Looking at the problem at the wrong level: Closing a process's stdin

devblogs.microsoft.com/oldnewthing/20110706-00

July 6, 2011

Raymond Chen

A customer was having trouble manipulating the stdin stream that was given to a process.

> How do you simulate sending Ctrl+Z to a hidden console process programmatically?
>
> I am using `RedirectStandardInput` and want to send the console a Ctrl+Z. I've tried sending ASCII code 26, but that doesn't work. `GenerateConsoleCtrlEvent` supports Ctrl+C and Ctrl+Break but not Ctrl+Z.
>
> Here's what I'm doing, but it doesn't work:
>
> ```
> ProcessStartInfo info = new ProcessStartInfo(@"...");
> info.CreateNoWindow = true;
> info.RedirectStandardError = true;
> info.RedirectStandardOutput = true;
> info.RedirectStandardInput = true;
> info.UseShellExecute = false;
> Process p = Process.Start(info);
> // 0x1A is ASCII code of Ctrl+Z but it does not work
> p.StandardInput.WriteLine("\x1A");
> ```

The customer was kind enough to do more than simply ask the question. The customer set up the scenario and even provided a code fragment that illustrates the problem. Which is good, because the original question *was the wrong question*.

The customer asked about simulating typing Ctrl+Z to a console, but what they actually doing was sending a character to stdin; they weren't sending it to a console. In fact, the way they created the process, *there is no console at all*.

The customer confused stdin with consoles. It's true that Ctrl+Z is the convention used by console windows to indicate that stdin should be closed. But that is hardly any consolation when you took control of stdin yourself and are not using a console window to manage it.

It's like saying, "Normally, when I want somebody to take my order, I pull into a parking space and turn on my headlights, and somebody will come out. But I can't get it to work."

— Um, that's because you pulled into your own driveway.

Ctrl+Z is a convention used by console windows to indicate that stdin should be closed, but if you said "I am going to manage stdin myself," then you aren't using a console window, and that convention carries no weight. If you write a Ctrl+Z to the process's stdin, it will simply read a Ctrl+Z. But since you are managing stdin yourself, you can do it yourself: Just take the stream you set as the process's stdin and close it.

Exercise: Perhaps you can answer this related question from a different customer:

> I am trying to send a Ctrl+C (SIGINT) to a process.
>
> ```
> CurrentProcess = new Process();
> CurrentProcess.StartInfo.FileName = "foo.exe";
> CurrentProcess.StartInfo.UseShellExecute = false;
> CurrentProcess.StartInfo.RedirectStandardInput = true;
> StandardInputWriter = CurrentProcess.StandardInput;
> char c = '\u0003';
> StandardInputWriter.Write(c);
> StandardInputWriter.Flush();
> StandardInputWriter.Close();
> ```
>
> If I launch the process from a command prompt and type Ctrl+C, it flushes its output and terminates, but when I start it from within my application and send it a Ctrl+C via the code above, the process is still running. How do I send a Ctrl+C to a process?

Raymond Chen

Follow