NAME

IO::Socket::INET6 − Object interface for AF_INET/AF_INET6 domain sockets

SYNOPSIS

```
use IO::Socket::INET6;
```

DESCRIPTION

`IO::Socket::INET6` provides an object interface to creating and using sockets in either AF_INET or AF_INET6 domains. It is built upon the IO::Socket interface and inherits all the methods defined by IO::Socket.

CONSTRUCTOR

new ([ARGS])

Creates an `IO::Socket::INET6` object, which is a reference to a newly created symbol (see the `Symbol` package). `new` optionally takes arguments, these arguments are in key-value pairs.

In addition to the key-value pairs accepted by IO::Socket, `IO::Socket::INET6` provides.

```
Domain      Address family              AF_INET | AF_INET6 | AF_UNSPEC (defa
PeerAddr    Remote host address         <hostname>[:<port>]
PeerHost    Synonym for PeerAddr
PeerPort    Remote port or service      <service>[(<no>)] | <no>
PeerFlow    Remote flow information
PeerScope   Remote address scope
LocalAddr   Local host bind address     hostname[:port]
LocalHost   Synonym for LocalAddr
LocalPort   Local host bind port        <service>[(<no>)] | <no>
LocalFlow   Local host flow information
LocalScope  Local host address scope
Proto       Protocol name (or number)   "tcp" | "udp" | ...
Type        Socket type                 SOCK_STREAM | SOCK_DGRAM | ...
Listen      Queue size for listen
ReuseAddr   Set SO_REUSEADDR before binding
Reuse       Set SO_REUSEADDR before binding (deprecated, prefer ReuseAddr)
ReusePort   Set SO_REUSEPORT before binding
Broadcast   Set SO_BROADCAST before binding
Timeout     Timeout value for various operations
MultiHomed  Try all addresses for multi−homed hosts
Blocking    Determine if connection will be blocking mode
```

If `Listen` is defined then a listen socket is created, else if the socket type, which is derived from the protocol, is SOCK_STREAM then *connect()* is called.

Although it is not illegal, the use of `MultiHomed` on a socket which is in non-blocking mode is of little use. This is because the first connect will never fail with a timeout as the connect call will not block.

The `PeerAddr` can be a hostname, the IPv6−address on the "2001:800:40:2a05::10" form , or the IPv4−address on the "213.34.234.245" form. The `PeerPort` can be a number or a symbolic service name. The service name might be followed by a number in parenthesis which is used if the service is not known by the system. The `PeerPort` specification can also be embedded in the `PeerAddr` by preceding it with a ":", and closing the IPv6 address on brackets "[]" if necessary: "124.678.12.34:23","[2a05:345f::10]:23","any.server.com:23".

If `Domain` is not given, AF_UNSPEC is assumed, that is, both AF_INET and AF_INET6 will be both considered when resolving DNS names. AF_INET6 has priority. If you guess you are in trouble not reaching the peer,(the service is not available via AF_INET6 but AF_INET) you can either try Multihomed (try any address/family until reach) or concrete your address `family` (AF_INET, AF_INET6).

If `Proto` is not given and you specify a symbolic `PeerPort` port, then the constructor will try to derive `Proto` from the service name. As a last resort `Proto` "tcp" is assumed. The `Type` parameter will be deduced from `Proto` if not specified.

If the constructor is only passed a single argument, it is assumed to be a `PeerAddr` specification.

If `Blocking` is set to 0, the connection will be in nonblocking mode. If not specified it defaults to 1 (blocking mode).

Examples:

```
   $sock = IO::Socket::INET6->new(PeerAddr => 'www.perl.org',
                                  PeerPort => 'http(80)',
                                  Proto    => 'tcp');
```

Suppose either you have no IPv6 connectivity or www.perl.org has no http service on IPv6. Then,

(Trying all address/families until reach)

```
   $sock = IO::Socket::INET6->new(PeerAddr => 'www.perl.org',
                                  PeerPort => 'http(80)',
                                  Multihomed => 1 ,
                                  Proto    => 'tcp');
```

(Concrete to IPv4 protocol)

```
   $sock = IO::Socket::INET6->new(PeerAddr => 'www.perl.org',
                                  PeerPort => 'http(80)',
                                  Domain => AF_INET ,
                                  Proto    => 'tcp');


   $sock = IO::Socket::INET6->new(PeerAddr => 'localhost:smtp(25)');

   $sock = IO::Socket::INET6->new(Listen    => 5,
                                  LocalAddr => 'localhost',
                                  LocalPort => 9000,
                                  Proto     => 'tcp');

   $sock = IO::Socket::INET6->new('[::1]:25');

   $sock = IO::Socket::INET6->new(PeerPort  => 9999,
                                  PeerAddr  => Socket6::inet_ntop(AF_INET6,in6addr
                                  Proto     => udp,
                                  LocalAddr => 'localhost',
                                  Broadcast => 1 )
                              or die "Can't bind : $@\n";
```

```
 NOTE NOTE NOTE NOTE NOTE NOTE NOTE NOTE NOTE NOTE NOTE NOTE
```

As of VERSION 1.18 all IO::Socket objects have autoflush turned on by default. This was not the case with earlier releases.

```
 NOTE NOTE NOTE NOTE NOTE NOTE NOTE NOTE NOTE NOTE NOTE NOTE
```

METHODS

accept ()

See IO::Socket::INET.

bind ()
: See IO::Socket::INET.

configure ()
: This function exists in this module, but I (= Shlomi Fish) don't know what it does, or understand it. It's also not tested anywhere. I'll be happy to be enlightened.

connect ()
: See IO::Socket::INET.

sockaddr ()
: Return the address part of the sockaddr structure for the socket

sockdomain()
: Returns the domain of the socket − AF_INET or AF_INET6 or whatever.

sockport ()
: Return the port number that the socket is using on the local host

sockhost ()
: Return the address part of the sockaddr structure for the socket in a text form ("2001:800:40:2a05::10" or "245.245.13.27")

sockflow ()
: Return the flow information part of the sockaddr structure for the socket

sockscope ()
: Return the scope identification part of the sockaddr structure for the socket

peeraddr ()
: Return the address part of the sockaddr structure for the socket on the peer host

peerport ()
: Return the port number for the socket on the peer host.

peerhost ()
: Return the address part of the sockaddr structure for the socket on the peer host in a text form ("2001:800:40:2a05::10" or "245.245.13.27")

peerflow ()
: Return the flow information part of the sockaddr structure for the socket on the peer host

peerscope ()
: Return the scope identification part of the sockaddr structure for the socket on the peer host

REPOSITORY

The Subversion repository for this module carrying complete version history and other information is:

<http://svn.berlios.de/svnroot/repos/web−cpan/IO−Socket−INET6/>

SEE ALSO

Socket,Socket6, IO::Socket

AUTHOR

This program is based on IO::Socket::INET by Graham Barr <gbarr@pobox.com> and currently maintained by the Perl Porters.

Modified by Rafael Martinez Torres <rafael.martinez@novagnet.com> and Euro6IX project.

Modified further by Shlomi Fish <shlomif@iglu.org.il>, while disclaiming all copyrights.

COPYRIGHT