RTP

Benoît Legat

13 octobre 2014

We need to choose family 10 (IPv6) and protocol 17 (UDP).

A Arguments

/usr/include/bits/socket.h

```
/* Address families.  */
#define AF_UNSPEC       PF_UNSPEC
#define AF_LOCAL        PF_LOCAL
#define AF_UNIX         PF_UNIX
#define AF_FILE         PF_FILE
#define AF_INET         PF_INET
#define AF_AX25         PF_AX25
#define AF_IPX          PF_IPX
#define AF_APPLETALK    PF_APPLETALK
#define AF_NETROM       PF_NETROM
#define AF_BRIDGE       PF_BRIDGE
#define AF_ATMPVC       PF_ATMPVC
#define AF_X25          PF_X25
#define AF_INET6        PF_INET6
#define AF_ROSE         PF_ROSE
#define AF_DECnet       PF_DECnet
#define AF_NETBEUI      PF_NETBEUI
#define AF_SECURITY     PF_SECURITY
#define AF_KEY          PF_KEY
#define AF_NETLINK      PF_NETLINK
#define AF_ROUTE        PF_ROUTE
#define AF_PACKET       PF_PACKET
#define AF_ASH          PF_ASH
#define AF_ECONET       PF_ECONET
#define AF_ATMSVC       PF_ATMSVC
#define AF_RDS          PF_RDS
#define AF_SNA          PF_SNA
#define AF_IRDA         PF_IRDA
#define AF_PPPOX        PF_PPPOX
#define AF_WANPIPE      PF_WANPIPE
#define AF_LLC          PF_LLC
#define AF_CAN          PF_CAN
#define AF_TIPC         PF_TIPC
#define AF_BLUETOOTH    PF_BLUETOOTH
#define AF_IUCV         PF_IUCV
#define AF_RXRPC        PF_RXRPC
#define AF_ISDN         PF_ISDN
#define AF_PHONET       PF_PHONET
```

```
#define AF_IEEE802154   PF_IEEE802154
#define AF_CAIF         PF_CAIF
#define AF_ALG          PF_ALG
#define AF_NFC          PF_NFC
#define AF_VSOCK        PF_VSOCK
#define AF_MAX          PF_MAX
```

/usr/include/bits/socket_type.h Only 1, 2 and 3 are returned when we set 0 with protocols 6, 17 and 0 respectively.

```
/* Types of sockets.  */
enum __socket_type
{
  SOCK_STREAM = 1,              /* Sequenced, reliable, connection-based
                                   byte streams.  */
#define SOCK_STREAM SOCK_STREAM
  SOCK_DGRAM = 2,               /* Connectionless, unreliable datagrams
                                   of fixed maximum length.  */
#define SOCK_DGRAM SOCK_DGRAM
  SOCK_RAW = 3,                 /* Raw protocol interface.  */
#define SOCK_RAW SOCK_RAW
  SOCK_RDM = 4,                 /* Reliably-delivered messages.  */
#define SOCK_RDM SOCK_RDM
  SOCK_SEQPACKET = 5,           /* Sequenced, reliable, connection-based,
                                   datagrams of fixed maximum length.  */
#define SOCK_SEQPACKET SOCK_SEQPACKET
  SOCK_DCCP = 6,                /* Datagram Congestion Control Protocol.  */
#define SOCK_DCCP SOCK_DCCP
  SOCK_PACKET = 10,             /* Linux specific way of getting packets
                                   at the dev level.  For writing rarp and
                                   other similar things on the user level. */
#define SOCK_PACKET SOCK_PACKET

  /* Flags to be ORed into the type parameter of socket and socketpair and
     used for the flags parameter of paccept.  */

  SOCK_CLOEXEC = 02000000,      /* Atomically set close-on-exec flag for the
                                   new descriptor(s).  */
#define SOCK_CLOEXEC SOCK_CLOEXEC
  SOCK_NONBLOCK = 00004000      /* Atomically mark descriptor(s) as
                                   non-blocking.  */
#define SOCK_NONBLOCK SOCK_NONBLOCK
};
```

/usr/include/bits/in.h

```
/* Standard well-defined IP protocols.  */
enum
  {
    IPPROTO_IP = 0,         /* Dummy protocol for TCP.  */
#define IPPROTO_IP              IPPROTO_IP
    IPPROTO_ICMP = 1,       /* Internet Control Message Protocol.  */
#define IPPROTO_ICMP            IPPROTO_ICMP
    IPPROTO_IGMP = 2,       /* Internet Group Management Protocol. */
#define IPPROTO_IGMP            IPPROTO_IGMP
    IPPROTO_IPIP = 4,       /* IPIP tunnels (older KA9Q tunnels use 94).  */
```

```
#define IPPROTO_IPIP            IPPROTO_IPIP
    IPPROTO_TCP = 6,        /* Transmission Control Protocol.  */
#define IPPROTO_TCP             IPPROTO_TCP
    IPPROTO_EGP = 8,        /* Exterior Gateway Protocol.  */
#define IPPROTO_EGP             IPPROTO_EGP
    IPPROTO_PUP = 12,       /* PUP protocol.  */
#define IPPROTO_PUP             IPPROTO_PUP
    IPPROTO_UDP = 17,       /* User Datagram Protocol.  */
#define IPPROTO_UDP             IPPROTO_UDP
    IPPROTO_IDP = 22,       /* XNS IDP protocol.  */
#define IPPROTO_IDP             IPPROTO_IDP
    IPPROTO_TP = 29,        /* SO Transport Protocol Class 4.  */
#define IPPROTO_TP              IPPROTO_TP
    IPPROTO_DCCP = 33,      /* Datagram Congestion Control Protocol.  */
#define IPPROTO_DCCP            IPPROTO_DCCP
    IPPROTO_IPV6 = 41,      /* IPv6 header.  */
#define IPPROTO_IPV6            IPPROTO_IPV6
    IPPROTO_RSVP = 46,      /* Reservation Protocol.  */
#define IPPROTO_RSVP            IPPROTO_RSVP
    IPPROTO_GRE = 47,       /* General Routing Encapsulation.  */
#define IPPROTO_GRE             IPPROTO_GRE
    IPPROTO_ESP = 50,       /* encapsulating security payload.  */
#define IPPROTO_ESP             IPPROTO_ESP
    IPPROTO_AH = 51,        /* authentication header.  */
#define IPPROTO_AH              IPPROTO_AH
    IPPROTO_MTP = 92,       /* Multicast Transport Protocol.  */
#define IPPROTO_MTP             IPPROTO_MTP
    IPPROTO_BEETPH = 94,    /* IP option pseudo header for BEET.  */
#define IPPROTO_BEETPH          IPPROTO_BEETPH
    IPPROTO_ENCAP = 98,     /* Encapsulation Header.  */
#define IPPROTO_ENCAP           IPPROTO_ENCAP
    IPPROTO_PIM = 103,      /* Protocol Independent Multicast.  */
#define IPPROTO_PIM             IPPROTO_PIM
    IPPROTO_COMP = 108,     /* Compression Header Protocol.  */
#define IPPROTO_COMP            IPPROTO_COMP
    IPPROTO_SCTP = 132,     /* Stream Control Transmission Protocol.  */
#define IPPROTO_SCTP            IPPROTO_SCTP
    IPPROTO_UDPLITE = 136, /* UDP-Lite protocol.  */
#define IPPROTO_UDPLITE         IPPROTO_UDPLITE
    IPPROTO_RAW = 255,      /* Raw IP packets.  */
#define IPPROTO_RAW             IPPROTO_RAW
    IPPROTO_MAX
  };
```

Listing 1 – `/etc/protocols`

```
# Internet (IP) protocols
#
# Updated from http://www.iana.org/assignments/protocol-numbers and other
# sources.
# New protocols will be added on request if they have been officially
# assigned by IANA and are not historical.
# If you need a huge list of used numbers please install the nmap package.

ip      0       IP              # internet protocol, pseudo protocol number
hopopt  0       HOPOPT          # IPv6 Hop-by-Hop Option [RFC1883]
```

```
icmp     1       ICMP            # internet control message protocol
igmp     2       IGMP            # Internet Group Management
ggp      3       GGP             # gateway-gateway protocol
ipencap  4       IP-ENCAP        # IP encapsulated in IP (officially ``IP'')
st       5       ST              # ST datagram mode
tcp      6       TCP             # transmission control protocol
egp      8       EGP             # exterior gateway protocol
igp      9       IGP             # any private interior gateway (Cisco)
pup      12      PUP             # PARC universal packet protocol
udp      17      UDP             # user datagram protocol
hmp      20      HMP             # host monitoring protocol
xns-idp  22      XNS-IDP         # Xerox NS IDP
rdp      27      RDP             # "reliable datagram" protocol
iso-tp4  29      ISO-TP4         # ISO Transport Protocol class 4 [RFC905]
dccp     33      DCCP            # Datagram Congestion Control Prot. [RFC4340]
xtp      36      XTP             # Xpress Transfer Protocol
ddp      37      DDP             # Datagram Delivery Protocol
idpr-cmtp 38     IDPR-CMTP       # IDPR Control Message Transport
ipv6     41      IPv6            # Internet Protocol, version 6
ipv6-route 43    IPv6-Route      # Routing Header for IPv6
ipv6-frag 44     IPv6-Frag       # Fragment Header for IPv6
idrp     45      IDRP            # Inter-Domain Routing Protocol
rsvp     46      RSVP            # Reservation Protocol
gre      47      GRE             # General Routing Encapsulation
esp      50      IPSEC-ESP       # Encap Security Payload [RFC2406]
ah       51      IPSEC-AH        # Authentication Header [RFC2402]
skip     57      SKIP            # SKIP
ipv6-icmp 58     IPv6-ICMP       # ICMP for IPv6
ipv6-nonxt 59    IPv6-NoNxt      # No Next Header for IPv6
ipv6-opts 60     IPv6-Opts       # Destination Options for IPv6
rspf     73      RSPF CPHB       # Radio Shortest Path First (officially CPHB)
vmtp     81      VMTP            # Versatile Message Transport
eigrp    88      EIGRP           # Enhanced Interior Routing Protocol (Cisco)
ospf     89      OSPFIGP         # Open Shortest Path First IGP
ax.25    93      AX.25           # AX.25 frames
ipip     94      IPIP            # IP-within-IP Encapsulation Protocol
etherip  97      ETHERIP         # Ethernet-within-IP Encapsulation [RFC3378]
encap    98      ENCAP           # Yet Another IP encapsulation [RFC1241]
#        99                      # any private encryption scheme
pim      103     PIM             # Protocol Independent Multicast
ipcomp   108     IPCOMP          # IP Payload Compression Protocol
vrrp     112     VRRP            # Virtual Router Redundancy Protocol [RFC5798]
l2tp     115     L2TP            # Layer Two Tunneling Protocol [RFC2661]
isis     124     ISIS            # IS-IS over IPv4
sctp     132     SCTP            # Stream Control Transmission Protocol
fc       133     FC              # Fibre Channel
mobility-header 135 Mobility-Header # Mobility Support for IPv6 [RFC3775]
udplite  136     UDPLite         # UDP-Lite [RFC3828]
mpls-in-ip 137   MPLS-in-IP      # MPLS-in-IP [RFC4023]
manet    138                     # MANET Protocols [RFC5498]
hip      139     HIP             # Host Identity Protocol
shim6    140     Shim6           # Shim6 Protocol [RFC5533]
wesp     141     WESP            # Wrapped Encapsulating Security Payload
rohc     142     ROHC            # Robust Header Compression
```

```
/* Possible values for `ai_flags' field in `addrinfo' structure.  */
# define AI_PASSIVE     0x0001  /* Socket address is intended for `bind'.  */
# define AI_CANONNAME   0x0002  /* Request for canonical name.  */
# define AI_NUMERICHOST 0x0004  /* Don't use name resolution.  */
# define AI_V4MAPPED    0x0008  /* IPv4 mapped addresses are acceptable.  */
# define AI_ALL         0x0010  /* Return IPv4 mapped and IPv6 addresses.  */
# define AI_ADDRCONFIG  0x0020  /* Use configuration of this host to choose
                                   returned address type..  */
# ifdef __USE_GNU
#  define AI_IDN        0x0040  /* IDN encode input (assuming it is encoded
                                   in the current locale's␣character␣set)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣before␣looking␣it␣up.␣*/
#␣␣define␣AI_CANONIDN␣␣␣0x0080␣␣/*␣Translate␣canonical␣name␣from␣IDN␣format.␣*/
#␣␣define␣AI_IDN_ALLOW_UNASSIGNED␣0x0100␣/*␣Don't reject unassigned Unicode
                                            code points.  */
#  define AI_IDN_USE_STD3_ASCII_RULES 0x0200 /* Validate strings according to
                                                STD3 rules.  */
# endif
# define AI_NUMERICSERV 0x0400  /* Don't use name resolution.  */

/* Error values for `getaddrinfo' function.  */
# define EAI_BADFLAGS     -1    /* Invalid value for `ai_flags' field.  */
# define EAI_NONAME       -2    /* NAME or SERVICE is unknown.  */
# define EAI_AGAIN        -3    /* Temporary failure in name resolution.  */
# define EAI_FAIL         -4    /* Non-recoverable failure in name res.  */
# define EAI_FAMILY       -6    /* `ai_family' not supported.  */
# define EAI_SOCKTYPE     -7    /* `ai_socktype' not supported.  */
# define EAI_SERVICE      -8    /* SERVICE not supported for `ai_socktype'.  */
# define EAI_MEMORY       -10   /* Memory allocation failure.  */
# define EAI_SYSTEM       -11   /* System error returned in `errno'.  */
# define EAI_OVERFLOW     -12   /* Argument buffer overflow.  */
# ifdef __USE_GNU
#  define EAI_NODATA      -5    /* No address associated with NAME.  */
#  define EAI_ADDRFAMILY  -9    /* Address family for NAME not supported.  */
#  define EAI_INPROGRESS  -100  /* Processing request in progress.  */
#  define EAI_CANCELED    -101  /* Request canceled.  */
#  define EAI_NOTCANCELED -102  /* Request not canceled.  */
#  define EAI_ALLDONE     -103  /* All requests done.  */
#  define EAI_INTR        -104  /* Interrupted by a signal.  */
#  define EAI_IDN_ENCODE  -105  /* IDN encoding failed.  */
# endif

# ifdef __USE_MISC
#  define NI_MAXHOST      1025
#  define NI_MAXSERV      32
# endif

# define NI_NUMERICHOST 1       /* Don't try to look up hostname.  */
# define NI_NUMERICSERV 2       /* Don't convert port number to name.  */
# define NI_NOFQDN      4       /* Only return nodename portion.  */
# define NI_NAMEREQD    8       /* Don't return numeric addresses.  */
# define NI_DGRAM       16      /* Look up UDP service rather than TCP.  */
# ifdef __USE_GNU
#  define NI_IDN        32      /* Convert name from IDN format.  */
#  define NI_IDN_ALLOW_UNASSIGNED 64 /* Don't reject unassigned Unicode
                                        code points.  */
```

```
57  #  define NI_IDN_USE_STD3_ASCII_RULES 128 /* Validate strings according to
58                                              STD3 rules.  */
59  # endif
60
61  /* Translate name of a service location and/or a service name to set of
62     socket addresses.
63
64     This function is a possible cancellation point and therefore not
65     marked with __THROW.  */
66  extern int getaddrinfo (const char *__restrict __name,
67                          const char *__restrict __service,
68                          const struct addrinfo *__restrict __req,
69                          struct addrinfo **__restrict __pai);
70
71  /* Free `addrinfo' structure AI including associated storage.  */
72  extern void freeaddrinfo (struct addrinfo *__ai) __THROW;
73
74  /* Convert error return from getaddrinfo() to a string.  */
75  extern const char *gai_strerror (int __ecode) __THROW;
```