NAME

getipnodebyname, **getipnodebyaddr**, **freehostent** - nodename-to-address and address-to-nodename translation

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <sys/types.h>
#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*);

DESCRIPTION

The **getipnodebyname**() and **getipnodebyaddr**() functions are very similar to gethostbyname(3), gethostbyname2(3) and gethostbyaddr(3). The functions cover all the functionalities provided by the older ones, and provide better interface to programmers. The functions require additional arguments, *af*, and *flags*, for specifying address family and operation mode. The additional arguments allow programmer to get address for a nodename, for specific address family (such as AF_INET or AF_INET6). The functions also require an additional pointer argument, *error_num* to return the appropriate error code, to support thread safe error code returns.

The type and usage of the return value, struct hostent is described in gethostbyname(3).

For **getipnodebyname**(), the *name* argument can be either a node name or a numeric address string (i.e., a dotted-decimal IPv4 address or an IPv6 hex address). The *af* argument specifies the address family, either AF_INET or AF_INET6. The *flags* argument specifies the types of addresses that are searched for, and the types of addresses that are returned. We note that a special flags value of AI_DEFAULT (defined below) should handle most applications. That is, porting simple applications to use IPv6 replaces the call

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

with

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

Applications desiring finer control over the types of addresses searched for and returned, can specify other combinations of the *flags* argument.

A *flags* of 0 implies a strict interpretation of the *af* argument:

- If *flags* is 0 and *af* is AF_INET, then the caller wants only IPv4 addresses. A query is made for A records. If successful, the IPv4 addresses are returned and the h_length member of the hostent structure will be 4, else the function returns a NULL pointer.

- If *flags* is 0 and if *af* is AF_INET6, then the caller wants only IPv6 addresses. A query is made for AAAA records. If successful, the IPv6 addresses are returned and the h_length member of the hostent structure will be 16, else the function returns a NULL pointer.

Other constants can be logically-ORed into the *flags* argument, to modify the behavior of the function.

- If the AI_V4MAPPED flag is specified along with an *af* of AF_INET6, then the caller will accept IPv4-mapped IPv6 addresses. That is, if no AAAA records are found then a query is made for A records and any found are returned as IPv4-mapped IPv6 addresses (h_length will be 16). The AI_V4MAPPED flag is ignored unless *af* equals AF_INET6.

- The AI_V4MAPPED_CFG flag is exact same as the AI_V4MAPPED flag only if the kernel supports IPv4-mapped IPv6 address.

- If the AI_ALL flag is used in conjunction with the AI_V4MAPPED flag, and only used with the IPv6 address family. When AI_ALL is logically or'd with AI_V4MAPPED flag then the caller wants all addresses: IPv6 and IPv4-mapped IPv6. A query is first made for AAAA records and if successful, the IPv6 addresses are returned. Another query is then made for A records and any found are returned as IPv4-mapped IPv6 addresses. h_length will be 16. Only if both queries fail does the function return a NULL pointer. This flag is ignored unless af equals AF_INET6. If both AI_ALL and AI_V4MAPPED are specified, AI_ALL takes precedence.

- The AI_ADDRCONFIG flag specifies that a query for AAAA records should occur only if the node has at least one IPv6 source address configured and a query for A records should occur only if the node has at least one IPv4 source address configured.

  For example, if the node has no IPv6 source addresses configured, and *af* equals AF_INET6, and the node name being looked up has both AAAA and A records, then: (a) if only AI_ADDRCONFIG is specified, the function returns a NULL pointer; (b) if AI_ADDRCONFIG | AI_V4MAPPED is specified, the A records are returned as IPv4-mapped IPv6 addresses;

The special flags value of AI_DEFAULT is defined as

```
#define  AI_DEFAULT  (AI_V4MAPPED_CFG | AI_ADDRCONFIG)
```

We noted that the **getipnodebyname**() function must allow the *name* argument to be either a node name or a literal address string (i.e., a dotted-decimal IPv4 address or an IPv6 hex address). This saves applications from having to call inet_pton(3) to handle literal address strings. When the *name* argument is a literal address string, the *flags* argument is always ignored.

There are four scenarios based on the type of literal address string and the value of the *af* argument. The two simple cases are when *name* is a dotted-decimal IPv4 address and *af* equals AF_INET, or when *name* is an IPv6 hex address and *af* equals AF_INET6. The members of the returned hostent structure are: h_name points to a copy of the *name* argument, h_aliases is a NULL pointer, h_addrtype is a copy of the *af* argument, h_length is either 4 (for AF_INET) or 16 (for AF_INET6), h_addr_list[0] is a pointer to the 4-byte or 16-byte binary address, and h_addr_list[1] is a NULL pointer.

When *name* is a dotted-decimal IPv4 address and *af* equals AF_INET6, and AI_V4MAPPED is specified, an IPv4-mapped IPv6 address is returned: h_name points to an IPv6 hex address containing the IPv4-mapped IPv6 address, h_aliases is a NULL pointer, h_addrtype is AF_INET6, h_length is 16, h_addr_list[0] is a pointer to the 16-byte binary address, and h_addr_list[1] is a NULL pointer.

It is an error when *name* is an IPv6 hex address and *af* equals AF_INET. The function's return value is a NULL pointer and the value pointed to by *error_num* equals HOST_NOT_FOUND.

The **getipnodebyaddr**() function takes almost the same argument as gethostbyaddr(3), but adds a pointer to return an error number. Additionally it takes care of IPv4-mapped IPv6 addresses, and IPv4-compatible IPv6 addresses.

The **getipnodebyname**() and **getipnodebyaddr**() functions dynamically allocate the structure to be returned to the caller. The **freehostent**() function reclaims memory region allocated and returned by **getipnodebyname**() or **getipnodebyaddr**().

FILES

/etc/hosts
/etc/nsswitch.conf
/etc/resolv.conf

DIAGNOSTICS

The **getipnodebyname**() and **getipnodebyaddr**() functions returns NULL on errors. The integer values pointed to by *error_num* may then be checked to see whether this is a temporary failure or an invalid or unknown host. The meanings of each error code are described in gethostbyname(3).

SEE ALSO

getaddrinfo(3), gethostbyaddr(3), gethostbyname(3), getnameinfo(3), hosts(5), nsswitch.conf(5), services(5), hostname(7)

R. Gilligan, S. Thomson, J. Bound, and W. Stevens, *Basic Socket Interface Extensions for IPv6*, RFC2553, March 1999.

STANDARDS

The **getipnodebyname**() and **getipnodebyaddr**() functions are documented in "Basic Socket Interface Extensions for IPv6" (RFC2553).

HISTORY

The implementation first appeared in KAME advanced networking kit.

BUGS

The **getipnodebyname**() and **getipnodebyaddr**() functions do not handle scoped IPv6 address properly. If you use these functions, your program will not be able to handle scoped IPv6 addresses. For IPv6 address manipulation, **getaddrinfo**(*3*) and **getnameinfo**(*3*) are recommended.

The text was shamelessly copied from RFC2553.