NAME

ip, eth, psip, udp, tcp − Internet Protocol server devices and definitions

DESCRIPTION

The **ip***, **eth***, **psip***, **tcp***, and **udp*** devices give access to the Internet Protocol (IP) services in Minix. There are 16 such devices for four different networks, and four different ways of accessing them. The **eth*** and **psip*** devices give direct access to the network packets at the lowest level. The **ip***, **tcp***, and **udp*** devices give access to IP, TCP, or UDP services. See **netdefault**(8) for a description of the default network devices and environment variables. The proper TCP device to use for instance is found by:

```
if ((tcp_device= getenv("TCP_DEVICE")) == NULL)
        tcp_device= "/dev/tcp";
```

Access to the IP services is provided using filedescriptors to open IP devices. These open IP channels can be configured with **ioctl**(2) calls, and data can be transferred by calls to **read**(2), and **write**(2).

Types (general)

<sys/types.h>
: defines **u8_t**, **u16_t**, **u32_t** and **i32_t** (and **U8_t**, **U16_t**, **U32_t** and **I32_t** for use in prototypes).

Types (eth)

<net/gen/ether.h>
: defines struct ether_addr (**ether_addr_t**) and **ether_type_t** and **Ether_type_t** for use in prototypes.

<net/gen/eth_io.h>
: defines struct nwio_ethopt (**nwio_ethopt_t**) and struct nwio_ethstat (**nwio_ethstat_t**)

<net/gen/eth_hdr.h>
: defines struct eth_hdr (**eth_hdr_t**)

Types (ip)

<net/gen/in.h>
: defines **ipaddr_t**, **ipproto_t** and struct ip_hdropt (**ip_hdropt_t**).

<net/gen/ip_io.h>
: defines struct nwio_ipconf (**nwio_ipconf_t**) and struct nwio_ipopt (**nwio_ipopt_t**)

<net/gen/ip_hdr.h>
: defines struct ip_hdr (**ip_hdr_t**)

<net/gen/route.h>
: defines struct nwio_route (**nwio_route_t**)

Types (tcp)

<net/gen/tcp.h>
: defines **tcpport_t** and **Tcpport_t** for use in prototypes.

<net/gen/tcp_io.h>
: defines struct nwio_tcpconf (**nwio_tcpconf_t**), struct nwio_tcpcl (**nwio_tcpcl_t**), struct nwio_tcpatt (**nwio_tcpatt_t**) and struct nwio_tcpopt (**nwio_tcpopt_t**).

<net/gen/tcp_hdr.h>
: defines struct tcp_hdr (**tcp_hdr_t**) and struct tcp_hdropt (**tcp_hdropt_t**).

Types (udp)

<net/gen/udp.h>
: defines **udpport_t** and **Udpport_t** for use in prototypes.

<net/gen/udp_io.h>
: defines struct nwio_udpopt (**nwio_udpopt_t**).

<net/gen/udp_hdr.h>
defines struct udp_hdr (**udp_hdr_t**) and struct udp_io_hdr (**udp_io_hdr_t**).

Byte Order Conversion

All 16-bit and 32-bit quantities in IP headers must be in network byte order. The macros described in **hton**(3) can be used to convert these values to and from the byte order used by the host machine.

The Internet Checksum

The **OneC_sum** function (see **oneC_sum**(3)) is used to calculate the one's complement checksum needed for IP network packets.

General Functions

fd **= open(***tcpip_device***, O_RDWR)**

This is how one normally obtains a filedescriptor for a new TCP/IP channel. *tcpip_device* names one of the TCP/IP devices. The channel may be used both to send or to receive data.

n **= read(***fd***,** *buf***,** *size***)**

Receives one packet (low level devices) or a number of bytes (TCP stream). Returns the the number of bytes placed into *buf*, or returns -1 with an error code placed into **errno**.

n **= write(***fd***,** *buf***,** *size***)**

Sends one packet (low level devices) or a number of bytes (TCP stream). Returns *size* or -1 with the error code placed into **errno**. The TCP/IP **read** and **write** functions behave like reads and writes on pipes when it comes to signals.

ETH Functions

ioctl(*fd***, NWIOGETHSTAT, &struct nwio_ethstat)**

The **NWIOGETHSTAT** ioctl returns the Ethernet address and some statistics of the Ethernet server of the channel *fd*. The result is returned in the nwio_ethstat structure. The **struct nwio_ethstat** is defined in <net/gen/eth_io.h>:

```
typedef struct nwio_ethstat
{
        ether_addr_t nwes_addr;
        eth_stat_t nwes_stat;
} nwio_ethstat_t;

typedef struct eth_stat
{
    unsigned long ets_recvErr,  /* # receive errors */
      ets_sendErr,             /* # send error */
      ets_OVW,                 /* # buffer overwrite warnings,
                                    (packets arrive faster than
                                     can be processed) */
      ets_CRCerr,              /* # crc errors of read */
      ets_frameAll,            /* # frames not aligned (# bits
                                    not a multiple of 8) */
      ets_missedP,             /* # packets missed due to too
                                    slow packet processing */
      ets_packetR,             /* # packets received */
      ets_packetT,             /* # packets transmitted */
      ets_transDef,            /* # transmission deferred (there
                                    was a transmission of an
                                    other station in progress */
      ets_collision,           /* # collisions */
      ets_transAb,             /* # transmissions aborted due
                                    to excessive collisions */
      ets_carrSense,           /* # carrier sense lost */
      ets_fifoUnder,          /* # fifo underruns (processor
```

```
                                        is too busy) */
      ets_fifoOver,           /* # fifo overruns (processor is
                                        too busy) */
      ets_CDheartbeat,         /* # times unable to transmit
                                        collision signal */
      ets_OWC;                 /* # times out of window
                                        collision */
} eth_stat_t;
```

ioctl(*fd***, NWIOSETHOPT, &struct nwio_ethopt)**

Before an Ethernet channel can be used to send or receive Ethernet packets, it has to be configured using the **NWIOSETHOPT** ioctl. The structure **nwio_ethopt** is defined in <net/gen/eth_io.h>:

```
typedef struct nwio_ethopt
{
      u32_t nweo_flags;
      ether_addr_t nweo_multi, nweo_rem;
      ether_type_t nweo_type;
} nwio_ethopt_t;

#define NWEO_NOFLAGS    0x0000L
#define NWEO_ACC_MASK   0x0003L
#       define NWEO_EXCL        0x00000001L
#       define NWEO_SHARED      0x00000002L
#       define NWEO_COPY        0x00000003L
#define NWEO_LOC_MASK   0x0010L
#       define NWEO_EN_LOC      0x00000010L
#       define NWEO_DI_LOC      0x00100000L
#define NWEO_BROAD_MASK 0x0020L
#       define NWEO_EN_BROAD    0x00000020L
#       define NWEO_DI_BROAD    0x00200000L
#define NWEO_MULTI_MASK 0x0040L
#       define NWEO_EN_MULTI    0x00000040L
#       define NWEO_DI_MULTI    0x00400000L
#define NWEO_PROMISC_MASK 0x0080L
#       define NWEO_EN_PROMISC  0x00000080L
#       define NWEO_DI_PROMISC  0x00800000L
#define NWEO_REM_MASK   0x0100L
#       define NWEO_REMSPEC     0x00000100L
#       define NWEO_REMANY      0x01000000L
#define NWEO_TYPE_MASK  0x0200L
#       define NWEO_TYPESPEC    0x00000200L
#       define NWEO_TYPEANY     0x02000000L
#define NWEO_RW_MASK    0x1000L
#       define NWEO_RWDATONLY   0x00001000L
#       define NWEO_RWDATALL    0x10000000L
```

The configuration is divided in a number of section (covered by the xx_MASK macros). Options can be set in the **nweo_flags** field. The first section (**NWEO_ACC_MASK**) controls the access to a certain Ethernet packet type. If **NWEO_EXCL** is selected then this is the only channel that can send or receive Ethernet packets of the selected type. If **NWEO_SHARED** is selected then multiple channels (which all have to select **NWEO_SHARED**) can use the same Ethernet type, they all can send packets but incoming packets will be delivered to at most one of them. If **NWEO_COPY** is selected then multiple channels have access to the same Ethernet type and all receive a copy of an incoming packet.

The **NWEO_LOC_MASK** flags control the delivery of packets with a destination address equal to the Ethernet address of the machine. If **NWEO_EN_LOC** is selected then these packets will be delivered and with **NWEO_DI_LOC** they will be discarded.

NWEO_BROAD_MASK, **NWEO_MULTI_MASK**, and **NWEO_PROMISC_MASK** do the same to broadcast packets, multicast packets and promiscuous mode packets as **NWEO_LOC_MASK** does for local packets. Except that the precise multicast address is taken from the **nweo_multi** field.

The **NWEO_REM_MASK** flags control whether communication is restricted to single destination or not. **NWEO_REMSPEC** restricts sending and receiving of packets to the single remote computer specified in the **nweo_rem** field. **NWEO_REMANY** allows sending to and receiving from any remote computer.

NWEO_TYPESPEC restricts sending and receiving of packets to the type specified in **nweo_type**. The type has to be in network byte order (using **hton**(3)). **NWEO_TYPEANY** allows any type.

If the Ethernet header is completely specified by the **nweo_flags** i.e., all of **NWEO_EN_LOC**, **NWEO_DI_BROAD**, **NWEO_DI_MULTI**, **NWEO_DI_PROMISC**, **NWEO_REMSPEC** and **NWEO_TYPESPEC** are specified, then **NWEO_RWDATONLY** can be used to send and receive only the data part of an Ethernet packet. If **NWEO_RWDATALL** is specified then both Ethernet header and data are used.

PSIP Functions

[[[No description available yet.]]]

IP Functions

ioctl(*fd*, NWIOGIPCONF, &struct nwio_ipconf)

The **NWIOGIPCONF** ioctl reports the Internet Address and the netmask. For the *nwio_ipconf* structure see the **NWIOSIPCONF** ioctl below.

ioctl(*fd*, NWIOGIPOROUTE, &struct nwio_route)

The **NWIOGIPOROUTE** ioctl can be used to query an IP server about its routing table. [[[NWIODIPOROUTE, NWIOGIPIROUTE, NWIODIPIROUTE?]]] The structure **nwio_route** is defined in <net/gen/route.h>:

```
typedef struct nwio_route
{
        u32_t nwr_ent_no;
        u32_t nwr_ent_count;
        ipaddr_t nwr_dest;
        ipaddr_t nwr_netmask;
        ipaddr_t nwr_gateway;
        u32_t nwr_dist;
        u32_t nwr_flags;
        u32_t nwr_pref;
} nwio_route_t;

#define NWRF_EMPTY      0
#define NWRF_INUSE      1
#define NWRF_FIXED      2
```

The requested entry is taken from **nwr_ent_no**. Entries are counted from 0, so the value 0 can be used for an initial query. The size of the routing table is returned in **nwr_ent_count**. The **nwr_flags** indicates if the entry is in use (**NWRF_INUSE**) and if the entry was inserted manually (using **NWIOSIPOROUTE**) or generated by the IP server itself. The route is described by **nwr_dest**, **nwr_netmask**, **nwr_gateway**, **nwr_dist**, and **nwr_pref**. **Nwr_dest** and **nwr_netmask** select the destination addresses. A value of 0.0.0.0 (0x0) in both **nwr_dest** and **nwr_netmask** means every host. A value of 255.255.255.255 (0xffffffff) in **nwr_netmask** means a single host. Other values of **nwr_netmask** are netmasks for the network specified by **nwr_dest**. **Nwr_gateway** is gateway that should be used. **Nwr_dist** is a minimal distance. Packets with a time to live smaller than **nwr_dist** will not reach the destination. If two routes have equal netmask and distance fields but different gateways then the gateway with highest value in **nwr_pref** is used.

ioctl(*fd*, NWIOSIPCONF, &struct nwio_ipconf)

The **NWIOSIPCONF** ioctl can be used to inform the IP server about its Internet Address and/or its netmask. Normally an IP server will discover its Internet Address using the RARP protocol. **NWIOSIPCONF** can be used in the case that the RARP failed, or the netmask has to be changed. Note that higher level protocols (TCP and UDP) assume that the Internet Address of an IP device does not change, therefore TCP and UDP stop functioning if the Internet Address is changed.

The structure **nwio_ipconf** is defined in <net/gen/ip_io.h>:

```
typedef struct nwio_ipconf
{
      u32_t   nwic_flags;
      ipaddr_t nwic_ipaddr;
      ipaddr_t nwic_netmask;
} nwio_ipconf_t;

#define NWIC_NOFLAGS            0x0
#define NWIC_FLAGS              0x3
#       define NWIC_IPADDR_SET          0x1
#       define NWIC_NETMASK_SET         0x2
```

The function of **nwio_ipconf** depends on the value of **nwic_flags**. If **NWIC_IPADDR_SET** is set then the Internet Address will be set to **nwic_ipaddr**. If **NWIC_NETMASK_SET** is set then the Internet Address will be set to **nwic_netmask**.

ioctl(*fd*, NWIOSIPOPT, &struct nwio_ipopt)

Before an IP channel can be used, it has to be configured using the **NWIOSIPOPT** ioctl. The structure **nwio_ipopt** is defined in <net/gen/ip_io.h>:

```
typedef struct nwio_ipopt
{
        u32_t nwio_flags;
        ipaddr_t nwio_rem;
        ip_hdropt_t nwio_hdropt;
        u8_t nwio_tos;
        u8_t nwio_ttl;
        u8_t nwio_df;
        ipproto_t nwio_proto;
} nwio_ipopt_t;

#define NWIO_NOFLAGS    0x0000L
#define NWIO_ACC_MASK   0x0003L
#       define NWIO_EXCL        0x00000001L
#       define NWIO_SHARED      0x00000002L
#       define NWIO_COPY        0x00000003L
#define NWIO_LOC_MASK   0x0010L
#       define NWIO_EN_LOC      0x00000010L
#       define NWIO_DI_LOC      0x00100000L
#define NWIO_BROAD_MASK 0x0020L
#       define NWIO_EN_BROAD    0x00000020L
#       define NWIO_DI_BROAD    0x00200000L
#define NWIO_REM_MASK   0x0100L
#       define NWIO_REMSPEC     0x00000100L
#       define NWIO_REMANY      0x01000000L
#define NWIO_PROTO_MASK 0x0200L
#       define NWIO_PROTOSPEC   0x00000200L
#       define NWIO_PROTOANY    0x02000000L
#define NWIO_HDR_O_MASK 0x0400L
#       define NWIO_HDR_O_SPEC  0x00000400L
```

```
#       define NWIO_HDR_O_ANY   0x04000000L
#define NWIO_RW_MASK    0x1000L
#       define NWIO_RWDATONLY   0x00001000L
#       define NWIO_RWDATALL    0x10000000L
```

The options are divided in several categories: **NWIO_ACC_MASK**, **NWIO_LOC_MASK**, **NWIO_BROAD_MASK**, **NWIO_REM_MASK**, **NWIO_PROTO_MASK**, **NWIO_HDR_O_MASK** and **NWIO_RW_MASK**. A channel is configured when one option of each category is set.

The options covered by **NWIO_ACC_MASK** control the number of channels that can use one IP protocol. If **NWIO_EXCL** is specified then only that channel can use a certain IP protocol. If **NWIO_SHARED** then multiple channels that all have to specify **NWIO_SHARED** can use the same IP protocol, but incoming packets will be delivered to a most one channel. **NWIO_SHARED** does not impose any restrictions. Every channel gets a copy of an incoming packet.

NWIO_LOC_MASK and **NWIO_BROAD_MASK** control the delivery of packets. If **NWIO_EN_LOC** is specified then packets that are explicitly send to the IP server are delivered. If **NWIO_EN_BROAD** is specified then broadcast packets are delivered. Either one or both of them can be disabled with **NWIO_DI_LOC** and **NWIO_DI_BROAD**.

NWIO_REMSPEC can be used to restrict communication to one remote host. This host is taken from the **nwio_rem** field. If any remote host is to be allowed then **NWIO_REMANY** can be used.

NWIO_PROTOSPEC restricts communication to one IP protocol, specified in **nwio_proto**. **NWIO_PROTOANY** allows any protocol to be sent or received.

NWIO_HDR_O_SPEC specifies all IP header options in advance. The values are taken from **nwio_hdropt**, **nwio_tos**, **nwio_ttl**, and **nwio_df**. **Nwio_hdropt** specifies the IP options that should be present in an outgoing packet. **Ip_hdropt_t** is defined in <net/gen/in.h>:

```
typedef struct ip_hdropt
{
        u8_t iho_opt_siz;
        u8_t iho_data[IP_MAX_HDR_SIZE-IP_MIN_HDR_SIZE];
} ip_hdropt_t;
```

The bytes of size **iho_opt_siz** in **iho_data** are appended to the IP header. **Nwio_tos** specifies the value of the ‘‘type of service’’ bits, **nwio_ttl** gives the value of the ‘‘time to live’’ field and **nwio_df** specifies whether fragmentation is disallowed or not. **NWIO_HDR_O_ANY** specifies that the header options should be specified at each write request.

NWIO_RWDATONLY specifies that the header should be omitted from a write request. This option can only be used when all header fields are specified in previous options: **NWIO_EN_LOC**, **NWIO_DI_BROAD**, **NWIO_REMSPEC**, **NWIO_PROTOSPEC** and **NWIO_HDR_O_SPEC**. A read operation will also only return the data part, so the IP options will be lost.

ioctl(*fd***, NWIOSIPOROUTE, &struct nwio_route)**

The **NWIOSIPOROUTE** ioctl adds a route to the routing table. See **NWIOGIPOROUTE** above for a description of the **nwio_route** structure. The fields **nwr_ent_no** and **nwr_ent_count** are ignored.

TCP Functions

ioctl(*fd***, NWIOTCPCONN, &struct nwio_tcpcl)**

The **NWIOTCPCONN** ioctl tries to setup a connection with a remote TCP/IP server. The channel must be fully configured (see **NWIOSTCPCONF**) and values for the local port, the remote port and the remote address have be specified using **NWTC_LP_SET** or **NWTC_LP_SEL**, **NWTC_SET_RA** and **NWTC_SET_RP**. The struct nwio_tcpcl is defined in <net/gen/tcp_io.h> as:

```
typedef struct nwio_tcpcl
{
      long nwtcl_flags;
      long nwtcl_ttl;
```

```
} nwio_tcpcl_t;
```

Set the **nwtcl_flags** field to zero before the connect or listen call. [[[Further explanation of nwio_tcpcl?]]]

ioctl(*fd*, NWIOGTCPCONF, &struct nwio_tcpconf)

This call reports the current configuration of a TCP channel. The **nwtc_flags** field shows the status of the **access**, **locport**, **remaddr** and **remport** fields. **Nwtc_locaddr** contains the Internet address of the TCP/IP server. **Remaddr** contains the Internet address of the remote TCP/IP server when set with **NWTC_SET_RA** or after a successful connect or listen (see **NWIOTCPCONN** or **NWIOTCPLISTEN**). **Nwio_locport** contains the local TCP/IP port set with **NWTC_LP_SET** or the selected port set with **NWTC_LP_SEL**. **Nwtc_remport** contains the TCP port of the remote TCP/IP server as set with **NWIO_SET_RP** or after a successful connect or listen.

A value of 0 (zero) is reported for **nwtc_remaddr**, **nwtc_locport** or **nwtc_remport** when no value is set either explicitly or implicitly.

ioctl(*fd*, NWIOTCPLISTEN, &struct nwio_tcpcl)

The **NWIOTCPLISTEN** ioctl waits until a remote TCP/IP server tries to connect to this channel. The channel has to be configured (see **NWIOSTCPCONF**). An additional restriction is that the local port must be set (with **NWTC_LP_SET**) or selected (with **NWTC_LP_SEL**). When a remote address is set only connections for that host are accepted, and when a remote port is set only connections from that port are accepted. After a successful listen **NWIOGTCPCONF** can be used to find out what the address and port of the other side are.

ioctl(*fd*, NWIOSTCPCONF, &struct nwio_tcpconf)

Before a TCP channel can be used it must configured using the **NWIOSTCPCONF** ioctl. The parameters to **NWIOSTCPCONF** are the channel file descriptor and a **struct nwio_tcpconf** as defined in <net/gen/tcp_io.h>:

```
typedef struct nwio_tcpconf
{
      u32_t nwtc_flags;
      ipaddr_t nwtc_locaddr;
      ipaddr_t nwtc_remaddr;
      tcpport_t nwtc_locport;
      tcpport_t nwtc_remport;
} nwio_tcpconf_t;

#define NWTC_NOFLAGS    0x0000L
#define NWTC_ACC_MASK   0x0003L
#       define NWTC_EXCL        0x00000001L
#       define NWTC_SHARED      0x00000002L
#       define NWTC_COPY        0x00000003L
#define NWTC_LOCPORT_MASK       0x0030L
#       define NWTC_LP_UNSET    0x00000010L
#       define NWTC_LP_SET      0x00000020L
#       define NWTC_LP_SEL      0x00000030L
#define NWTC_REMADDR_MASK       0x0100L
#       define NWTC_SET_RA      0x00000100L
#       define NWTC_UNSET_RA    0x01000000L
#define NWTC_REMPORT_MASK       0x0200L
#       define NWTC_SET_RP      0x00000200L
#       define NWTC_UNSET_RP    0x02000000L
```

A tcp channel is considered configured when one flag in each category has been selected. Thus one of **NWTC_EXCL**, **NWTC_SHARED** or **NWTC_COPY**, one of **NWTC_LP_UNSET**, **NWTC_LP_SET** or **NWTC_LP_SEL**, one of **NWTC_SET_RA** or **NWTC_UNSET_RA**, and one of **NWTC_SET_RP** or **NWTC_UNSET_RP**.

The acc flags control the access to a certain TCP port. **NWTC_EXCL** means exclusive access. An attempt to configure a channel will be denied if the same port is specified as that of a channel that requested exclusive access. **NWTC_SHARED** indicates that several channels use the same port but cooperate. If the shared mode is specified for one channel than all other channel that use the same port should also be configured with the **NWTC_SHARED** flag. **NWTC_COPY** is specified when the programmer does not care about other channels. This is the default.

The locport flags control which TCP port is used for communication. **NWTC_LP_UNSET** indicates the absence of a local port. This is the default. **NWTC_LP_SET** means that the **nwtc_locport** field contains the local port to be used by TCP. This value must be in network byte order (see **hton**(3).) **NWTC_LP_SEL** requests the TCP server to pick a port. This port will be in the range from 32768 to 65535 and will be unique.

The **remaddr** flags specify which hosts are acceptable for connections. **NWTC_SET_RA** indicates that only connection to the host specified in **nwtc_remaddr** are acceptable. **Nwtc_remaddr** should be in network byte order (see **hton**(3).) **NWTC_UNSET_RA** allows every host on the other side of a connection. This is the default.

The **remport** flags specify which remote ports are acceptable for connections. **NWTC_SET_RP** indicates that only the port specified in **nwtc_remport** is acceptable. **NWTC_UNSET_RP** allows every port on the other side of a connection. This is the default.

ioctl(*fd***, NWIOTCPSHUTDOWN)**

The **NWIOTCPSHUTDOWN** tells the TCP/IP server that no more data will be sent over the channel specified by *fd*. This command can be issued when the channel is connected to a remote TCP/IP server. The TCP/IP server will tell the remote TCP/IP server and the client of the remote TCP/IP server will receive an end-of-file indication.

ioctl(*fd***, NWIOGTCPOPT, &struct nwio_tcpopt)**
ioctl(*fd***, NWIOSTCPOPT, &struct nwio_tcpopt)**

The behaviour of a TCP channel may be changed by setting a number of options. The TCP options can be obtained with the **NWIOGTCPOPT** ioctl and set with the **NWIOSTCPOPT** ioctl. The options are passed in a **struct nwio_tcpopt** as defined in <net/gen/tcp_io.h>:

```
typedef struct nwio_tcpopt
{
      u32_t nwto_flags;
} nwio_tcpopt_t;

#define NWTO_NOFLAG     0x0000L
#define NWTO_SND_URG_MASK       0x0001L
#       define NWTO_SND_URG     0x00000001L
#       define NWTO_SND_NOTURG  0x00010000L
#define NWTO_RCV_URG_MASK       0x0002L
#       define NWTO_RCV_URG     0x00000002L
#       define NWTO_RCV_NOTURG  0x00020000L
#define NWTO_BSD_URG_MASK       0x0004L
#       define NWTO_BSD_URG     0x00000004L
#define NWTO_DEL_RST_MASK       0x0008L
#       define NWTO_DEL_RST     0x00000008L
```

The **NWTO_SND_URG** option causes bytes written to the channel to be send out as urgent data. On receiving an **EURG** error the **NWTO_RCV_URG** option must be set to switch over to reading urgent data. When all urgent data has been read an **ENOURG** error will follow, indicating that the option must be cleared with **NWTO_RCV_NOTURG**. Alas the BSD implementation of urgent data disagrees with the RFC's, so to be BSD compatible one must set the **NWTO_BSD_URG** option beforehand on a channel that is to send or receive urgent data. Given that the BSD implementation is the regarded as the TCP/IP standard one should always use the BSD style. The **NWTO_DEL_RST** option delays a failure response on a connect to the same port as the current open connection. Without this option a connect would fail if a server is not yet listening. With this option a connect will linger on until the server starts listening. This option is useful for a server that opens a connection, tells the remote end the local port number and then listens (FTP), or for a program that forks off servers for incoming connections (TELNET). A new connection may come in before a new listen can be started, so it is nice if the new connect doesn't fail. Use this option only when it is clearly needed.

UDP Functions

ioctl(*fd***, NWIOGUDPOPT, &struct nwio_udpopt)**

The **NWIOGUDPOPT** ioctl returns the current options that result from the default options and the options set with **NWIOSUDPOPT**. When **NWUO_LP_SEL** or **NWUO_LP_SET** is selected the local port is returned in **nwuo_locport**. When **NWUO_RP_SET** is selected the remote port is returned in **nwuo_remport**. The local address is always returned in **nwuo_locaddr**, and when **NWUO_RA_SET** is selected the remote address is returned in **nwuo_remaddr**.

ioctl(*fd***, NWIOSUDPOPT, &struct nwio_udpopt)**

A UDP channel must be configured using the **NWIOSUDPOPT** ioctl before any data can be read or written. **NWIOSUDPOPT** takes two parameters, a file descriptor to an open UDP device and pointer to a **nwio_udpopt** structure that describes the requested configuration. The **nwio_udpopt** structure is defined in <net/gen/udp_io.h> as:

```
typedef struct nwio_udpopt
{
      unsigned long nwuo_flags;
      udpport_t nwuo_locport;
      udpport_t nwuo_remport;
      ipaddr_t nwuo_locaddr;
      ipaddr_t nwuo_remaddr;
} nwio_udpopt_t;

#define NWUO_NOFLAGS            0x0000L
#define NWUO_ACC_MASK           0x0003L
#define         NWUO_EXCL               0x00000001L
#define         NWUO_SHARED             0x00000002L
#define         NWUO_COPY               0x00000003L
#define NWUO_LOCPORT_MASK       0x000CL
#define         NWUO_LP_SEL             0x00000004L
#define         NWUO_LP_SET             0x00000008L
#define         NWUO_LP_ANY             0x0000000CL
#define NWUO_LOCADDR_MASK       0x0010L
#define         NWUO_EN_LOC             0x00000010L
#define         NWUO_DI_LOC             0x00100000L
#define NWUO_BROAD_MASK         0x0020L
#define         NWUO_EN_BROAD           0x00000020L
#define         NWUO_DI_BROAD           0x00200000L
#define NWUO_REMPORT_MASK       0x0100L
#define         NWUO_RP_SET             0x00000100L
#define         NWUO_RP_ANY             0x01000000L
#define NWUO_REMADDR_MASK       0x0200L
#define         NWUO_RA_SET             0x00000200L
#define         NWUO_RA_ANY             0x02000000L
#define NWUO_RW_MASK            0x1000L
#define         NWUO_RWDATONLY          0x00001000L
#define         NWUO_RWDATALL           0x10000000L
#define NWUO_IPOPT_MASK         0x2000L
#define         NWUO_EN_IPOPT           0x00002000L
#define         NWUO_DI_IPOPT           0x20000000L
```

A UDP channel is considered configured when one flag in each category has been selected. Thus one of **NWUO_EXCL**, **NWUO_SHARED** or **NWUO_COPY**, one of **NWUO_LP_SEL**, **NWUO_LP_SET** or **NWUO_LP_ANY**, one of **NWUO_EN_LOC** or **NWUO_DI_LOC**, one of **NWUO_EN_BROAD**, or **NWUO_DI_BROAD**, one of **NWUO_RP_SET**, or **NWUO_RP_ANY**, one of **NWUO_RA_SET**, or **NWUO_RA_ANY**, one of **NWUO_RWDATONLY**, or **NWUO_RWDATALL**, and one of **NWUO_EN_IPOPT**, or **NWUO_DI_IPOPT**. The acc flags control the access to a certain UDP port. **NWUO_EXCL** means exclusive access: no other channel can use this port. **NWUO_SHARED** means shared access: only channels that specify shared access can use this port and all packets that are received are handed to at most one channel. **NWUO_COPY** imposes no access restriction and all channels get a copy of every received packet for that port.

The **locport** flags control the selection of the UDP port for this channel. **NWUO_LP_SEL** requests the server to pick a port. This port will be in the range from 32768 to 65535 and it will be unique. **NWUO_LP_SET** sets the local port to the value of the **nwuo_locport** field. **NWUO_LP_ANY** does not select a port. Reception of data is therefore not possible but it is possible to send data.

The **locaddr** flags control the reception of packets. **NWUO_EN_LOC** enables the reception of packets with the local IP address as destination. **NWUO_DI_LOC** disables the reception of packet for the local IP address.

The **broad** flags control the reception of broadcast packets. **NWUO_EN_BROAD** enables the reception of broadcast packets and **NWUO_DI_BROAD** disables the reception of broadcast packets.

The **remport** flags let the client to specify one specific remote UDP port or to allow any remote port. **NWUO_RP_SET** sets the remote UDP port to the value of **nwuo_remport**. Only packets with a matching remote port will be delivered and all packets will be sent to that port. **NWUO_RP_ANY** allows reception of packets form any port and when transmitting packets the remote port has to be specified.

The **remaddr** flags control the remote IP address. **NWUO_RA_SET** sets the remote IP address the value of **nwuo_remaddr**. Only packets from that address will be delivered and all packets will be sent to that address. **NWUO_RA_ANY** allows reception of packets from any host and when transmitting packets the remote host has to be specified.

The **rw** flags control the format of the data to be sent or received. With **NWUO_RWDATONLY** only the data part of a UDP packet is sent to the server and only the data part is received from the server. The **NWUO_RWDATALL** mode presents the data part of a UDP packet with a header that contains the source and destination IP address, source and destination UDP ports, the IP options, etc. The server expects such a header in front of the data to be transmitted.

The **ipopt** flags control the delivery and transmission of IP options. When **NWUO_EN_IPOPT** is set IP, options will be delivered and sent. When **NWUO_DI_IPOPT** is set IP option will be stripped from received packets and no IP options will be sent.

FILES

/dev/eth[01]	First and second raw ethernet.
/dev/psip[23]	First and second Pseudo IP network.
/dev/ip[0123]	IP devices for two ethernets and two Pseudo IP networks.
/dev/tcp[0123]	TCP devices for same four networks.
/dev/udp[0123]	UDP devices.

/dev/eth, /dev/psip, /dev/ip, /dev/tcp, /dev/udp
Devices for the default network, links to the devices above. **Eth** is only present if ethernet is the default, **psip** only for pseudo IP.

SEE ALSO

hton(3), **oneC_sum**(3), **netdefault**(8), **boot**(8).

DIAGNOSTICS

Several errors may be returned by the TCP/IP server. The error code is found in the **errno** variable if the **read**, **write**, or **ioctl** call returns -1. The TCP/IP error codes defined in <errno.h> are:

EPACKSIZE	This indicates an attempt to read or write with a buffer that is too large or too small.
EOUTOFBUFS	The TCP/IP server has insufficient memory to execute the request.
EBADIOCTL	This indicates an attempt to execute a command the particular server does not understand. For example, a **NWIOGTCPCONF** on an ETH channel.
EBADMODE	The request is refused because the channel is not fully configured, in the wrong state or the parameters are invalid.
EBADDEST	This indicates an illegal destination address for a packet.
EDSTNORCH	The destination is not reachable.
EISCONN	The channel is already connected so a second request is refused.
EADDRINUSE	This address is in use.
ECONNREFUSED	The connection is refused by the other side.
ECONNRESET	The connection is reset (non-gracefully terminated) by the other side.
ETIMEDOUT	The connection is terminated due to an expired timer.
EURG	Urgent data is present and the current receive mode does not allow urgent data to be transferred.
ENOURG	No urgent data is present and a request came for urgent data.
ENOTCONN	The request requires a connected channel and the channel is not connected.
ESHUTDOWN	The connection is shut down. That is, a **NWIOTCPSHUTDOWN** has been executed so no more data can be transmitted.
ENOCONN	The connection does not exist.
EGENERIC	A generic error code for extremely weird cases.

AUTHOR

Philip Homburg (philip@cs.vu.nl)