Ruby trunk - Feature #13095

[PATCH] io.c (rb_f_syscall): remove deprecation notice

01/02/2017 04:52 AM - normalperson (Eric Wong)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

io.c (rb_f_syscall): remove deprecation notice

New, perhaps experimental syscalls appear all the time which
may not and never be be supported by the system C library.

Furthermore, on common GNU/Linux platforms, kernel development
and releases happens at a much faster pace than GNU C library
(glibc) development. In my experience, is more common for users
to run recent, custom-built Linux kernels than a recent,
custom-built glibc. (This is likely because Linux upstream has
better built-in tooling for distro package management integration
than glibc upstream).

So, ruby should support users who want to deal with the latest
and greatest syscalls in the kernel without having them wait for
C library support.

History

#1 - 01/02/2017 01:14 PM - kosaki (Motohiro KOSAKI)

I disagree.

http://man7.org/linux/man-pages/man2/syscall.2.html clearly explains a caller must care Architecture-specific requirements. But current this interface have no way Ruby core care alignment etc.

syscall is misdesigned interface. It assumed old good "integer = pointer = 32bit" world like it born.

#2 - 01/02/2017 10:21 PM - normalperson (Eric Wong)

kosaki.motohiro@gmail.com wrote:

> I disagree.
>
> http://man7.org/linux/man-pages/man2/syscall.2.html clearly explains a caller must care Architecture-specific requirements. But current this interface have no way Ruby core care alignment etc.

We should expect users of this to be able to read and follow
documentation. We already have a warning about it. I prefer we
allow it to improve, and allow users to shoot themselves in the
foot if necessary. As I've said before: I don't want Ruby to be
a nanny scripting language.

Anyways, I plan on having this release GVL for slow syscalls
and maybe other small improvements.

#3 - 01/03/2017 11:59 AM - kosaki (Motohiro KOSAKI)

> We should expect users of this to be able to read and follow documentation. We already have a warning about it. I prefer we allow it to improve, and allow users to shoot themselves in the foot if necessary.

This is unrelated what I said. I said current interface (both C level and Ruby level) is not designed well and then, user have no way to write correct code. Ruby code have no way to care about memory alignment. I didn't only talk about just careless user.

> As I've said before: I don't want Ruby to be
> a nanny scripting language.

I agree. But I don't think this patch is a right direction.
I'm curious. Why you don't like to make proper new C extension?
C program have a way to treat syscall(2) interface correctly.

> Anyways, I plan on having this release GVL for slow syscalls
> and maybe other small improvements.

Wait. This? Which patch do you talk about? As far as I can see, current attached patch only remove a warning. Doesn't it?

#4 - 01/03/2017 06:31 PM - normalperson (Eric Wong)

kosaki.motohiro@gmail.com wrote:

> > We should expect users of this to be able to read and follow
> > documentation. We already have a warning about it. I prefer we
> > allow it to improve, and allow users to shoot themselves in the
> > foot if necessary.
>
> This is unrelated what I said. I said current interface (both C level and Ruby level) is not designed well and then, user have no way to write correct code. Ruby code have no way to care about memory alignment. I didn't only talk about just careless user.

We can support String#pack for those cases, I think (or add support).
I haven't tried the incompatible functions/arch, yet.

> > As I've said before: I don't want Ruby to be
> > a nanny scripting language.
>
> I agree. But I don't think this patch is a right direction.
> I'm curious. Why you don't like to make proper new C extension?
> C program have a way to treat syscall(2) interface correctly.

C extensions require user to either have a compiler, or install
a pre-built binary. Both have extra distribution and
installation costs which are high for small (old i686) systems
and users with limited bandwidth/storage. For those reasons, I
prefer to use scripting as much as possible. Over the past
year or so, I've been trying to avoid programming in any
compiled languages.

> > Anyways, I plan on having this release GVL for slow syscalls
> > and maybe other small improvements.
>
> Wait. This? Which patch do you talk about? As far as I can see, current attached patch only remove a warning. Doesn't it?

This patch to undeprecate, first. I have not implemented GVL
release, yet; I will if I can get this undeprecated.

#5 - 01/03/2017 07:01 PM - normalperson (Eric Wong)

Eric Wong normalperson@yhbt.net wrote:

> We can support String#pack

I meant Array#pack ENOCOFFEE :x

#6 - 01/03/2017 11:27 PM - shevegen (Robert A. Heiler)

"Over the past year or so, I've been trying to avoid programming in any compiled languages."

I am going a bit off-topic here but I could not resist to add one comment to this.

I try this even more generally - avoid any other language than Ruby itself at all costs. ;-)

#7 - 01/09/2017 04:25 AM - kernigh (George Koehler)

Kernel.syscall was a wrapper around syscall() or __syscall() in libc, but it didn't use the correct types for arguments and return values. I suggest to use Fiddle to call syscall() or __syscall() with the correct types for your system call. Here is a quick example for getdents64() in 32-bit PowerPC Linux.

```
require 'fiddle'

libc = Fiddle.dlopen('libc.so.6')
libc_syscall = Fiddle::Function.new(
  libc['syscall'],
  [Fiddle::TYPE_INT, Fiddle::TYPE_INT, Fiddle::TYPE_VOIDP,
   Fiddle::TYPE_SIZE_T],
  Fiddle::TYPE_INT)
getdents64 = 202

buf = "X" * 4096
fd = IO.sysopen('/dev', IO::RDONLY)
ret = libc_syscall.(getdents64, fd, buf, buf.size)
ret < 0 and raise SystemCallError, Fiddle.last_error

while ret > 0
  ino, off, reclen, type, name = buf.unpack('QqSCZ*')
  puts name
  buf.slice!(0...reclen)
  ret -= reclen
end
IO.new(fd).close
```

```
$ ruby -v
ruby 2.1.5p273 (2014-11-13) [powerpc-linux-gnu]
$ ruby exam.rb | head
.
..
hidraw1
vcsa6
vcs6
vcsa5
vcs5
vcsa4
vcs4
vcsa3
```

#8 - 01/09/2017 10:21 AM - normalperson (Eric Wong)

xkernigh@netscape.net wrote:

> Kernel.syscall was a wrapper around syscall() or __syscall() in libc, but it didn't use the correct types for arguments and return values. I suggest to use Fiddle to call syscall() or __syscall() with the correct types for your system call. Here is a quick example for getdents64() in 32-bit PowerPC Linux.
>
> ```
> require 'fiddle'
>
> libc = Fiddle.dlopen('libc.so.6')
> ```

I encounter more problems with calling dlopen properly, even for libc, more than I encounter with mismatched syscall numbers.

I prefer to avoid Fiddle/FFI because I find it's dynamic loading interface even less predictable across systems than using bare syscall numbers.

Also, libffi isn't available or costs extra download time on some systems. syscall has been in Ruby for ages, and I've relied on it's equivalent builtin function in Perl every single day for many years, now. It's never failed me as a Perl user. Once a syscall-using script works, it'll work on that system as long as that scripting language is installed. No need for extra libraries to install or potentially get broken during upgrades.

#9 - 03/13/2017 02:39 PM - shyouhei (Shyouhei Urabe)

We looked at this issue in today's developer meeting.

We see that using fiddle to wrap libc is too complicated. At the same time however, it is true that memory alignment is not exposed to ruby level so it is not possible right now to write a "correct" code that calls this interface. The current situation is sort of "nice in theory, broken in practice".

So the problem is: do we want to make it complete, like by adding alignment support? Maybe other things are also needed like SYS_foobar definitions. Should we do that in-core? or in new library (like, say, ext/syscall)? Or in reverse, should we just delete this as the warning says now? We at the meeting could not conclude.

#10 - 03/26/2017 06:21 AM - normalperson (Eric Wong)

ext/syscall with numbers defined would be nice, but I think optional.

Mainly, I want to be able to continue trying new syscalls before
they have libc support.

Files

0001-io.c-rb_f_syscall-remove-deprecation-notice.patch	1.33 KB	01/02/2017	normalperson (Eric Wong)