



Linux Ubuntu 26.04 Manual Pages on command 'freeifaddrs.3'

\$ man freeifaddrs.3

getifaddrs(3) Library Functions Manual getifaddrs(3)

NAME

getifaddrs, freeifaddrs - get interface addresses

LIBRARY

Standard C library (libc, -lc)

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 describing 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 address. (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 associated 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 address of the point-to-point interface.

The `ifa_data` field points to a buffer containing address-family-specific 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 to indicate the error.

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)`.

ATTRIBUTES

For an explanation of the terms used in this section, see `attributes(7)`.


```

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 defs of NI_MAXSERV and NI_MAXHOST */

#include <arpa/inet.h>
#include <sys/socket.h>
#include <netdb.h>
#include <ifaddrs.h>
#include <stdio.h>
#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,
        NULL, 0, NI_NUMERICHOST);
    if (s != 0) {
        printf("getnameinfo() failed: %s\n", gai_strerror(s));
        exit(EXIT_FAILURE);
    }
    printf("\t\t\taddress: <%s>\n", host);
} else if (family == AF_PACKET && ifa->ifa_data != NULL) {
    struct rtnl_link_stats *stats = ifa->ifa_data;
    printf("\t\t\ttx_packets = %10u; rx_packets = %10u\n"
        "\t\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\(8\)](#)

Linux man-pages 6.17

2026-02-08

[getifaddrs\(3\)](#)