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

$ man inet_net_pton.3

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

NAME

    inet_net_pton, inet_net_ntop - Internet network number conversion

SYNOPSIS

    #include <arpa/inet.h>

    int inet_net_pton(int af, const char *pres,

          void *netp, size_t nsize);

    char *inet_net_ntop(int af, const void *netp, int bits,

          char *pres, size_t psize);

    Link with -lresolv.

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

    inet_net_pton(), inet_net_ntop():

      Since glibc 2.20:

        _DEFAULT_SOURCE

      Before glibc 2.20:

        _BSD_SOURCE || _SVID_SOURCE

DESCRIPTION

    These functions convert network numbers between presentation (i.e., printable) format and

    network (i.e., binary) format.

    For both functions, af specifies the address family for the conversion; the only supported

    value is AF_INET.

  inet_net_pton()

    The inet_net_pton() function converts pres, a null-terminated string containing an Inter?

net network number in presentation format to network format. The result of the conver‐
sion, which is in network byte order, is placed in the buffer pointed to by net. (The
netp argument typically points to an in_addr structure.) The nsize argument specifies the
number of bytes available in netp.

On success, inet_net_pton() returns the number of bits in the network number field of the
result placed in netp. For a discussion of the input presentation format and the return
value, see NOTES.

Note: the buffer pointed to by netp should be zeroed out before calling inet_net_pton(),
since the call writes only as many bytes as are required for the network number (or as are
explicitly specified by pres), which may be less than the number of bytes in a complete
network address.

inet_net_ntop()

The inet_net_ntop() function converts the network number in the buffer pointed to by netp
to presentation format; *netp is interpreted as a value in network byte order. The bits
argument specifies the number of bits in the network number in *netp.

The null-terminated presentation-format string is placed in the buffer pointed to by pres.
The psize argument specifies the number of bytes available in pres. The presentation
string is in CIDR format: a dotted-decimal number representing the network address, fol‐
lowed by a slash, and the size of the network number in bits.

RETURN VALUE

On success, inet_net_pton() returns the number of bits in the network number. On error,
it returns -1, and errno is set to indicate the cause of the error.

On success, inet_net_ntop() returns pres. On error, it returns NULL, and errno is set to
indicate the cause of the error.

ERRORS

EAFNOSUPPORT

af specified a value other than AF_INET.

EMSGSIZE

The size of the output buffer was insufficient.

ENOENT (inet_net_pton()) pres was not in correct presentation format.

CONFORMING TO

The inet_net_pton() and inet_net_ntop() functions are nonstandard, but widely available.

NOTES

Input presentation format for inet_net_pton()

The network number may be specified either as a hexadecimal value or in dotted-decimal no?

tation.

Hexadecimal values are indicated by an initial "0x" or "0X". The hexadecimal digits popu?

late the nibbles (half octets) of the network number from left to right in network byte

order.

In dotted-decimal notation, up to four octets are specified, as decimal numbers separated

by dots. Thus, any of the following forms are accepted:

a.b.c.d

a.b.c

a.b

a

Each part is a number in the range 0 to 255 that populates one byte of the resulting net?

work number, going from left to right, in network-byte (big endian) order. Where a part

is omitted, the resulting byte in the network number is zero.

For either hexadecimal or dotted-decimal format, the network number can optionally be fol?

lowed by a slash and a number in the range 0 to 32, which specifies the size of the net?

work number in bits.

Return value of inet_net_pton()

The return value of inet_net_pton() is the number of bits in the network number field. If

the input presentation string terminates with a slash and an explicit size value, then

that size becomes the return value of inet_net_pton(). Otherwise, the return value, bits,

is inferred as follows:

* If the most significant byte of the network number is greater than or equal to 240,

then bits is 32.

* Otherwise, if the most significant byte of the network number is greater than or equal

to 224, then bits is 4.

* Otherwise, if the most significant byte of the network number is greater than or equal

to 192, then bits is 24.

* Otherwise, if the most significant byte of the network number is greater than or equal

to 128, then bits is 16.

* Otherwise, bits is 8.

If the resulting bits value from the above steps is greater than or equal to 8, but the

number of octets specified in the network number exceed bits/8, then bits is set to 8
times the number of octets actually specified.

EXAMPLES

The program below demonstrates the use of inet_net_pton() and inet_net_ntop().   It uses
inet_net_pton() to convert the presentation format network address provided in its first
command-line argument to binary form, displays the return value from inet_net_pton().   It
then uses inet_net_ntop() to convert the binary form back to presentation format, and dis‐
plays the resulting string.

In order to demonstrate that inet_net_pton() may not write to all bytes of its netp argu‐
ment, the program allows an optional second command-line argument, a number used to ini‐
tialize the buffer before inet_net_pton() is called. As its final line of output, the
program displays all of the bytes of the buffer returned by inet_net_pton() allowing the
user to see which bytes have not been touched by inet_net_pton().

An example run, showing that inet_net_pton() infers the number of bits in the network num‐
ber:

    $ ./a.out 193.168
    inet_net_pton() returned: 24
    inet_net_ntop() yielded: 193.168.0/24
    Raw address: c1a80000

Demonstrate that inet_net_pton() does not zero out unused bytes in its result buffer:

    $ ./a.out 193.168 0xffffffff
    inet_net_pton() returned: 24
    inet_net_ntop() yielded: 193.168.0/24
    Raw address: c1a800ff

Demonstrate that inet_net_pton() will widen the inferred size of the network number, if
the supplied number of bytes in the presentation string exceeds the inferred value:

    $ ./a.out 193.168.1.128
    inet_net_pton() returned: 32
    inet_net_ntop() yielded: 193.168.1.128/32
    Raw address: c1a80180

Explicitly specifying the size of the network number overrides any inference about its
size (but any extra bytes that are explicitly specified will still be used by
inet_net_pton(): to populate the result buffer):

```
$ ./a.out 193.168.1.128/24
inet_net_pton() returned: 24
inet_net_ntop() yielded:  193.168.1/24
Raw address:          c1a80180
```

Program source

```c
/* Link with "-lresolv" */

#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#define errExit(msg)    do { perror(msg); exit(EXIT_FAILURE); \
                        } while (0)
int
main(int argc, char *argv[])
{
    char buf[100];
    struct in_addr addr;
    int bits;
    if (argc < 2) {
        fprintf(stderr,
            "Usage: %s presentation-form [addr-init-value]\n",
            argv[0]);
        exit(EXIT_FAILURE);
    }
    /* If argv[2] is supplied (a numeric value), use it to initialize
       the output buffer given to inet_net_pton(), so that we can see
       that inet_net_pton() initializes only those bytes needed for
       the network number. If argv[2] is not supplied, then initialize
       the buffer to zero (as is recommended practice). */
    addr.s_addr = (argc > 2) ? strtod(argv[2], NULL) : 0;
    /* Convert presentation network number in argv[1] to binary */
    bits = inet_net_pton(AF_INET, argv[1], &addr, sizeof(addr));
    if (bits == -1)
        errExit("inet_net_ntop");
```

```
        printf("inet_net_pton() returned: %d\n", bits);

        /* Convert binary format back to presentation, using 'bits'
           returned by inet_net_pton() */

        if (inet_net_ntop(AF_INET, &addr, bits, buf, sizeof(buf)) == NULL)
            errExit("inet_net_ntop");

        printf("inet_net_ntop() yielded:  %s\n", buf);

        /* Display 'addr' in raw form (in network byte order), so we can
           see bytes not displayed by inet_net_ntop(); some of those bytes
           may not have been touched by inet_net_ntop(), and so will still
           have any initial value that was specified in argv[2]. */

        printf("Raw address:            %x\n", htonl(addr.s_addr));

        exit(EXIT_SUCCESS);
    }
```

SEE ALSO

inet(3), networks(5)

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/.

Linux   2020-06-09   INET_NET_PTON(3)