

**NAME**

getipnodebyname, getipnodebyaddr, freehostent – get IP node entry

**SYNOPSIS**

```
cc [ flag... ] file... -lsocket -lnsl [ library... ]
```

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

```
#include <netdb.h>
```

```
struct hostent *getipnodebyname(const char *name, int af, int flags,
    int *error_num);
```

```
struct hostent *getipnodebyaddr(const void *src, size_t len, int af,
    int *error_num);
```

```
void freehostent(struct hostent *ptr);
```

**PARAMETERS**

<i>af</i>	Address family
<i>flags</i>	Various flags
<i>name</i>	Name of host
<i>error_num</i>	Error storage
<i>src</i>	Address for lookup
<i>len</i>	Length of address
<i>ptr</i>	Pointer to <b>hostent</b> structure

**DESCRIPTION**

The **getipnodebyname()** function searches the **ipnodes** database from the beginning. The function finds the first **h\_name** member that matches the hostname specified by *name*. The function takes an *af* argument that specifies the address family. The address family can be **AF\_INET** for IPv4 addresses or **AF\_INET6** for IPv6 addresses. The *flags* argument determines what results are returned based on the value of *flags*. If the *flags* argument is set to **0** (zero), the default operation of the function is specified as follows:

- If the *af* argument is **AF\_INET**, a query is made for an IPv4 address. If successful, IPv4 addresses are returned and the **h\_length** member of the **hostent** structure is 4. Otherwise, the function returns a *NULL* pointer.
- If the *af* argument is **AF\_INET6**, a query is made for an IPv6 address. If successful, IPv6 addresses are returned and the **h\_length** member of the **hostent** structure is 16. Otherwise, the function returns a *NULL* pointer.

The *flags* argument changes the default actions of the function. Set the *flags* argument with a logical **OR** operation on any of combination of the following values:

- **AI\_V4MAPPED**
- **AI\_ALL**
- **AI\_ADDRCONFIG**

The special flags value, **AI\_DEFAULT**, should handle most applications. Porting simple applications to use IPv6 replaces the call

```
hptr = gethostbyname(name);
```

with

```
hptr = getipnodebyname(name, AF_INET6, AI_DEFAULT, &error_num);
```

The *flags* value **0** (zero) implies a strict interpretation of the *af* argument:

- If *flags* is **0** and *af* is **AF\_INET**, the caller wants only IPv4 addresses. A query is made for **A** records. If successful, IPv4 addresses are returned and the **h\_length** member of the **hostent** structure is 4. Otherwise, the function returns a *NULL* pointer.
- If *flags* is **0** and *af* is **AF\_INET6**, the caller wants only IPv6 addresses. A query is made for **AAAA** records. If successful, IPv6 addresses are returned and the **h\_length** member of the **hostent** structure is 16. Otherwise, the function returns a *NULL* pointer.

Logically **OR** other constants into the *flags* argument to modify the behavior of the **getipnodebyname()** function.

- If the **AI\_V4MAPPED** flag is specified with *af* set to **AF\_INET6**, the caller can accept IPv4-mapped IPv6 addresses. If no **AAAA** records are found, a query is made for **A** records. Any **A** records found are returned as IPv4-mapped IPv6 addresses and the **h\_length** is 16. The **AI\_V4MAPPED** flag is ignored unless *af* equals **AF\_INET6**.
- The **AI\_ALL** flag is used in conjunction with the **AI\_V4MAPPED** flag, exclusively with the IPv6 address family. When **AI\_ALL** is logically **OR**ed with **AI\_V4MAPPED** flag, the caller wants all addresses: IPv6 and IPv4-mapped IPv6 addresses. A query is first made for **AAAA** records and, if successful, IPv6 addresses are returned. Another query is then made for **A** records. Any **A** records found are returned as IPv4-mapped IPv6 addresses and the **h\_length** is 16. Only when both queries fail does the function return a *NULL* pointer. The **AI\_ALL** flag is ignored unless *af* is set to **AF\_INET6**.
- The **AI\_ADDRCONFIG** flag specifies that a query for **AAAA** records should occur only when the node is configured with at least one IPv6 source address. A query for **A** records should occur only when the node is configured with at least one IPv4 source address. For example, if a node is configured with no IPv6 source addresses, *af* equals **AF\_INET6**, and the node name queried has both **AAAA** and **A** records, then:
  - A *NULL* pointer is returned when only the **AI\_ADDRCONFIG** value is specified.
  - The **A** records are returned as IPv4-mapped IPv6 addresses when the **AI\_ADDRCONFIG** and **AI\_V4MAPPED** values are specified.

The special flags value, **AI\_DEFAULT**, is defined as

```
#define AI_DEFAULT (AI_V4MAPPED | AI_ADDRCONFIG)
```

The **getipnodebyname()** function allows the *name* argument to be a node name or a literal address string: a dotted-decimal IPv4 address or an IPv6 hex address. Applications do not have to call **inet\_pton(3SOCKET)** to handle literal address strings.

Four scenarios arise based on the type of literal address string and the value of the *af* argument. The two simple cases occur when *name* is a dotted-decimal IPv4 address and *af* equals **AF\_INET** and when *name* is an IPv6 hex address and *af* equals **AF\_INET6**. The members of the returned **hostent** structure are:

<b>h_name</b>	Pointer to a copy of the name argument
<b>h_aliases</b>	<i>NULL</i> pointer.
<b>h_addrtype</b>	Copy of the <i>af</i> argument.
<b>h_length</b>	4 for <b>AF_INET</b> or 16 for <b>AF_INET6</b> .
<b>h_addr_list</b>	Array of pointers to 4-byte or 16-byte binary addresses. The array is terminated by a <i>NULL</i> pointer.

#### RETURN VALUES

Upon successful completion, **getipnodebyname()** and **getipnodebyaddr()** return a **hostent** structure. Otherwise they return *NULL*.

The **hostent** structure does not change from the existing definition when used with **gethostbyname(3NSL)**. For example, host entries are represented by the **struct hostent** structure defined in `<netdb.h>`:

```
struct hostent {
    char    *h_name;        /* canonical name of host */
    char    **h_aliases;   /* alias list */
    int     h_addrtype;    /* host address type */
    int     h_length;      /* length of address */
    char    **h_addr_list; /* list of addresses */
};
```

An error occurs when *name* is an IPv6 hex address and *af* equals **AF\_INET**. The return value of the function is a *NULL* pointer and **error\_num** equals **HOST\_NOT\_FOUND**.

The **getipnodebyaddr()** function has the same arguments as the existing **gethostbyaddr(3NSL)** function, but adds an error number. As with **getipnodebyname()**, **getipnodebyaddr()** is thread-safe. The **error\_num** value is returned to the caller with the appropriate error code to support thread-safe error code returns. The following error conditions can be returned for **error\_num**:

**HOST\_NOT\_FOUND** Host is unknown.

**NO\_DATA** No address is available for the *name* specified in the server request. This error is not a soft error. Another type of *name* server request might be successful.

**NO\_RECOVERY** An unexpected server failure occurred, which is a non-recoverable error.

**TRY\_AGAIN** This error is a soft error that indicates that the local server did not receive a response from an authoritative server. A retry at some later time might be successful.

One possible source of confusion is the handling of IPv4-mapped IPv6 addresses and IPv4-compatible IPv6 addresses, but the following logic should apply:

1. If *af* is **AF\_INET6**, and if *len* equals 16, and if the IPv6 address is an IPv4-mapped IPv6 address or an IPv4-compatible IPv6 address, then skip over the first 12 bytes of the IPv6 address, set *af* to **AF\_INET**, and set *len* to 4.
2. If *af* is **AF\_INET**, lookup the *name* for the given IPv4 address.
3. If *af* is **AF\_INET6**, lookup the *name* for the given IPv6 address.
4. If the function is returning success, then the single address that is returned in the **hostent** structure is a copy of the first argument to the function with the same address family that was passed as an argument to this function.

All four steps listed are performed in order.

This structure, and the information pointed to by this structure, are dynamically allocated by **getipnodebyname()** and **getipnodebyaddr()**. The **freehostent()** function frees this memory.

#### EXAMPLES

**Example 1** Getting the Canonical Name, Aliases, and Internet IP Addresses for a Given Hostname

The following is a sample program that retrieves the canonical name, aliases, and all Internet IP addresses, both version 6 and version 4, for a given hostname.

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>

main(int argc, const char **argv)
{
    char abuf[INET6_ADDRSTRLEN];
    int error_num;
    struct hostent *hp;
    char **p;

    if (argc != 2) {
        (void) printf("usage: %s hostname0, argv[0]);
        exit (1);
    }

    /* argv[1] can be a pointer to a hostname or literal IP address */
    hp = getipnodebyname(argv[1], AF_INET6, AI_ALL | AI_ADDRCONFIG |
        AI_V4MAPPED, &error_num);
    if (hp == NULL) {
```

```

    if (error_num == TRY_AGAIN) {
        printf("%s: unknown host or invalid literal address "
            "(try again later)0, argv[1]);
    } else {
        printf("%s: unknown host or invalid literal address0,
            argv[1]);
    }
    exit (1);
}
for (p = hp->h_addr_list; *p != 0; p++) {
    struct in6_addr in6;
    char **q;

    bcopy(*p, (caddr_t)&in6, hp->h_length);
    (void) printf("%s%s", inet_ntop(AF_INET6, (void *)&in6,
        abuf, sizeof(abuf)), hp->h_name);
    for (q = hp->h_aliases; *q != 0; q++)
        (void) printf(" %s", *q);
    (void) putchar('\0');
}
freehostent(hp);
exit (0);
}

```

**ATTRIBUTES**

See **attributes(5)** for descriptions of the following attributes:

tab() box; cw(2.75i) |cw(2.75i) lw(2.75i) |lw(2.75i) ATTRIBUTE TYPEATTRIBUTE VALUE \_ Interface StabilityEvolving \_ MT-LevelSafe

**SEE ALSO**

**getaddrinfo(3SOCKET)**, **gethostbyname(3NSL)**, **htonl(3SOCKET)**, **inet(3SOCKET)**, **netdb.h(3HEAD)**, **hosts(4)**, **nsswitch.conf(4)**, **attributes(5)**

**NOTES**

No enumeration functions are provided for IPv6. Existing enumeration functions such as **sethostent(3NSL)** do not work in combination with the **getipnodebyname()** and **getipnodebyaddr()** functions.

All the functions that return a **struct hostent** must always return the canonical in the **h\_name** field. This name, by definition, is the well-known and official hostname shared between all aliases and all addresses. The underlying source that satisfies the request determines the mapping of the input name or address into the set of names and addresses in **hostent**. Different sources might make such a determination in different ways. If more than one alias and more than one address in **hostent** exist, no pairing is implied between the alias and address.

The current implementations of these functions return or accept only addresses for the Internet address family (type **AF\_INET**) or the Internet address family Version 6 (type **AF\_INET6**).

The form for an address of type **AF\_INET** is a **struct in\_addr** defined in **<netinet/in.h>**. The form for an address of type **AF\_INET6** is a **struct in6\_addr**, also defined in **<netinet/in.h>**. The functions described in **inet\_ntop(3SOCKET)** and **inet\_pton(3SOCKET)** that are illustrated in the **EXAMPLES** section are helpful in constructing and manipulating addresses in either of these forms.