



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

\$ man inet_ntoa.3

inet(3) Library Functions Manual inet(3)

NAME

inet_aton, inet_addr, inet_network, inet_ntoa, inet_makeaddr, inet_lnaof, inet_netof - In?
ternet address manipulation routines

LIBRARY

Standard C library (libc, -lc)

SYNOPSIS

```
#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

int inet_aton(const char *cp, struct in_addr *inp);

in_addr_t inet_addr(const char *cp);

in_addr_t inet_network(const char *cp);

[[deprecated]] char *inet_ntoa(struct in_addr in);

[[deprecated]] struct in_addr inet_makeaddr(in_addr_t net,
                                           in_addr_t host);

[[deprecated]] in_addr_t inet_lnaof(struct in_addr in);

[[deprecated]] in_addr_t inet_netof(struct in_addr in);
```

Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

inet_aton(), inet_ntoa():

Since glibc 2.19:

_DEFAULT_SOURCE

In glibc up to and including 2.19:

DESCRIPTION

`inet_aton()` converts the Internet host address `cp` from the IPv4 numbers-and-dots notation into binary form (in network byte order) and stores it in the structure that `inp` points to.

`inet_aton()` returns nonzero if the address is valid, zero if not. The address supplied in `cp` can have one of the following forms:

a.b.c.d Each of the four numeric parts specifies a byte of the address; the bytes are as? signed in left-to-right order to produce the binary address.

a.b.c Parts `a` and `b` specify the first two bytes of the binary address. Part `c` is interpreted as a 16-bit value that defines the rightmost two bytes of the binary address. This notation is suitable for specifying (outmoded) Class B network addresses.

a.b Part `a` specifies the first byte of the binary address. Part `b` is interpreted as a 24-bit value that defines the rightmost three bytes of the binary address. This notation is suitable for specifying (outmoded) Class A network addresses.

a The value `a` is interpreted as a 32-bit value that is stored directly into the binary address without any byte rearrangement.

In all of the above forms, components of the dotted address can be specified in decimal, octal (with a leading 0), or hexadecimal, with a leading 0X. Addresses in any of these forms are collectively termed IPv4 numbers-and-dots notation. The form that uses exactly four decimal numbers is referred to as IPv4 dotted-decimal notation (or sometimes: IPv4 dotted-quad notation).

`inet_aton()` returns 1 if the supplied string was successfully interpreted, or 0 if the string is invalid (`errno` is not set on error).

The `inet_addr()` function converts the Internet host address `cp` from IPv4 numbers-and-dots notation into binary data in network byte order. If the input is invalid, `INADDR_NONE` (usually -1) is returned. Use of this function is problematic because -1 is a valid address (255.255.255.255). Avoid its use in favor of `inet_aton()`, `inet_pton(3)`, or `getaddrinfo(3)`, which provide a cleaner way to indicate error return.

The `inet_network()` function converts `cp`, a string in IPv4 numbers-and-dots notation, into a number in host byte order suitable for use as an Internet network address. On success, the converted address is returned. If the input is invalid, -1 is returned.

The `inet_ntoa()` function converts the Internet host address `in`, given in network byte order,

to a string in IPv4 dotted-decimal notation. The string is returned in a statically allocated buffer, which subsequent calls will overwrite.

The `inet_lnaof()` function returns the local network address part of the Internet address in. The returned value is in host byte order.

The `inet_netof()` function returns the network number part of the Internet address `in`. The returned value is in host byte order.

The `inet_makeaddr()` function is the converse of `inet_netof()` and `inet_lnaof()`. It returns an Internet host address in network byte order, created by combining the network number from `net` with the local address `host`, both in host byte order.

The structure `in_addr` as used in `inet_ntoa()`, `inet_makeaddr()`, `inet_lnaof()`, and `inet_netof()` is defined in `<netinet/in.h>` as:

```
typedef uint32_t in_addr_t;

struct in_addr {
    in_addr_t s_addr;
};
```

ATTRIBUTES

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

[illegible]

STANDARDS

inet_addr()
inet_ntoa()
POSIX.1-2008.
inet_aton()
None.

STANDARDS

inet_addr()
inet_ntoa()

POSIX.1-2001, 4.3BSD.

`inet_lnaof()`, `inet_netof()`, and `inet_makeaddr()` are legacy functions that assume they are dealing with classful network addresses. Classful networking divides IPv4 network addresses into host and network components at byte boundaries, as follows:

Class A This address type is indicated by the value 0 in the most significant bit of the (network byte ordered) address. The network address is contained in the most significant byte, and the host address occupies the remaining three bytes.

Class B This address type is indicated by the binary value 10 in the most significant two bits of the address. The network address is contained in the two most significant bytes, and the host address occupies the remaining two bytes.

Class C This address type is indicated by the binary value 110 in the most significant three bits of the address. The network address is contained in the three most significant bytes, and the host address occupies the remaining byte.

Classful network addresses are now obsolete, having been superseded by Classless Inter-Domain Routing (CIDR), which divides addresses into network and host components at arbitrary bit (rather than byte) boundaries.

NOTES

On x86 architectures, the host byte order is Least Significant Byte first (little endian), whereas the network byte order, as used on the Internet, is Most Significant Byte first (big endian).

EXAMPLES

An example of the use of `inet_aton()` and `inet_ntoa()` is shown below. Here are some example runs:

```
$ ./a.out 226.000.000.037    # Last byte is in octal
226.0.0.31

$ ./a.out 0x7f.1            # First byte is in hex
127.0.0.1
```

Program source

```
#define _DEFAULT_SOURCE
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>

int
```

```

main(int argc, char *argv[])
{
    struct in_addr addr;
    if (argc != 2) {
        fprintf(stderr, "%s <dotted-address>\n", argv[0]);
        exit(EXIT_FAILURE);
    }
    if (inet_aton(argv[1], &addr) == 0) {
        fprintf(stderr, "Invalid address\n");
        exit(EXIT_FAILURE);
    }
    printf("%s\n", inet_ntoa(addr));
    exit(EXIT_SUCCESS);
}

```

SEE ALSO

[byteorder\(3\)](#), [getaddrinfo\(3\)](#), [gethostbyname\(3\)](#), [getnameinfo\(3\)](#), [getnetent\(3\)](#),
[inet_net_pton\(3\)](#), [inet_ntop\(3\)](#), [inet_pton\(3\)](#), [hosts\(5\)](#), [networks\(5\)](#)