Rocky Enterprise Linux 9.2 Manual Pages on command 'getifaddrs.3'

$ man getifaddrs.3

GETIFADDRS(3)　　　　　Linux Programmer's Manual　　　　　GETIFADDRS(3)

NAME

getifaddrs, freeifaddrs - get interface addresses

SYNOPSIS

#include <sys/types.h>

#include <ifaddrs.h>

int getifaddrs(struct ifaddrs **ifap);

void freeifaddrs(struct ifaddrs *ifa);

DESCRIPTION

The getifaddrs() function creates a linked list of structures describ?

ing the network interfaces of the local system, and stores the address

of the first item of the list in *ifap. The list consists of ifaddrs

structures, defined as follows:

```
struct ifaddrs {
    struct ifaddrs  *ifa_next;    /* Next item in list */
    char            *ifa_name;    /* Name of interface */
    unsigned int     ifa_flags;   /* Flags from SIOCGIFFLAGS */
    struct sockaddr *ifa_addr;    /* Address of interface */
```

```
    struct sockaddr *ifa_netmask; /* Netmask of interface */

    union {

        struct sockaddr *ifu_broadaddr;

                    /* Broadcast address of interface */

        struct sockaddr *ifu_dstaddr;

                    /* Point-to-point destination address */

    } ifa_ifu;

  #define       ifa_broadaddr ifa_ifu.ifu_broadaddr

  #define        ifa_dstaddr   ifa_ifu.ifu_dstaddr

    void       *ifa_data;   /* Address-specific data */

  };
```

The ifa_next field contains a pointer to the next structure on the

list, or NULL if this is the last item of the list.

The ifa_name points to the null-terminated interface name.

The ifa_flags field contains the interface flags, as returned by the

SIOCGIFFLAGS ioctl(2) operation (see netdevice(7) for a list of these

flags).

The ifa_addr field points to a structure containing the interface ad‐

dress. (The sa_family subfield should be consulted to determine the

format of the address structure.)   This field may contain a null

pointer.

The ifa_netmask field points to a structure containing the netmask as‐

sociated with ifa_addr, if applicable for the address family. This

field may contain a null pointer.

Depending on whether the bit IFF_BROADCAST or IFF_POINTOPOINT is set in

ifa_flags (only one can be set at a time), either ifa_broadaddr will

contain the broadcast address associated with ifa_addr (if applicable

for the address family) or ifa_dstaddr will contain the destination ad‐

dress of the point-to-point interface.

The ifa_data field points to a buffer containing address-family-spe‐

cific data; this field may be NULL if there is no such data for this

interface.

The data returned by getifaddrs() is dynamically allocated and should

be freed using freeifaddrs() when no longer needed.

RETURN VALUE

On success, getifaddrs() returns zero; on error, -1 is returned, and
errno is set appropriately.

ERRORS

getifaddrs() may fail and set errno for any of the errors specified for
socket(2), bind(2), getsockname(2), recvmsg(2), sendto(2), malloc(3),
or realloc(3).

VERSIONS

The getifaddrs() function first appeared in glibc 2.3, but before glibc
2.3.3, the implementation supported only IPv4 addresses; IPv6 support
was added in glibc 2.3.3. Support of address families other than IPv4
is available only on kernels that support netlink.

ATTRIBUTES

For an explanation of the terms used in this section, see at?
tributes(7).

???
?Interface   ? Attribute   ? Value   ?
???
?getifaddrs(), freeifaddrs() ? Thread safety ? MT-Safe ?
???

CONFORMING TO

Not in POSIX.1. This function first appeared in BSDi and is present on
the BSD systems, but with slightly different semantics documented?re?
turning one entry per interface, not per address. This means ifa_addr
and other fields can actually be NULL if the interface has no address,
and no link-level address is returned if the interface has an IP ad?
dress assigned. Also, the way of choosing either ifa_broadaddr or
ifa_dstaddr differs on various systems.

NOTES

The addresses returned on Linux will usually be the IPv4 and IPv6 ad?
dresses assigned to the interface, but also one AF_PACKET address per
interface containing lower-level details about the interface and its

physical layer. In this case, the ifa_data field may contain a pointer

to a struct rtnl_link_stats, defined in <linux/if_link.h> (in Linux 2.4

and earlier, struct net_device_stats, defined in <linux/netdevice.h>),

which contains various interface attributes and statistics.

EXAMPLES

The program below demonstrates the use of getifaddrs(), freeifaddrs(),

and getnameinfo(3).   Here is what we see when running this program on

one system:

    $ ./a.out

    lo   AF_PACKET (17)

            tx_packets =   524; rx_packets =   524

            tx_bytes =   38788; rx_bytes =   38788

    wlp3s0   AF_PACKET (17)

            tx_packets =   108391; rx_packets =   130245

            tx_bytes =   30420659; rx_bytes =   94230014

    em1   AF_PACKET (17)

            tx_packets =   0; rx_packets =   0

            tx_bytes =   0; rx_bytes =   0

    lo   AF_INET (2)

            address: <127.0.0.1>

    wlp3s0   AF_INET (2)

            address: <192.168.235.137>

    lo   AF_INET6 (10)

            address: <::1>

    wlp3s0   AF_INET6 (10)

            address: <fe80::7ee9:d3ff:fef5:1a91%wlp3s0>

  Program source

    #define _GNU_SOURCE   /* To get defns of NI_MAXSERV and NI_MAXHOST */

    #include <arpa/inet.h>

    #include <sys/socket.h>

    #include <netdb.h>

    #include <ifaddrs.h>

    #include <stdio.h>

```c
#include <stdlib.h>

#include <unistd.h>

#include <linux/if_link.h>

int main(int argc, char *argv[])

{

    struct ifaddrs *ifaddr;

    int family, s;

    char host[NI_MAXHOST];

    if (getifaddrs(&ifaddr) == -1) {

        perror("getifaddrs");

        exit(EXIT_FAILURE);

    }

    /* Walk through linked list, maintaining head pointer so we

       can free list later */

    for (struct ifaddrs *ifa = ifaddr; ifa != NULL;

            ifa = ifa->ifa_next) {

        if (ifa->ifa_addr == NULL)

            continue;

        family = ifa->ifa_addr->sa_family;

        /* Display interface name and family (including symbolic

           form of the latter for the common families) */

        printf("%-8s %s (%d)\n",

            ifa->ifa_name,

            (family == AF_PACKET) ? "AF_PACKET" :

            (family == AF_INET) ? "AF_INET" :

            (family == AF_INET6) ? "AF_INET6" : "???",

            family);

        /* For an AF_INET* interface address, display the address */

        if (family == AF_INET || family == AF_INET6) {

            s = getnameinfo(ifa->ifa_addr,

                    (family == AF_INET) ? sizeof(struct sockaddr_in) :

                                sizeof(struct sockaddr_in6),

                    host, NI_MAXHOST,
```

```c
                NULL, 0, NI_NUMERICHOST);
            if (s != 0) {
                printf("getnameinfo() failed: %s\n", gai_strerror(s));
                exit(EXIT_FAILURE);
            }

            printf("\t\taddress: <%s>\n", host);

        } else if (family == AF_PACKET && ifa->ifa_data != NULL) {
            struct rtnl_link_stats *stats = ifa->ifa_data;

            printf("\t\ttx_packets = %10u; rx_packets = %10u\n"
                   "\t\ttx_bytes   = %10u; rx_bytes   = %10u\n",
                   stats->tx_packets, stats->rx_packets,
                   stats->tx_bytes, stats->rx_bytes);
        }
    }

    freeifaddrs(ifaddr);
    exit(EXIT_SUCCESS);
}
```

SEE ALSO

bind(2), getsockname(2), socket(2), packet(7), ifconfig(8)

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/.

GNU   2020-11-01   GETIFADDRS(3)