

**NAME**

`inet_pton` - convert IPv4 and IPv6 addresses from text to binary form

**SYNOPSIS**

```
#include <arpa/inet.h>
```

```
int inet_pton(int af, const char *src, void *dst);
```

**DESCRIPTION**

This function converts the character string *src* into a network address structure in the *af* address family, then copies the network address structure to *dst*. The *af* argument must be either **AF\_INET** or **AF\_INET6**.

The following address families are currently supported:

**AF\_INET**

*src* points to a character string containing an IPv4 network address in dotted-decimal format, *ddd.ddd.ddd.ddd*, where *ddd* is a decimal number of up to three digits in the range 0 to 255. The address is converted to a *struct in\_addr* and copied to *dst*, which must be `sizeof(struct in_addr)` (4) bytes (32 bits) long.

**AF\_INET6**

*src* points to a character string containing an IPv6 network address. The address is converted to a *struct in6\_addr* and copied to *dst*, which must be `sizeof(struct in6_addr)` (16) bytes (128 bits) long. The allowed formats for IPv6 addresses follow these rules:

1. The preferred format is *x:x:x:x:x:x:x:x*. This form consists of eight hexadecimal numbers, each of which expresses a 16-bit value (i.e., each *x* can be up to 4 hex digits).
2. A series of contiguous zero values in the preferred format can be abbreviated to *::*. Only one instance of *::* can occur in an address. For example, the loopback address *0:0:0:0:0:0:0:1* can be abbreviated as *::1*. The wildcard address, consisting of all zeros, can be written as *::*.
3. An alternate format is useful for expressing IPv4-mapped IPv6 addresses. This form is written as *x:x:x:x:x:x:d.d.d.d*, where the six leading *xs* are hexadecimal values that define the six most-significant 16-bit pieces of the address (i.e., 96 bits), and the *ds* express a value in dotted-decimal notation that defines the least significant 32 bits of the address. An example of such an address is *::FFFF:204.152.189.116*.

See RFC 2373 for further details on the representation of IPv6 addresses.

**RETURN VALUE**

`inet_pton()` returns 1 on success (network address was successfully converted). 0 is returned if *src* does not contain a character string representing a valid network address in the specified address family. If *af* does not contain a valid address family, -1 is returned and *errno* is set to **EAFNOSUPPORT**.

**CONFORMING TO**

POSIX.1-2001.

**NOTES**

Unlike [inet\\_aton\(3\)](#) and [inet\\_addr\(3\)](#), `inet_pton()` supports IPv6 addresses. On the other hand, `inet_pton()` accepts only IPv4 addresses in dotted-decimal notation, whereas [inet\\_aton\(3\)](#) and [inet\\_addr\(3\)](#) allow the more general numbers-and-dots notation (hexadecimal and octal number formats, and formats that don't require all four bytes to be explicitly written). For an interface that handles both IPv6 addresses, and IPv4 addresses in numbers-and-dots notation, see [getaddrinfo\(3\)](#).

**BUGS**

**AF\_INET6** does not recognize IPv4 addresses. An explicit IPv4-mapped IPv6 address must be supplied in *src* instead.

**EXAMPLE**

The program below demonstrates the use of `inet_pton()` and `inet_ntop(3)`. Here are some example runs:

```
$ ./a.out i6 0:0:0:0:0:0:0
::
$ ./a.out i6 1:0:0:0:0:0:8
1::8
$ ./a.out i6 0:0:0:0:0:FFFF:204.152.189.116
::ffff:204.152.189.116
```

**Program source**

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

int
main(int argc, char *argv[])
{
    unsigned char buf[sizeof(struct in6_addr)];
    int domain, s;
    char str[INET6_ADDRSTRLEN];

    if (argc != 3) {
        fprintf(stderr, Usage: %s {i4|i6|<num>} stringn, argv[0]);
        exit(EXIT_FAILURE);
    }

    domain = (strcmp(argv[1], i4) == 0) ? AF_INET :
             (strcmp(argv[1], i6) == 0) ? AF_INET6 : atoi(argv[1]);

    s = inet_pton(domain, argv[2], buf);
    if (s <= 0) {
        if (s == 0)
            fprintf(stderr, Not in presentation format);
        else
            perror(inet_pton);
        exit(EXIT_FAILURE);
    }

    if (inet_ntop(domain, buf, str, INET6_ADDRSTRLEN) == NULL) {
        perror(inet_ntop);
        exit(EXIT_FAILURE);
    }

    printf(%sn, str);

    exit(EXIT_SUCCESS);
}
```

**SEE ALSO**

[getaddrinfo\(3\)](#), [inet\(3\)](#), [inet\\_ntop\(3\)](#)

**COLOPHON**

This page is part of release 3.74 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 <http://www.kernel.org/doc/man-pages/>.