Ruby trunk - Bug #10908

Addrinfo.new appears to ignore the afamily argument when using a String for sockaddr

02/26/2015 11:36 AM - yorickpeterse (Yorick Peterse)

Status:	Rejected		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.2.0p0 (2014-12-25 revision 49005) [x86_64-linux]	**Backport:**	2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN

Description

When creating a new Addrinfo instance the new class method appears to ignore the 2nd (afamily) argument and always sets it to AF_INET. Some examples:

```
Socket::AF_INET # => 2

Addrinfo.new(Socket.sockaddr_in(80, 'localhost')).afamily                    # => 2
Addrinfo.new(Socket.sockaddr_in(80, 'localhost'), Socket::AF_INET6).afamily  # => 2
Addrinfo.new(Socket.sockaddr_in(80, 'localhost'), Socket::PF_UNSPEC).afamily # => 2
Addrinfo.new(Socket.sockaddr_in(80, 'localhost'), Socket::AF_IPX).afamily    # => 2
Addrinfo.new(Socket.sockaddr_in(80, 'localhost'), Socket::AF_LOCAL).afamily  # => 2
```

Is this correct, or is this a bug?

The documentation states the following about this argument:

> family is specified as an integer to specify the protocol family such as Socket::PF_INET. It can be a symbol or a string which is the constant name with or without PF_ prefix such as :INET, :INET6, :UNIX, "PF_INET", etc. If omitted, PF_UNSPEC is assumed.

I looked at the tests but couldn't find any specific examples of this behaviour. For Rubinius I ended up writing the following Rubyspec which currently passes on both Rubinius (you'll need the Git master branch for this, for now) and Ruby 2.2: http://git.io/AhpQ. I've attached the spec as well in case the URL stops working.

Related issues:

Related to Ruby trunk - Misc #10907: Documentation of Addrinfo.new suggests d...	**Rejected**	**02/26/2015**

History

#1 - 02/26/2015 01:06 PM - brightbits (Michael Baldry)

Yorick Peterse wrote:

> When creating a new Addrinfo instance the new class method appears to ignore the 2nd (afamily) argument and always sets it to AF_INET. Some examples:
>
> ```
> Socket::AF_INET # => 2
>
> Addrinfo.new(Socket.sockaddr_in(80, 'localhost')).afamily                    # => 2
> Addrinfo.new(Socket.sockaddr_in(80, 'localhost'), Socket::AF_INET6).afamily  # => 2
> Addrinfo.new(Socket.sockaddr_in(80, 'localhost'), Socket::PF_UNSPEC).afamily # => 2
> Addrinfo.new(Socket.sockaddr_in(80, 'localhost'), Socket::AF_IPX).afamily    # => 2
> Addrinfo.new(Socket.sockaddr_in(80, 'localhost'), Socket::AF_LOCAL).afamily  # => 2
> ```
>
> Is this correct, or is this a bug?
>
> The documentation states the following about this argument:
>
> > family is specified as an integer to specify the protocol family such as Socket::PF_INET. It can be a symbol or a string which is the constant name with or without PF_ prefix such as :INET, :INET6, :UNIX, "PF_INET", etc. If omitted, PF_UNSPEC is assumed.

> I looked at the tests but couldn't find any specific examples of this behaviour.
> For Rubinius I ended up writing the following Rubyspec which currently passes on
> both Rubinius (you'll need the Git master branch for this, for now) and Ruby
> 2.2: http://git.io/AhpQ. I've attached the spec as well in case the URL stops
> working.

I have confirmed this on ruby 2.1.2p95 (2014-05-08 revision 45877) [x86_64-darwin13.0]

After some investigation, the afamily, address family, is always coming from the Scoket.sockaddr_in() which deafults to AF_INET/AF_INET6, but the parameter you are passing in is set on the pfamily, protocol family... I'm not in a good position to understand the difference and what this means, but this test passes:

```
def test_addrinfo_new_family
addr = Socket.sockaddr_in(80, 'localhost')
assert_equal(Socket::AF_LOCAL, Addrinfo.new(addr, Socket::AF_LOCAL).pfamily)
assert_equal(Socket::AF_IPX, Addrinfo.new(addr, Socket::AF_IPX).pfamily)
assert_equal(Socket::AF_INET, Addrinfo.new(addr, Socket::AF_INET).pfamily)
assert_equal(Socket::AF_INET6, Addrinfo.new(addr, Socket::AF_INET6).pfamily)
end
```

where as you expect

```
def test_addrinfo_new_family
addr = Socket.sockaddr_in(80, 'localhost')
assert_equal(Socket::AF_LOCAL, Addrinfo.new(addr, Socket::AF_LOCAL).afamily)
assert_equal(Socket::AF_IPX, Addrinfo.new(addr, Socket::AF_IPX).afamily)
assert_equal(Socket::AF_INET, Addrinfo.new(addr, Socket::AF_INET).afamily)
assert_equal(Socket::AF_INET6, Addrinfo.new(addr, Socket::AF_INET6).afamily)
end
```

to pass and it doesn't, always returning the value fromthe addr family

I'm not sure if anything needs to be done, will let someone with a better understanding of sockets comment.

update: reading the documentation again, it mentions PROTOCOL FAMILY, not ADDRESS FAMILY. So I think this is working as expected.

#2 - 02/27/2015 10:05 AM - yorickpeterse (Yorick Peterse)

@Michael

Hm, good catch, I hadn't thought of that. In that case it indeed looks like I
was misunderstanding the documentation. I'll work on the specs to
confirm or rule this out.

#3 - 03/01/2015 12:37 AM - akr (Akira Tanaka)

- Status changed from Open to Rejected

afamily returns the family in sockaddr.
2nd argument for Addrinfo.new doesn't affect afamily.

pfamily (and 2nd argument for Addrinfo.new) corresponds to ai_family field of struct addrinfo and will be used for 1st argument of socket().

afamily (and first 1 or 2 bytes in 1st argument for Addrinfo.new) corresponds to sa_family field of struct sockaddr and will be used for bind() or connect().

#4 - 03/01/2015 12:37 AM - akr (Akira Tanaka)

- Related to Misc #10907: Documentation of Addrinfo.new suggests default family of PF_UNSPEC while in practise it appears to be AF_INET added

Files

initialize_spec.rb	3.68 KB	02/26/2015	yorickpeterse (Yorick Peterse)