

Re: system calls

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.mfc/2004-11/0450.html>

From: Joseph M. Newcomer (*newcomer_at_flounder.com*)

Date: 11/05/04

Date: Fri, 05 Nov 2004 11:49:00 -0500

Yes. If you are writing a Windows App, you are expected to program it using the Windows API. If you are writing a POSIX app, use the POSIX API, and be prepared for whatever it does under Windows. This is an MFC newsgroup, and we write programs using the native Windows API and that API as wrapped by MFC. POSIX questions would be answered in a POSIX newsgroup.

If you are writing a Windows program, it is so locked into the Windows API that adding CreateProcess is in no way going to decrease its portability. Writing a simple subroutine to scan a directory is not a massive intellectual effort.

Rewrite them back to C system calls? WHAT C system calls? Have you ever tried to write a directory iterator in native Unix? I did, and It Is Not A Pretty Sight.

Calling the Microsoft command shell with the text of a Microsoft command to delete a directory is "portable"? Or is this a use of the word "portable" of which I have been previously unaware?

Dozens? Why should a tiny number like "dozens" be of any concern at all? I have HUNDREDS of Windows programs, and there is no indication that Windows will ever die. After all, do the numbers: how many POSIX-compliant systems exist in the world today? How many Windows systems? There is something like a 20:1 ratio in favor of Windows. I may be wrong however; the ratio may be much larger.

As far as I know, POSIX does not support windowing. Now, if I rewrite my Windows programs to a Unix API, should I write to Gnome, or KDE, or what? Or just raw X windows? What version of X? Which of the mutually-incompatible Unix windowing systems should I interface to?

Obviously, in thinking about portability, people who think Unix (or Linux) is the answer have not paid attention to issues that are common in Unix/Linux such as "I have this application that is written to Gnome, and I do a copy-to-clipboard operation. I run this KDE application and try to do a paste. Why doesn't it work?" Unix can't even get something as simple as a clipboard right. Why should I care about being compatible with a system this chaotic?

I spent years trying to get "portable" Unix code to run on some other "compatible" Unix system. Just System V to BSD 4.x was a challenge, never mind the couple dozen other mutually incompatible Unix clones that existed. When people talk about "portable" Unix

code, I often wonder if they have been ingesting strange and illegal substances, because I never met two Unix systems that were compatible, except at the most trivial and superficial level for the most trivial and simplistic code. I remember one piece of code I worked on that had SPECIFIC CONDITIONALS for EACH OF THIRTY DIFFERENT UNIX SYSTEMS. And

you think CreateProcess is a problem? We had conditionals that worked around compiler bugs because the vendors refused to fix their compilers! In those days, the attitude of the vendors was "tough. You buy our system, you use our compiler! Bugs? Not our problem!".

Have you ever LOOKED at the GNU C compiler to see how many conditionals have to be in it to make it "portable"? I have. It is frightening. I ported some GNU code for a client a couple years ago, and it reinforced my belief that Unix is still in a completely chaotic state. I had more problems figuring out which of the conditionals to set to make it compile for Windows than I had changing the dozen or so lines required to interface to the operating system. I've successfully ported X-Windows code to Windows in three days, retaining a code base of 80% or better, and needed only six conditionals in two header files to make it port. If you take your MFC application and port it to some other OS, exactly HOW is CreateProcess going to complicate this task beyond a reasonable effort?

And what is POSIX? Exactly WHICH of the multitude of mutually incompatible POSIX standards are you referring to? I used to write critiques to various POSIX groups about the problems with their "standard" being incompatible with other POSIX standards. This was back in the days when I thought Unix was what we were going to be stuck with forever, and thought I had to do something to lessen the overall misery of trying to use it.

Frankly, nothing pleased me more than to abandon Unix.

The interesting thing is that since I abandoned Unix in the mid-1980s, I have never had the slightest interest in getting a set of "Unix tools" so my MS-DOS, Win16 or Win32 system resembled in any way at all the Unix shell. I'm even hard-pressed to think of a Unix program, other than awk, that I would find useful today. My editor has a built-in "grep" far superior to the actual Unix program, so I haven't even had a standalone grep in over a decade.

Scripting? Shell? Why would anyone in their right mind want to write shell scripts? Exactly HOW MANY instances of ` are needed to make sure that this construct does not expand in the wrong place? Oh, I'm sorry, I refactored the code and now all the callers need ONE MORE level of ` to avoid expanding at the wrong level. THIS is a clean and elegant programming methodology?

I can't recall when I have written a "script" longer than about five lines of straight code, usually with no parameters at all. And it is usually to run awk and direct the output to a file.

If you need to write scripts, use a real language. Perl or Python are good contemporary candidates. But the shell scripting languages in Unix were amateur attempts done by people who were largely clueless about language design, and they had so many features and kludges patched in over the years that the result was an incompatible mess. I would get "portable" scripts for shells I'd never heard of, or worse still, for shells I had that were different implementations of what someone once thought some other implementation might have intended to do. Scripts were not portable. Programs were not portable. GUI interfaces

were not portable. Let's see, what WAS portable? I'm thinking...thinking...ah, LOGON was portable! No, wait, some systems accepted no more than 8-character passwords and threw the extras away. Some accepted longer passwords. Some idiots insisted that I use my three-letter initials, a few intelligent sysops let me use a real username of my choice, some insisted I use the local standard, whatever it was. Some allowed me to choose my password; some changed it every month to something I HAD to write down to remember. No, not even logon was portable...

CreateProcess is so far down in the noise that it is not even worth discussing.

joe

On 4 Nov 2004 00:11:58 -0800, kochkarev@pisem.net (Sergey Kochkarev) wrote:

*>Joseph M. Newcomer <newcomer@flounder.com> wrote in message
news:<46tgo0pom92fnng1lsfs3dmgnk1r45k7t3@4ax.com>...*

*>> You mean how to keep the rather quaint and antiquated "system()" call from popping up a
>> DOS box? The answer is simple. Stop using something that obsolete and use CreateProcess to
>> create a process without a visible console.*

>

*>That's what MS wants us to do – use its own API instead of following
>POSIX. When you have dosens of Win32 programs and Windows dies what
>would you do? Rewrite them back using C system calls? How much effort
>would it take?*

>

*>Kevin, what about using cygwin and mingw? These create standard
>Unix-like environment atop of Windows and pretty good for system
>utilities.*

Joseph M. Newcomer [MVP]

email: newcomer@flounder.com

Web: <http://www.flounder.com>

MVP Tips: http://www.flounder.com/mvp_tips.htm