NAME

if_tcp, if – general properties of Internet Protocol network interfaces

DESCRIPTION

A network interface is a device for sending and receiving packets on a network. A network interface is usually a hardware device, although one may be implemented in software. Network interfaces used by the Internet Protocol (IP) must be STREAMS devices conforming to the Datalink Provider Interface (DLPI). See **dlpi**(7P).

APPLICATION PROGRAMMING INTERFACE

An interface becomes available to IP when it is opened and the IP module is pushed onto the stream with the **I_PUSH** **ioctl**(2) command (see **streamio**(7I)). This may be initiated by the kernel at boot time or by a user program some time after the system is running. Each interface must be assigned an IP address with the **SIOCSIFADDR** **ioctl( )** before it can be used. On interfaces where the network-to-link layer address mapping is static, only the network number is taken from the **ioctl( )** request; the remainder is found in a hardware specific manner. On interfaces which provide dynamic network-to-link layer address mapping facilities (for example, 10Mb/s Ethernets using **arp**(7P)), the entire address specified in the **ioctl( )** is used. A routing table entry for destinations on the network of the interface is installed automatically when an interface's address is set.

IOCTLS

The following **ioctl( )** calls may be used to manipulate IP network interfaces. Unless specified otherwise, the request takes an **ifreq** structure as its parameter. This structure has the form:

```
/* Interface request structure used for socket ioctls.  All */
/* interface ioctls must have parameter definitions which */
/* begin with ifr_name.  The remainder may be interface specific. */

struct ifreq {
#define  IFNAMSIZ       16

     char      ifr_name[IFNAMSIZ];                  /* if name, for example */
                                                    /* "emd1" */
     union {
          struct sockaddr  ifru_addr;
          struct sockaddr  ifru_dstaddr;
          char             ifru_oname[IFNAMSIZ];    /* other if name */
          struct sockaddr  ifru_broadaddr;
          short            ifru_flags;
          int              ifru_metric;
          char             ifru_data[1];            /* interface dependent data */
          char             ifru_enaddr[6];
          int              if_muxid[2];             /* mux id's for arp and ip */
     } ifr_ifru;

#define  ifr_addr       ifr_ifru.ifru_addr          /* address */
#define  ifr_dstaddr    ifr_ifru.ifru_dstaddr       /* other end of p-to-p
                                                       link */
```

```
#define  ifr_oname      ifr_ifru.ifru_oname       /* other if name */
#define  ifr_broadaddr  ifr_ifru.ifru_broadaddr   /* broadcast address */
#define  ifr_flags      ifr_ifru.ifru_flags
                                                  /* flags */
#define  ifr_metric     ifr_ifru.ifru_metric      /* metric */
#define  ifr_data       ifr_ifru.ifru_data        /* for use by interface */
#define  ifr_enaddr     ifr_ifru.ifru_enaddr      /* ethernet address */
};
```

SIOCSIFADDR Set interface address. Following the address assignment, the "initialization" routine for the interface is called.

SIOCGIFADDR Get interface address.

SIOCSIFDSTADDR Set point to point address for interface.

SIOCGIFDSTADDR Get point to point address for interface.

SIOCSIFFLAGS Set interface flags field. If the interface is marked down, any processes currently routing packets through the interface are notified.

SIOCGIFFLAGS Get interface flags.

SIOCGIFCONF Get interface configuration list. This request takes an **ifconf** structure (see below) as a value-result parameter. The **ifc_len** field should be initially set to the size of the buffer pointed to by **ifc_buf**. On return it will contain the length, in bytes, of the configuration list.

SIOGIFNUM Get number of interfaces. This request returns an integer which is the number of interface descriptions (**struct ifreq**) that will be returned by the **SIOCGIFCONF** ioctl; that is, it gives an indication of how large **ifc_len** has to be.

SIOCSIFMTU Set the maximum transmission unit (MTU) size for interface. Place the result of this request in **ifru_metric** field. The MTU has to be smaller than physical MTU limitation (which is reported in the DLPI **DL_INFO_ACK** message).

SIOCGIFMTU Get the maximum transmission unit size for interface. Place the result of this request in **ifru_metric** field.

SIOCSIFMETRIC Set the metric associated with the interface. The metric is used by routine daemons such as **in.routed**(1M).

SIOCGIFMETRIC Get the metric associated with the interface.

SIOCGIFMUXID Get the ip and arp muxid associated with the interface.

SIOCSIFMUXID Set the ip and arp muxid associated with the interface.

The **ifconf** structure has the form:

```
/*
 * Structure used in SIOCGIFCONF request.
 * Used to retrieve interface configuration
 * for machine (useful for programs which
 * must know all networks accessible).
 */
struct ifconf {
    int ifc_len;                        /* size of associated buffer */

    union {
        caddr_t         ifcu_buf;
        struct ifreq    *ifcu_req;
    } ifc_ifcu;

#define   ifc_buf     ifc_ifcu.ifcu_buf     /* buffer address */
#define   ifc_req     ifc_ifcu.ifcu_req     /* array of structures returned */
};
```

ERRORS

EPERM The effective user id of the calling process in not superuser.

ENXIO The **ifr_name** member of the **ifreq** structure contains an invalid value.

EBADADDR Wrong address family or malformed address.

EBUSY For **SIOCSIFFLAGS**, this error is returned when the order of bringing the primary/physical interface (e.g. **le0**) and a secondary/logical interface associated with the same physical interface (e.g. **le0:1**) up or down is violated. The physical interface must be configured up first and cannot be configured down until all the corresponding logical interfaces have been configured down.

EINVAL For **SIOCGIFCONF**, this error is returned when the size of the buffer pointed to by the **ifc_buf** member of the **ifconf** structure is too small.

For **SIOCSIFMTU**, this error is returned when the requested MTU size is invalid. This error indicates the MTU size is greater than the MTU size supported by the DLPI provider or less than **68**.

SEE ALSO

ifconfig(1M), **in.routed**(1M), **ioctl**(2), **arp**(7P), **dlpi**(7P), **ip**(7P), **streamio**(7I)