes or messages written to the pipe are read by the client. **FlushFileBuffers** does not return until the client has read all data from the pipe. The server then calls the [**DisconnectNamedPipe**](https://msdn.microsoft.com/en-us/library/Aa365166(v=VS.85).aspx) function to close the connection to the pipe client. This function makes the client's handle invalid, if it has not already been closed. Any unread data in the pipe is discarded. After the client is disconnected, the server calls the [**CloseHandle**](https://docs.microsoft.com/windows/desktop/api/handleapi/nf-handleapi-closehandle) function to close its handle to the pipe instance. Alternatively, the server can use [**ConnectNamedPipe**](https://msdn.microsoft.com/en-us/library/Aa365146(v=VS.85).aspx) to enable a new client to connect to this instance of the pipe.

A process can retrieve information about a named pipe by calling the [**GetNamedPipeInfo**](https://msdn.microsoft.com/en-us/library/Aa365445(v=VS.85).aspx) function, which returns the type of the pipe, the size of the input and output buffers, and the maximum number of pipe instances that can be created. The [**GetNamedPipeHandleState**](https://docs.microsoft.com/en-us/windows/desktop/api/Winbase/nf-winbase-getnamedpipehandlestatea) function reports on the read and wait modes of a pipe handle, the current number of pipe instances, and additional information for pipes that communicate over a network. The [**SetNamedPipeHandleState**](https://msdn.microsoft.com/en-us/library/Aa365787(v=VS.85).aspx) function sets the read mode and wait modes of a pipe handle. For pipe clients communicating with a remote server, the function also controls the maximum number of bytes to collect or the maximum time to wait before transmitting a message (assuming the client's handle was not opened with write-through mode enabled).

# [Synchronous and Overlapped Input and Output](https://docs.microsoft.com/zh-cn/windows/win32/ipc/synchronous-and-overlapped-input-and-output)

The [**ReadFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-readfile), [**WriteFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-writefile), [**TransactNamedPipe**](https://msdn.microsoft.com/en-us/library/Aa365790(v=VS.85).aspx), and [**ConnectNamedPipe**](https://msdn.microsoft.com/en-us/library/Aa365146(v=VS.85).aspx) functions can perform input and output operations on a pipe either synchronously or asynchronously. When a function runs synchronously, it does not return until the operation it is performing is completed. This means that the execution of the calling thread can be blocked for an indefinite period while it waits for a time-consuming operation to be completed. When a function runs asynchronously, it returns immediately, even if the operation has not been completed. This enables a time-consuming operation to be executed in the background while the calling thread is free to perform other tasks.

Using asynchronous I/O enables a pipe server to use a loop that performs the following steps:

1. Specify multiple event objects in a call to the wait function, and wait for one of the objects to be set to the signaled state.
2. Use the wait function's return value to determine which overlapped operation has finished.
3. Perform the tasks necessary to clean up the completed operation and initiate the next operation for that pipe handle. This can involve starting another overlapped operation for the same pipe handle.

Overlapped operations make it possible for one pipe to read and write data simultaneously and for a single thread to perform simultaneous I/O operations on multiple pipe handles. This enables a single-threaded pipe server to handle communications with multiple pipe clients efficiently. For an example, see [Named Pipe Server Using Overlapped I/O](https://docs.microsoft.com/zh-cn/windows/win32/ipc/named-pipe-server-using-overlapped-i-o).

For a pipe server to use synchronous operations to communicate with more than one client, it must create a separate thread for each pipe client so that one or more threads can run while other threads are waiting. For an example of a multithreaded pipe server that uses synchronous operations, see [Multithreaded Pipe Server](https://docs.microsoft.com/zh-cn/windows/win32/ipc/multithreaded-pipe-server).

## 6.1 Enabling Asynchronous Operation

The [**ReadFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-readfile), [**WriteFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-writefile), [**TransactNamedPipe**](https://msdn.microsoft.com/en-us/library/Aa365790(v=VS.85).aspx), and [**ConnectNamedPipe**](https://msdn.microsoft.com/en-us/library/Aa365146(v=VS.85).aspx) functions can be performed asynchronously only if you enable overlapped mode for the specified pipe handle and specify a valid pointer to an [**OVERLAPPED**](https://docs.microsoft.com/windows/desktop/api/minwinbase/ns-minwinbase-overlapped) structure. If the **OVERLAPPED** pointer is **NULL**, the function return value can incorrectly indicate that the operation has been completed. Therefore, it is strongly recommended that if you create a handle with FILE\_FLAG\_OVERLAPPED and want asynchronous behavior, you should always specify a valid **OVERLAPPED** structure.

The **hEvent** member of the specified [**OVERLAPPED**](https://docs.microsoft.com/windows/desktop/api/minwinbase/ns-minwinbase-overlapped) structure must contain a handle to a manual-reset event object. This is a synchronization object created by the [**CreateEvent**](https://docs.microsoft.com/windows/desktop/api/synchapi/nf-synchapi-createeventa) function. The thread that initiates the overlapped operation uses the event object to determine when the operation has finished. You should not use the pipe handle for synchronization when performing simultaneous operations on the same handle because there is no way of knowing which operation's completion caused the pipe handle to be signaled. The only reliable technique for performing simultaneous operations on the same pipe handle is to use a separate **OVERLAPPED** structure with its own event object for each operation. For more information about event objects, see [Synchronization](https://docs.microsoft.com/windows/desktop/Sync/synchronization).

Also, you can be notified when an overlapped operation completes by using the [**GetQueuedCompletionStatus**](https://docs.microsoft.com/windows/desktop/api/ioapiset/nf-ioapiset-getqueuedcompletionstatus) or [**GetQueuedCompletionStatusEx**](https://docs.microsoft.com/windows/desktop/FileIO/getqueuedcompletionstatusex-func) functions. In this case, you do not need to assign the manual-reset event in the [**OVERLAPPED**](https://docs.microsoft.com/windows/desktop/api/minwinbase/ns-minwinbase-overlapped) structure, and the completion happens against the pipe handle in the same way as an asynchronous read or write operation. For more information, see [I/O Completion Ports](https://docs.microsoft.com/windows/desktop/FileIO/i-o-completion-ports).

When [**ReadFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-readfile), [**WriteFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-writefile), [**TransactNamedPipe**](https://msdn.microsoft.com/en-us/library/Aa365790(v=VS.85).aspx), and [**ConnectNamedPipe**](https://msdn.microsoft.com/en-us/library/Aa365146(v=VS.85).aspx) operations are performed asynchronously, one of the following occurs:

* If the operation is complete when the function returns, the return value indicates the success or failure of the operation. If an error occurs, the return value is zero and the [**GetLastError**](https://docs.microsoft.com/windows/desktop/api/errhandlingapi/nf-errhandlingapi-getlasterror) function returns something other than ERROR\_IO\_PENDING.
* If the operation has not finished when the function returns, the return value is zero and [**GetLastError**](https://docs.microsoft.com/windows/desktop/api/errhandlingapi/nf-errhandlingapi-getlasterror) returns ERROR\_IO\_PENDING. In this case, the calling thread must wait until the operation has finished. The calling thread must then call the [**GetOverlappedResult**](https://docs.microsoft.com/windows/desktop/api/ioapiset/nf-ioapiset-getoverlappedresult) function to determine the results.

## 6.2 Using Completion Routines

The [**ReadFileEx**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-readfileex) and [**WriteFileEx**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-writefileex) functions provide another form of overlapped I/O. Unlike the overlapped [**ReadFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-readfile) and [**WriteFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-writefile) functions, which use an event object to signal completion, the extended functions specify a completion routine. A completion routine is a function that is queued for execution when the read or write operation is finished. The completion routine is not executed until the thread that called **ReadFileEx** and **WriteFileEx** starts an alertable wait operation by calling one of the [alertable wait functions](https://docs.microsoft.com/windows/desktop/Sync/wait-functions) with the fAlertable parameter set to **TRUE**. In an alertable wait operation, the functions also return when a **ReadFileEx** or **WriteFileEx** completion routine is queued for execution. A pipe server can use the extended functions to perform a sequence of read and write operations for each client that connects to it. Each read or write operation in the sequence specifies a completion routine, and each completion routine initiates the next step in the sequence. For an example, see [Named Pipe Server Using Completion Routines](https://docs.microsoft.com/zh-cn/windows/win32/ipc/named-pipe-server-using-completion-routines).

# [Named Pipe Security and Access Rights](https://docs.microsoft.com/zh-cn/windows/win32/ipc/named-pipe-security-and-access-rights)

Windows security enables you to control access to named pipes. For more information about security, see [Access-Control Model](https://docs.microsoft.com/windows/desktop/SecAuthZ/access-control-model).

You can specify a [security descriptor](https://docs.microsoft.com/windows/desktop/SecAuthZ/security-descriptors) for a named pipe when you call the [**CreateNamedPipe**](https://docs.microsoft.com/en-us/windows/desktop/api/Winbase/nf-winbase-createnamedpipea) function. The security descriptor controls access to both client and server ends of the named pipe. If you specify **NULL**, the named pipe gets a default security descriptor. The ACLs in the default security descriptor for a named pipe grant full control to the LocalSystem account, administrators, and the creator owner. They also grant read access to members of the Everyone group and the anonymous account.

To retrieve a named pipe's security descriptor, call the [**GetSecurityInfo**](https://docs.microsoft.com/windows/desktop/api/aclapi/nf-aclapi-getsecurityinfo) function. To change the security descriptor of a named pipe, call the [**SetSecurityInfo**](https://docs.microsoft.com/windows/desktop/api/aclapi/nf-aclapi-setsecurityinfo) function.

When a thread calls [**CreateNamedPipe**](https://docs.microsoft.com/en-us/windows/desktop/api/Winbase/nf-winbase-createnamedpipea) to open a handle to the server end of an existing named pipe, the system performs an access check before returning the handle. The access check compares the thread's access token and the requested [access rights](https://docs.microsoft.com/windows/desktop/SecAuthZ/access-rights-and-access-masks) against the DACL in the named pipe's security descriptor. In addition to the requested access rights, the DACL must allow the calling thread FILE\_CREATE\_PIPE\_INSTANCE access to the named pipe.

Similarly, when a client calls the [**CreateFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-createfilea) or [**CallNamedPipe**](https://docs.microsoft.com/en-us/windows/desktop/api/Winbase/nf-winbase-callnamedpipea) function to connect to the client end of a named pipe, the system performs an access check before granting access to the client.

The handle returned by the [**CreateNamedPipe**](https://docs.microsoft.com/en-us/windows/desktop/api/Winbase/nf-winbase-createnamedpipea) function always has SYNCHRONIZE access. It also has GENERIC\_READ, GENERIC\_WRITE, or both, depending on the open mode of the pipe. The following are the access rights for each open mode.

| **Open mode** | **Access rights** |
| --- | --- |
| PIPE\_ACCESS\_DUPLEX (0x00000003) | FILE\_GENERIC\_READ, FILE\_GENERIC\_WRITE, and SYNCHRONIZE |
| PIPE\_ACCESS\_INBOUND (0x00000001) | FILE\_GENERIC\_READ and SYNCHRONIZE |
| PIPE\_ACCESS\_OUTBOUND (0x00000002) | FILE\_GENERIC\_WRITE and SYNCHRONIZE |

FILE\_GENERIC\_READ access for a named pipe combines the rights to read data from the pipe, read pipe attributes, read extended attributes, and read the pipe's DACL.

FILE\_GENERIC\_WRITE access for a named pipe combines the rights to write data to the pipe, append data to it, write pipe attributes, write extended attributes, and read the pipe's DACL. Because FILE\_APPEND\_DATA and FILE\_CREATE\_PIPE\_INSTANCE have the same definition, so FILE\_GENERIC\_WRITE enables permission to create the pipe. To avoid this problem, use the individual rights instead of using FILE\_GENERIC\_WRITE.

You can request the ACCESS\_SYSTEM\_SECURITY access right to a named pipe object if you want to read or write the object's SACL. For more information, see [Access-Control Lists (ACLs)](https://docs.microsoft.com/windows/desktop/SecAuthZ/access-control-lists) and [SACL Access Right](https://docs.microsoft.com/windows/desktop/SecAuthZ/sacl-access-right).

To prevent remote users or users on a different terminal services session from accessing a named pipe, use the logon SID on the DACL for the pipe. The logon SID is used in run-as logons as well; it is the SID used to protect the per-session object namespace. For more information, see [Getting the Logon SID in C++](https://docs.microsoft.com/previous-versions/aa446670(v=vs.85)).

# [Impersonating a Named Pipe Client](https://docs.microsoft.com/zh-cn/windows/win32/ipc/impersonating-a-named-pipe-client)

Impersonation is the ability of a thread to execute in a security context different from that of the process that owns the thread. Impersonation enables the server thread to perform actions on behalf of the client, but within the limits of the client's security context. The client typically has some lesser level of access rights. For more information, see [Impersonation](https://docs.microsoft.com/windows/desktop/SecAuthZ/client-impersonation).

A named pipe server thread can call the [**ImpersonateNamedPipeClient**](https://docs.microsoft.com/windows/desktop/api/namedpipeapi/nf-namedpipeapi-impersonatenamedpipeclient) function to assume the access token of the user connected to the client end of the pipe. For example, a named pipe server can provide access to a database or file system to which the pipe server has privileged access. When a pipe client sends a request to the server, the server impersonates the client and attempts to access the protected database. The system then grants or denies the server's access, based on the security level of the client. When the server is finished, it uses the [**RevertToSelf**](https://docs.microsoft.com/windows/desktop/api/securitybaseapi/nf-securitybaseapi-reverttoself) function to restore its original security token.

The [impersonation level](https://docs.microsoft.com/windows/desktop/SecAuthZ/impersonation-levels) determines the operations the server can perform while impersonating the client. By default, a server impersonates at the SecurityImpersonation impersonation level. However, when the client calls the [**CreateFile**](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-createfilea) function to open a handle to the client end of the pipe, the client can use the SECURITY\_SQOS\_PRESENT flag to control the server's impersonation level.

# [Using Pipes](https://docs.microsoft.com/zh-cn/windows/win32/ipc/using-pipes)

The following examples demonstrate the use of pipes to pass a continuous stream of data between processes:

* [Multithreaded pipe server](https://docs.microsoft.com/zh-cn/windows/win32/ipc/multithreaded-pipe-server)
* [Named pipe server using overlapped I/O](https://docs.microsoft.com/zh-cn/windows/win32/ipc/named-pipe-server-using-overlapped-i-o)
* [Named pipe server using completion routines](https://docs.microsoft.com/zh-cn/windows/win32/ipc/named-pipe-server-using-completion-routines)
* [Named pipe client](https://docs.microsoft.com/zh-cn/windows/win32/ipc/named-pipe-client)
* [Transactions on named pipes](https://docs.microsoft.com/zh-cn/windows/win32/ipc/transactions-on-named-pipes)