What's the difference between an asynchronous PIPE_WAIT pipe and a PIPE_NOWAIT pipe?

devblogs.microsoft.com/oldnewthing/20110114-00

January 14, 2011

Raymond Chen

When you operate on named pipes, you have a choice of opening them in `PIPE_WAIT` mode or `PIPE_NOWAIT` mode. When you read from a `PIPE_WAIT` pipe, the read blocks until data becomes available in the pipe. When you read from a `PIPE_NOWAIT` pipe, then the read completes immediately even if there is no data in the pipe. But how is this different from a `PIPE_WAIT` pipe opened in asynchronous mode by passing `FILE_FLAG_OVERLAPPED`? The difference is in when the I/O is deemed to have completed. When you issue an overlapped read against a `PIPE_WAIT` pipe, the call to `ReadFile` returns immediately, but the completion actions do not occur until there is data available in the pipe. (*Completion actions* are things like setting the event, running the completion routine, or queueing a completion to an I/O completion port.) On the other hand, when you issue a read against a `PIPE_NOWAIT` pipe, the call to `ReadFile` returns immediately *with completion*—if the pipe is empty, the read completes with a read of zero bytes and the error `ERROR_NO_DATA`. Here's a timeline, for people who prefer tables.

Event	**Asynchronous `PIPE_WAIT`**	**`PIPE_NOWAIT`**
pipe initially empty		
ReadFile	Returns immediately with `ERROR_IO_PENDING`	Returns immediately with `ERROR_NO_DATA` I/O completes with 0 bytes
time passes		
Data available	I/O completes with *n* > 0 bytes	

If you use the `PIPE_NOWAIT` flag, then the only way to know whether there is data is to poll for it. There is no way to be notified when data becomes available.

As the documentation notes, `PIPE_NOWAIT` remains solely for compatibility with LAN Manager 2.0. Since the only way to use pipes created as `PIPE_NOWAIT` is to poll them, this is obviously not a recommended model for a multitasking operating system.

Raymond Chen

Follow