Red Hat Enterprise Linux Release 9.2 Manual Pages on 'ipv6.7' command

$ man ipv6.7

IPV6(7) Linux Programmer's Manual IPV6(7)

NAME

ipv6 - Linux IPv6 protocol implementation

SYNOPSIS

```
#include <sys/socket.h>

#include <netinet/in.h>

tcp6_socket = socket(AF_INET6, SOCK_STREAM, 0);

raw6_socket = socket(AF_INET6, SOCK_RAW, protocol);

udp6_socket = socket(AF_INET6, SOCK_DGRAM, protocol);
```

DESCRIPTION

Linux 2.2 optionally implements the Internet Protocol, version 6. This man page contains a description of the IPv6 basic API as implemented by the Linux kernel and glibc 2.1. The interface is based on the BSD sockets interface; see socket(7).

The IPv6 API aims to be mostly compatible with the IPv4 API (see ip(7)). Only differences are described in this man page.

To bind an AF_INET6 socket to any process, the local address should be copied from the in6addr_any variable which has in6_addr type. In static initializations, IN6ADDR_ANY_INIT may also be used, which ex?pands to a constant expression. Both of them are in network byte or?der.

The IPv6 loopback address (::1) is available in the global in6addr_loopback variable. For initializations, IN6ADDR_LOOPBACK_INIT should be used.

IPv4 connections can be handled with the v6 API by using the v4-mapped-on-v6 address type; thus a program needs to support only this API type
to support both protocols. This is handled transparently by the ad?
dress handling functions in the C library.

IPv4 and IPv6 share the local port space. When you get an IPv4 connec?
tion or packet to an IPv6 socket, its source address will be mapped to
v6 and it will be mapped to v6.

Address format

```
    struct sockaddr_in6 {
        sa_family_t     sin6_family;   /* AF_INET6 */
        in_port_t       sin6_port;     /* port number */
        uint32_t        sin6_flowinfo; /* IPv6 flow information */
        struct in6_addr sin6_addr;     /* IPv6 address */
        uint32_t        sin6_scope_id; /* Scope ID (new in 2.4) */
    };
    struct in6_addr {
        unsigned char   s6_addr[16];   /* IPv6 address */
    };
```

sin6_family is always set to AF_INET6; sin6_port is the protocol port
(see sin_port in ip(7)); sin6_flowinfo is the IPv6 flow identifier;
sin6_addr is the 128-bit IPv6 address. sin6_scope_id is an ID depend?
ing on the scope of the address. It is new in Linux 2.4. Linux sup?
ports it only for link-local addresses, in that case sin6_scope_id con?
tains the interface index (see netdevice(7))

IPv6 supports several address types: unicast to address a single host,
multicast to address a group of hosts, anycast to address the nearest
member of a group of hosts (not implemented in Linux), IPv4-on-IPv6 to
address an IPv4 host, and other reserved address types.

The address notation for IPv6 is a group of 8 4-digit hexadecimal num?
bers, separated with a ':'. "::" stands for a string of 0 bits. Spe?
cial addresses are ::1 for loopback and ::FFFF:<IPv4 address> for
IPv4-mapped-on-IPv6.

The port space of IPv6 is shared with IPv4.

Socket options

IPv6 supports some protocol-specific socket options that can be set with setsockopt(2) and read with getsockopt(2). The socket option level for IPv6 is IPPROTO_IPV6. A boolean integer flag is zero when it is false, otherwise true.

IPV6_ADDRFORM

Turn an AF_INET6 socket into a socket of a different address family. Only AF_INET is currently supported for that. It is allowed only for IPv6 sockets that are connected and bound to a v4-mapped-on-v6 address. The argument is a pointer to an inte? ger containing AF_INET. This is useful to pass v4-mapped sock? ets as file descriptors to programs that don't know how to deal with the IPv6 API.

IPV6_ADD_MEMBERSHIP, IPV6_DROP_MEMBERSHIP

Control membership in multicast groups. Argument is a pointer to a struct ipv6_mreq.

IPV6_MTU

getsockopt(): Retrieve the current known path MTU of the current socket. Valid only when the socket has been connected. Returns an integer.

setsockopt(): Set the MTU to be used for the socket. The MTU is limited by the device MTU or the path MTU when path MTU discov? ery is enabled. Argument is a pointer to integer.

IPV6_MTU_DISCOVER

Control path-MTU discovery on the socket. See IP_MTU_DISCOVER in ip(7) for details.

IPV6_MULTICAST_HOPS

Set the multicast hop limit for the socket. Argument is a pointer to an integer. -1 in the value means use the route de? fault, otherwise it should be between 0 and 255.

IPV6_MULTICAST_IF

Set the device for outgoing multicast packets on the socket.

This is allowed only for SOCK_DGRAM and SOCK_RAW socket. The argument is a pointer to an interface index (see netdevice(7)) in an integer.

IPV6_MULTICAST_LOOP

Control whether the socket sees multicast packets that it has send itself. Argument is a pointer to boolean.

IPV6_RECVPKTINFO (since Linux 2.6.14)

Set delivery of the IPV6_PKTINFO control message on incoming datagrams. Such control messages contain a struct in6_pktinfo, as per RFC 3542. Allowed only for SOCK_DGRAM or SOCK_RAW sock? ets. Argument is a pointer to a boolean value in an integer.

IPV6_RTHDR, IPV6_AUTHHDR, IPV6_DSTOPTS, IPV6_HOPOPTS, IPV6_FLOWINFO,

IPV6_HOPLIMIT

Set delivery of control messages for incoming datagrams contain? ing extension headers from the received packet. IPV6_RTHDR de? livers the routing header, IPV6_AUTHHDR delivers the authentica? tion header, IPV6_DSTOPTS delivers the destination options, IPV6_HOPOPTS delivers the hop options, IPV6_FLOWINFO delivers an integer containing the flow ID, IPV6_HOPLIMIT delivers an inte? ger containing the hop count of the packet. The control mes? sages have the same type as the socket option. All these header options can also be set for outgoing packets by putting the ap? propriate control message into the control buffer of sendmsg(2). Allowed only for SOCK_DGRAM or SOCK_RAW sockets. Argument is a pointer to a boolean value.

IPV6_RECVERR

Control receiving of asynchronous error options. See IP_RECVERR in ip(7) for details. Argument is a pointer to boolean.

IPV6_ROUTER_ALERT

Pass forwarded packets containing a router alert hop-by-hop op? tion to this socket. Allowed only for SOCK_RAW sockets. The tapped packets are not forwarded by the kernel, it is the user's responsibility to send them out again. Argument is a pointer to an integer. A positive integer indicates a router alert option value to intercept. Packets carrying a router alert option with a value field containing this integer will be delivered to the socket. A negative integer disables delivery of packets with router alert options to this socket.

IPV6_UNICAST_HOPS

Set the unicast hop limit for the socket. Argument is a pointer to an integer. -1 in the value means use the route default, otherwise it should be between 0 and 255.

IPV6_V6ONLY (since Linux 2.4.21 and 2.6)

If this flag is set to true (nonzero), then the socket is re?stricted to sending and receiving IPv6 packets only. In this case, an IPv4 and an IPv6 application can bind to a single port at the same time.

If this flag is set to false (zero), then the socket can be used to send and receive packets to and from an IPv6 address or an IPv4-mapped IPv6 address.

The argument is a pointer to a boolean value in an integer.

The default value for this flag is defined by the contents of the file /proc/sys/net/ipv6/bindv6only. The default value for that file is 0 (false).

ERRORS

ENODEV The user tried to bind(2) to a link-local IPv6 address, but the sin6_scope_id in the supplied sockaddr_in6 structure is not a valid interface index.

VERSIONS

Linux 2.4 will break binary compatibility for the sockaddr_in6 for 64-bit hosts by changing the alignment of in6_addr and adding an addi?tional sin6_scope_id field. The kernel interfaces stay compatible, but a program including sockaddr_in6 or in6_addr into other structures may not be. This is not a problem for 32-bit hosts like i386.

The sin6_flowinfo field is new in Linux 2.4. It is transparently passed/read by the kernel when the passed address length contains it.

Some programs that pass a longer address buffer and then check the out?

going address length may break.

NOTES

The sockaddr_in6 structure is bigger than the generic sockaddr. Pro?

grams that assume that all address types can be stored safely in a

struct sockaddr need to be changed to use struct sockaddr_storage for

that instead.

SOL_IP, SOL_IPV6, SOL_ICMPV6, and other SOL_* socket options are non?

portable variants of IPPROTO_*. See also ip(7).

BUGS

The IPv6 extended API as in RFC 2292 is currently only partly imple?

mented; although the 2.2 kernel has near complete support for receiving

options, the macros for generating IPv6 options are missing in glibc

2.1.

IPSec support for EH and AH headers is missing.

Flow label management is not complete and not documented here.

This man page is not complete.

SEE ALSO

cmsg(3), ip(7)

RFC 2553: IPv6 BASIC API; Linux tries to be compliant to this.

RFC 2460: IPv6 specification.

COLOPHON

This page is part of release 5.10 of the Linux man-pages project. A

description of the project, information about reporting bugs, and the

latest version of this page, can be found at

https://www.kernel.org/doc/man-pages/.