NAME

ipv6, AF_INET6 − 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 expands to a constant expression. Both of them are in network byte order.

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 only needs to support this API type to support both protocols. This is handled transparently by the address handling functions in the C library.

IPv4 and IPv6 share the local port space. When you get an IPv4 connection or packet to a 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 depending on the scope of the address. It is new in Linux 2.4. Linux only supports it for link scope addresses, in that case *sin6_scope_id* contains 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 a IPv4 host, and other reserved address types.

The address notation for IPv6 is a group of 16 2-digit hexadecimal numbers, separated with a ':'. "::" stands for a string of 0 bits. Special 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 only allowed for IPv6 sockets that are connected and bound to a v4-mapped-on-v6 address. The argument is a pointer to an integer containing **AF_INET**. This is useful to pass v4-mapped sockets 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* structure.

IPV6_MTU
: Set the MTU to be used for the socket. The MTU is limited by the device MTU or the path mtu when path mtu discovery 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 default, otherwise it should be between 0 and 255.

IPV6_MULTICAST_IF
: Set the device for outgoing multicast packets on the socket. This is only allowed 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_PKTINFO
: Set delivery of the **IPV6_PKTINFO** control message on incoming datagrams. Only allowed for **SOCK_DGRAM** or **SOCK_RAW** sockets. Argument is a pointer to a boolean value in an integer.

IPV6_RTHDR, IPV6_AUTHHDR, IPV6_DSTOPS, IPV6_HOPOPTS, IPV6_FLOWINFO, IPV6_HOPLIMIT
: Set delivery of control messages for incoming datagrams containing extension headers from the received packet. **IPV6_RTHDR** delivers the routing header, **IPV6_AUTHHDR** delivers the authentication 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 integer containing the hop count of the packet. The control messages have the same type as the socket option. All these header options can also be set for outgoing packets by putting the appropriate control message into the control buffer of **sendmsg**(2). Only allowed 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 option to this socket. Only allowed 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 (non-zero), then the socket is restricted 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).

VERSIONS

The older *libinet6* libc5 based IPv6 API implementation for Linux is not described here and may vary in details.

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 additional *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 outgoing address length may break.

NOTES

The *sockaddr_in6* structure is bigger than the generic *sockaddr*. Programs 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.

BUGS

The IPv6 extended API as in RFC 2292 is currently only partly implemented; 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 3.22 of the Linux *man-pages* project. A description of the project, and information about reporting bugs, can be found at http://www.kernel.org/doc/man-pages/.