Ruby - Feature #2706

optional reverse_lookup argument for IPSocket#{addr,peeraddr} and Socket.getaddrinfo

02/02/2010 06:25 PM - nobu (Nobuyoshi Nakada)

Status:	Closed
Priority:	Normal
Assignee:	
Target version:	

Description

=begin
Hi,

I propose adding an optional argument to enable/disable reverse
lookup.

Index: ext/socket/ipsocket.c

--- ext/socket/ipsocket.c (revision 26539)
+++ ext/socket/ipsocket.c (working copy)
@@ -130,7 +130,30 @@ rsock_init_inetsock(VALUE sock, VALUE re
}

+static ID id_numeric, id_hostname;
+
+int
+rsock_revlookup_flag(VALUE revlookup, int *norevlookup)
+{
+#define return_norevlookup(x) {*norevlookup = x; return 1;}

- ID id;
- switch (revlookup) {
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- Check_Type(revlookup, T_SYMBOL);
- id = SYM2ID(revlookup);
- if (id == id_numeric) return_norevlookup(1);
- if (id == id_hostname) return_norevlookup(0);
- rb_raise(rb_eArgError, "invalid reverse_lookup flag: :%s", rb_id2name(id));
- }
- return 0;
  +#undef return_norevlookup
  +}

/*

- call-seq:
  - ipsocket.addr => [address_family, port, hostname, numeric_address]
  - ipsocket.addr([reverse_lookup]) => [address_family, port, hostname, numeric_address]
  - Returns the local address as an array which contains

@@ -138,29 +161,38 @@ rsock_init_inetsock(VALUE sock, VALUE re

- hostname is obtained from numeric_address using reverse lookup.
- If ipsocket.do_not_reverse_lookup is true,
- If +reverse_lookup+ is false, or it is nil and
- ipsocket.do_not_reverse_lookup is true,
- hostname is same as numeric_address.
- See Socket.getaddrinfo for possible values as +reverse_lookup+.
- TCPSocket.open("www.ruby-lang.org", 80) {|sock|
- `p sock.addr #=> ["AF_INET", 49429, "hal", "192.168.0.128"]`
- `p sock.addr(true)  #=> ["AF_INET", 49429, "hal", "192.168.0.128"]`
- `p sock.addr(false) #=> ["AF_INET", 49429, "192.168.0.128", "192.168.0.128"]`
- `p sock.addr(:hostname)  #=> ["AF_INET", 49429, "hal", "192.168.0.128"]`
- `p sock.addr(:numeric)   #=> ["AF_INET", 49429, "192.168.0.128", "192.168.0.128"]`
- }

*/
static VALUE
-ip_addr(VALUE sock)
+ip_addr(int argc, VALUE *argv, VALUE sock)
{
rb_io_t *fptr;
struct sockaddr_storage addr;
socklen_t len = sizeof addr;

- int norevlookup;

GetOpenFile(sock, fptr);

- if (argc < 1 || !rsock_revlookup_flag(argv[0], &norevlookup))

- norevlookup = fptr->mode & FMODE_NOREVLOOKUP;
if (getsockname(fptr->fd, (struct sockaddr*)&addr, &len) < 0)
rb_sys_fail("getsockname(2)");

- return rsock_ipaddr((struct sockaddr*)&addr, fptr->mode & FMODE_NOREVLOOKUP);

- return rsock_ipaddr((struct sockaddr*)&addr, norevlookup);
}

/*

- call-seq:
- ipsocket.peeraddr => [address_family, port, hostname, numeric_address]
- ipsocket.peeraddr([reverse_lookup]) => [address_family, port, hostname, numeric_address]
- Returns the remote address as an array which contains
@@ -168,21 +200,34 @@ ip_addr(VALUE sock)
- It is defined for connection oriented socket such as TCPSocket.

- hostname is obtained from numeric_address using reverse lookup.
- If +reverse_lookup+ is false, or it is nil and
- ipsocket.do_not_reverse_lookup is true,
- hostname is same as numeric_address.
- See Socket.getaddrinfo for possible values as +reverse_lookup+.
- TCPSocket.open("www.ruby-lang.org", 80) {|sock|
- p sock.peeraddr #=> ["AF_INET", 80, "carbon.ruby-lang.org", "221.186.184.68"]
- p sock.peeraddr(true) #=> ["AF_INET", 80, "221.186.184.68", "221.186.184.68"]
- p sock.peeraddr(false) #=> ["AF_INET", 80, "221.186.184.68", "221.186.184.68"]
- p sock.peeraddr(:hostname) #=> ["AF_INET", 80, "carbon.ruby-lang.org", "221.186.184.68"]
- p sock.peeraddr(:numeric) #=> ["AF_INET", 80, "221.186.184.68", "221.186.184.68"]
- }

```
*/
static VALUE
-ip_peeraddr(VALUE sock)
+ip_peeraddr(int argc, VALUE *argv, VALUE sock)
{
rb_io_t *fptr;
struct sockaddr_storage addr;
socklen_t len = sizeof addr;
```

- int norevlookup;

```
GetOpenFile(sock, fptr);
```

- if (argc < 1 || !rsock_revlookup_flag(argv[0], &norevlookup))
- norevlookup = fptr->mode & FMODE_NOREVLOOKUP;
  if (getpeername(fptr->fd, (struct sockaddr*)&addr, &len) < 0)
  rb_sys_fail("getpeername(2)");
- return rsock_ipaddr((struct sockaddr*)&addr, fptr->mode & FMODE_NOREVLOOKUP);
- return rsock_ipaddr((struct sockaddr*)&addr, norevlookup);
  }

```
@@ -244,9 +289,11 @@ Init_ipsocket(void)
{
rb_cIPSocket = rb_define_class("IPSocket", rb_cBasicSocket);
```

- rb_define_method(rb_cIPSocket, "addr", ip_addr, 0);
- rb_define_method(rb_cIPSocket, "peeraddr", ip_peeraddr, 0);
- rb_define_method(rb_cIPSocket, "addr", ip_addr, -1);
- rb_define_method(rb_cIPSocket, "peeraddr", ip_peeraddr, -1);
  rb_define_method(rb_cIPSocket, "recvfrom", ip_recvfrom, -1);
  rb_define_singleton_method(rb_cIPSocket, "getaddress", ip_s_getaddress, 1);
  rb_undef_method(rb_cIPSocket, "getpeereid");
- id_numeric = rb_intern_const("numeric");

- id_hostname = rb_intern_const("hostname");
  }
  Index: ext/socket/socket.c
  ===
  --- ext/socket/socket.c (revision 26539)
  +++ ext/socket/socket.c (working copy)
  @@ -871,5 +871,5 @@ sock_gethostname(VALUE obj)

static VALUE
-make_addrinfo(struct addrinfo *res0)
+make_addrinfo(struct addrinfo *res0, int norevlookup)
{
VALUE base, ary;
@@ -881,5 +881,5 @@ make_addrinfo(struct addrinfo *res0)
base = rb_ary_new();
for (res = res0; res; res = res->ai_next) {

- ary = rsock_ipaddr(res->ai_addr, rsock_do_not_reverse_lookup);

- ary = rsock_ipaddr(res->ai_addr, norevlookup);
  if (res->ai_canonname) {
  RARRAY_PTR(ary)[2] = rb_str_new2(res->ai_canonname);
  @@ -956,5 +956,5 @@ sock_s_gethostbyaddr(int argc, VALUE *ar
  t = rsock_family_arg(family);
  }
  -#ifdef INET6
  +#ifdef AF_INET6
  else if (RSTRING_LEN(addr) == 16) {
  t = AF_INET6;
  @@ -1070,5 +1070,5 @@ sock_s_getservbyport(int argc, VALUE *ar
  /

- call-seq:

  - Socket.getaddrinfo(nodename, servname[, family[, socktype[, protocol[, flags]]]]) => array

  - Socket.getaddrinfo(nodename, servname[, family[, socktype[, protocol[, flags[, reverse_lookup]]]]]) => array

  - Obtains address information for *nodename*:*servname*.
    @@ -1091,12 +1091,19 @@ sock_s_getservbyport(int argc, VALUE *ar

  - **["AF_INET", 0, "localhost", "127.0.0.1", 2, 3, 0] # PF_INET/SOCK_RAW/IPPROTO_IP**

  - *reverse_lookup* directs the form of the third element, and should

  - be true, false, nil or a Symbol.

  - true, :hostname : returns hostname string using reverse lookup, which may take a time.

  - false, :numeric : returns numeric host address, without reverse lookup.

  - nil : obey to the current +do_not_reverse_lookup+ flag.
    */
    static VALUE
    sock_s_getaddrinfo(int argc, VALUE *argv)
    {

- VALUE host, port, family, socktype, protocol, flags, ret;

- VALUE host, port, family, socktype, protocol, flags, ret, revlookup;

struct addrinfo hints, *res;
- int norevlookup;
- rb_scan_args(argc, argv, "24", &host, &port, &family, &socktype, &protocol, &flags);
- rb_scan_args(argc, argv, "25", &host, &port, &family, &socktype, &protocol, &flags, &revlookup);

MEMZERO(&hints, struct addrinfo, 1);
@@ -1112,7 +1119,10 @@ sock_s_getaddrinfo(int argc, VALUE *argv
hints.ai_flags = NUM2INT(flags);
}

- if (NIL_P(revlookup) || !rsock_revlookup_flag(revlookup, &norevlookup)) {
- norevlookup = rsock_do_not_reverse_lookup;
- }
res = rsock_getaddrinfo(host, port, &hints, 0);

- ret = make_addrinfo(res);
- ret = make_addrinfo(res, norevlookup);
freeaddrinfo(res);
return ret;

--
Nobu Nakada
=end

History

#1 - 02/03/2010 10:57 AM - matz (Yukihiro Matsumoto)

=begin
Hi,

In message "Re: [ruby-core:28007] [Feature:trunk] optional reverse_lookup argument for IPSocket#{addr,peeraddr} and Socket.getaddrinfo"
on Tue, 2 Feb 2010 18:24:49 +0900, Nobuyoshi Nakada nobu@ruby-lang.org writes:

|I propose adding an optional argument to enable/disable reverse
|lookup.

+1, but the document should mention the default value more clearly.
Currently, the behavior is not clear when reverse_lookup argument is
omitted.

```
matz.
```

=end

#2 - 02/03/2010 06:04 PM - nobu (Nobuyoshi Nakada)

=begin
Hi,

At Wed, 3 Feb 2010 10:57:23 +0900,
Yukihiro Matsumoto wrote in [ruby-core:28023]:

> |I propose adding an optional argument to enable/disable reverse
> |lookup.
>
> +1, but the document should mention the default value more clearly.
> Currently, the behavior is not clear when reverse_lookup argument is
> omitted.

Any other suggestions?

Index: ext/socket/ipsocket.c

--- ext/socket/ipsocket.c (revision 26546)
+++ ext/socket/ipsocket.c (working copy)
@@ -130,37 +130,70 @@ rsock_init_inetsock(VALUE sock, VALUE re
}

+static ID id_numeric, id_hostname;
+
+int
+rsock_revlookup_flag(VALUE revlookup, int *norevlookup)
+{
+#define return_norevlookup(x) {*norevlookup = x; return 1;}

- ID id;
- switch (revlookup) {
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- Check_Type(revlookup, T_SYMBOL);
- id = SYM2ID(revlookup);
- if (id == id_numeric) return_norevlookup(1);
- if (id == id_hostname) return_norevlookup(0);
- rb_raise(rb_eArgError, "invalid reverse_lookup flag: :%s", rb_id2name(id));
- }
- return 0;
  +#undef return_norevlookup
  +}

/*

- call-seq:
  - ipsocket.addr => [address_family, port, hostname, numeric_address]
  - ipsocket.addr([reverse_lookup]) => [address_family, port, hostname, numeric_address]
  - Returns the local address as an array which contains
  - address_family, port, hostname and numeric_address.
  - If +reverse_lookup+ is +true+ or +:hostname+,
  - hostname is obtained from numeric_address using reverse lookup.
  - If ipsocket.do_not_reverse_lookup is true,
  - Or if it is +false+, or +:numeric+,
  - hostname is same as numeric_address.
  - Or if it is +nil+ or ommitted, obeys to +ipsocket.do_not_reverse_lookup+.
  - See +Socket.getaddrinfo+ also.
  - TCPSocket.open("www.ruby-lang.org", 80) {|sock|
  - `p sock.addr #=> ["AF_INET", 49429, "hal", "192.168.0.128"]`
  - `p sock.addr(true)  #=> ["AF_INET", 49429, "hal", "192.168.0.128"]`
  - `p sock.addr(false) #=> ["AF_INET", 49429, "192.168.0.128", "192.168.0.128"]`
  - `p sock.addr(:hostname)  #=> ["AF_INET", 49429, "hal", "192.168.0.128"]`
  - `p sock.addr(:numeric)   #=> ["AF_INET", 49429, "192.168.0.128", "192.168.0.128"]`
  - }

```
 */
 static VALUE
 -ip_addr(VALUE sock)
 +ip_addr(int argc, VALUE *argv, VALUE sock)
 {
 rb_io_t *fptr;
 struct sockaddr_storage addr;
 socklen_t len = sizeof addr;
```

- int norevlookup;

```
 GetOpenFile(sock, fptr);
```

- if (argc < 1 || !rsock_revlookup_flag(argv[0], &norevlookup))

- norevlookup = fptr->mode & FMODE_NOREVLOOKUP;
```
 if (getsockname(fptr->fd, (struct sockaddr*)&addr, &len) < 0)
 rb_sys_fail("getsockname(2)");
```

- return rsock_ipaddr((struct sockaddr*)&addr, fptr->mode & FMODE_NOREVLOOKUP);

- return rsock_ipaddr((struct sockaddr*)&addr, norevlookup);
```
 }
```

/*

- call-seq:
  - ipsocket.peeraddr => [address_family, port, hostname, numeric_address]
  - ipsocket.peeraddr([reverse_lookup]) => [address_family, port, hostname, numeric_address]
  - Returns the remote address as an array which contains
    @@ -168,21 +201,35 @@ ip_addr(VALUE sock)
  - It is defined for connection oriented socket such as TCPSocket.
  - If +reverse_lookup+ is +true+ or +:hostname+,
  - hostname is obtained from numeric_address using reverse lookup.
  - Or if it is +false+, or +:numeric+,
  - hostname is same as numeric_address.
  - Or if it is +nil+ or ommitted, obeys to +ipsocket.do_not_reverse_lookup+.
  - See +Socket.getaddrinfo+ also.
  - TCPSocket.open("www.ruby-lang.org", 80) {|sock|
  - `p sock.peeraddr #=> ["AF_INET", 80, "carbon.ruby-lang.org", "221.186.184.68"]`
  - `p sock.peeraddr(true)  #=> ["AF_INET", 80, "221.186.184.68", "221.186.184.68"]`
  - `p sock.peeraddr(false) #=> ["AF_INET", 80, "221.186.184.68", "221.186.184.68"]`
  - `p sock.peeraddr(:hostname) #=> ["AF_INET", 80, "carbon.ruby-lang.org", "221.186.184.68"]`
  - `p sock.peeraddr(:numeric)  #=> ["AF_INET", 80, "221.186.184.68", "221.186.184.68"]`
  - }

```
 */
 static VALUE
 -ip_peeraddr(VALUE sock)
 +ip_peeraddr(int argc, VALUE *argv, VALUE sock)
 {
 rb_io_t *fptr;
 struct sockaddr_storage addr;
 socklen_t len = sizeof addr;
```

- int norevlookup;

  GetOpenFile(sock, fptr);

- if (argc < 1 || !rsock_revlookup_flag(argv[0], &norevlookup))

- norevlookup = fptr->mode & FMODE_NOREVLOOKUP;
  if (getpeername(fptr->fd, (struct sockaddr*)&addr, &len) < 0)
  rb_sys_fail("getpeername(2)");

- return rsock_ipaddr((struct sockaddr*)&addr, fptr->mode & FMODE_NOREVLOOKUP);

- return rsock_ipaddr((struct sockaddr*)&addr, norevlookup);
  }

@@ -244,9 +291,11 @@ Init_ipsocket(void)
{
rb_cIPSocket = rb_define_class("IPSocket", rb_cBasicSocket);

- rb_define_method(rb_cIPSocket, "addr", ip_addr, 0);
- rb_define_method(rb_cIPSocket, "peeraddr", ip_peeraddr, 0);

- rb_define_method(rb_cIPSocket, "addr", ip_addr, -1);

- rb_define_method(rb_cIPSocket, "peeraddr", ip_peeraddr, -1);
  rb_define_method(rb_cIPSocket, "recvfrom", ip_recvfrom, -1);
  rb_define_singleton_method(rb_cIPSocket, "getaddress", ip_s_getaddress, 1);
  rb_undef_method(rb_cIPSocket, "getpeereid");

- id_numeric = rb_intern_const("numeric");

- id_hostname = rb_intern_const("hostname");
  }
  Index: ext/socket/socket.c
  ===
  --- ext/socket/socket.c (revision 26546)
  +++ ext/socket/socket.c (working copy)
  @@ -871,5 +871,5 @@ sock_gethostname(VALUE obj)

static VALUE
-make_addrinfo(struct addrinfo *res0)
+make_addrinfo(struct addrinfo *res0, int norevlookup)
{
VALUE base, ary;
@@ -881,5 +881,5 @@ make_addrinfo(struct addrinfo *res0)
base = rb_ary_new();
for (res = res0; res; res = res->ai_next) {

- ary = rsock_ipaddr(res->ai_addr, rsock_do_not_reverse_lookup);

- ary = rsock_ipaddr(res->ai_addr, norevlookup);
  if (res->ai_canonname) {
  RARRAY_PTR(ary)[2] = rb_str_new2(res->ai_canonname);
  @@ -956,5 +956,5 @@ sock_s_gethostbyaddr(int argc, VALUE *ar
  t = rsock_family_arg(family);
  }
  -#ifdef INET6
  +#ifdef AF_INET6
  else if (RSTRING_LEN(addr) == 16) {
  t = AF_INET6;
  @@ -1070,5 +1070,5 @@ sock_s_getservbyport(int argc, VALUE *ar
  /

- call-seq:

  - Socket.getaddrinfo(nodename, servname[, family[, socktype[, protocol[, flags]]]]) => array

  - Socket.getaddrinfo(nodename, servname[, family[, socktype[, protocol[, flags[, reverse_lookup]]]]]) => array

  - Obtains address information for *nodename*:*servname*.
    @@ -1091,12 +1091,20 @@ sock_s_getservbyport(int argc, VALUE *ar

- ["AF_INET", 0, "localhost", "127.0.0.1", 2, 3, 0]] # PF_INET/SOCK_RAW/IPPROTO_IP
  - *reverse_lookup* directs the form of the third element, and has to
  - be one of below.
  - If it is ommitted, the default value is +nil+.
  - +true+, +:hostname+: hostname is obtained from numeric address using reverse lookup, which may take a time.
  - +false+, +:numeric+: hostname is same as numeric address.
  - +nil+: obey to the current +do_not_reverse_lookup+ flag.
    */
    static VALUE
    sock_s_getaddrinfo(int argc, VALUE *argv)
    {

- VALUE host, port, family, socktype, protocol, flags, ret;
- VALUE host, port, family, socktype, protocol, flags, ret, revlookup;
  struct addrinfo hints, *res;
- int norevlookup;
- rb_scan_args(argc, argv, "24", &host, &port, &family, &socktype, &protocol, &flags);
- rb_scan_args(argc, argv, "25", &host, &port, &family, &socktype, &protocol, &flags, &revlookup);

  MEMZERO(&hints, struct addrinfo, 1);
  @@ -1112,7 +1120,10 @@ sock_s_getaddrinfo(int argc, VALUE *argv
  hints.ai_flags = NUM2INT(flags);
  }
- if (NIL_P(revlookup) || !rsock_revlookup_flag(revlookup, &norevlookup)) {
- norevlookup = rsock_do_not_reverse_lookup;
- }
  res = rsock_getaddrinfo(host, port, &hints, 0);
- ret = make_addrinfo(res);
- ret = make_addrinfo(res, norevlookup);
  freeaddrinfo(res);
  return ret;

--
Nobu Nakada

=end

#3 - 02/03/2010 11:46 PM - matz (Yukihiro Matsumoto)

=begin
Hi,

In message "Re: [ruby-core:28027] Re: [Feature:trunk] optional reverse_lookup argument for IPSocket#{addr,peeraddr} and Socket.getaddrinfo"
on Wed, 3 Feb 2010 18:04:13 +0900, Nobuyoshi Nakada nobu@ruby-lang.org writes:

|> +1, but the document should mention the default value more clearly.
|> Currently, the behavior is not clear when reverse_lookup argument is
|> omitted.
|
|Any other suggestions?

Seems fine for me. The English speaker would improve the expression.
Contribution welcome.

```
matz.
```

=end

#4 - 02/06/2010 11:37 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

- % Done changed from 0 to 100

=begin
This issue was solved with changeset r26590.
Nobuyoshi, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

=end