

Re: should these be fixed for python 2.4?

Source: <http://coding.derkeiler.com/Archive/Python/comp.lang.python/2004-10/0167.html>

From: Donn Cave (donn_at_u.washington.edu)

Date: 10/01/04

Date: Fri, 01 Oct 2004 11:48:41 -0700

In article <yfshdpemlb4.fsf@black4.ex.ac.uk>, Alexander Schmolck <a.schmolck@gmx.net> wrote:

> *Two smallish things that have been bugging me; I'm not sure whether they're
> actually broken or not, but anyway here it goes:*

> 1. `os.system` (and `co`): Shouldn't `os.system`'s signature really be
> `os.system(cmd, arg1, arg2,...)` rather than `os.system(some_string)`?

...

> *BTW, if I'm not mistaken it's something that perl got right a long time
> ago.*

>

> `perl -e "system('touch', 'a b c', 'd')"`

> `ll`

> *total 0*

> `-rw-r--r-- 1 aschmolc phd 0 Oct 1 17:48 a b c`

> `-rw-r--r-- 1 aschmolc phd 0 Oct 1 17:48 d`

To elaborate on Martin v. Lewis' response, the `spawnv()` function, and presumably Perl's `system` as shown above, are distinctly different from `system(3)`, in that they execute the command directly with the indicated parameters, where `system()` executes the shell to interpret shell command text.

In my opinion Python got this much righter than Perl when it used a different name. Unlike the case with `Popen3`, where the same function represents 2 different APIs, both the string and the list – I see later in your message you may be among the people who have been led astray by that.

> *P.S. speaking of running external processes -- I'm also unaware of a good,
> platform-independent (or even just unix) way to run a process that possibly
> accepts input, but will just continue on EOF from stdin and getting back
> stdout and stderr (if supported) as strings and the exit status.*

>

> *I currently use this to run e.g. latex, but it doesn't seem to work*

comp.lang.python: Re: should these be fixed for python 2.4?

> *absolutely*

> *reliably and is unix only. I'm pretty sure there must be a better way.*

Yes, but either you use disk files or it's more work.

The unreliability, I imagine is due to limitations of pipes' device buffer space. You're trying to execute the whole thing within those limits – you wait() for process exit before you have read a byte. It would be better to read output first, since that will terminate when the process exits, and then wait() to retrieve the status. A fully reliable version needs to select on the two pipes to make sure that the error pipe doesn't fill up while you're trying to read all the output. Use os.read on the file descriptor so you're on the same page with select().

Or on the other hand, temporary disk files don't have these problems and are of course more portable.

Donn Cave, donn@u.washington.edu