

**NAME**

gethostbyname, gethostbyaddr, sethostent, gethostent, endhostent, h\_errno, perror, hstrerror, gethostbyaddr\_r, gethostbyname2, gethostbyname2\_r, gethostbyname\_r, gethostent\_r – get network host entry

**LIBRARY**

Standard C library (*libc*, *-lc*)

**SYNOPSIS**

```
#include <netdb.h>

void sethostent(int stayopen);
void endhostent(void);

[[deprecated]] extern int h_errno;

[[deprecated]] struct hostent *gethostbyname(const char *name);
[[deprecated]] struct hostent *gethostbyaddr(const void addr[len],
                                             socklen_t len, int type);

[[deprecated]] void perror(const char *s);
[[deprecated]] const char *hstrerror(int err);

/* System V/POSIX extension */
struct hostent *gethostent(void);

/* GNU extensions */
[[deprecated]]
struct hostent *gethostbyname2(const char *name, int af);

int gethostent_r(struct hostent *restrict ret,
                 char buf[restrict .buflen], size_t buflen,
                 struct hostent **restrict result,
                 int *restrict h_errnop);

[[deprecated]]
int gethostbyaddr_r(const void addr[restrict .len], socklen_t len,
                   int type,
                   struct hostent *restrict ret,
                   char buf[restrict .buflen], size_t buflen,
                   struct hostent **restrict result,
                   int *restrict h_errnop);

[[deprecated]]
int gethostbyname_r(const char *restrict name,
                   struct hostent *restrict ret,
                   char buf[restrict .buflen], size_t buflen,
                   struct hostent **restrict result,
                   int *restrict h_errnop);

[[deprecated]]
int gethostbyname2_r(const char *restrict name, int af,
                    struct hostent *restrict ret,
                    char buf[restrict .buflen], size_t buflen,
                    struct hostent **restrict result,
                    int *restrict h_errnop);
```

Feature Test Macro Requirements for glibc (see **feature\_test\_macros(7)**):

**gethostbyname2(), gethostent\_r(), gethostbyaddr\_r(), gethostbyname\_r(), gethostbyname2\_r():**

Since glibc 2.19:

`_DEFAULT_SOURCE`

glibc up to and including 2.19:

`_BSD_SOURCE || _SVID_SOURCE`

**herror(), hstrerror():**

Since glibc 2.19:

`_DEFAULT_SOURCE`

glibc 2.8 to glibc 2.19:

`_BSD_SOURCE` || `_SVID_SOURCE`

Before glibc 2.8:

none

**h\_errno:**

Since glibc 2.19

`_DEFAULT_SOURCE` || `_POSIX_C_SOURCE < 200809L`

glibc 2.12 to glibc 2.19:

`_BSD_SOURCE` || `_SVID_SOURCE` || `_POSIX_C_SOURCE < 200809L`

Before glibc 2.12:

none

**DESCRIPTION**

The **gethostbyname\*()**, **gethostbyaddr\*()**, **herror()**, and **hstrerror()** functions are obsolete. Applications should use **getaddrinfo(3)**, **getnameinfo(3)**, and **gai\_strerror(3)** instead.

The **sethostent()** function specifies, if *stayopen* is true (1), that a connected TCP socket should be used for the name server queries and that the connection should remain open during successive queries. Otherwise, name server queries will use UDP datagrams.

The **endhostent()** function ends the use of a TCP connection for name server queries.

The **gethostbyname()** function returns a structure of type *hostent* for the given host *name*. Here *name* is either a hostname or an IPv4 address in standard dot notation (as for **inet\_addr(3)**). If *name* is an IPv4 address, no lookup is performed and **gethostbyname()** simply copies *name* into the *h\_name* field and its *struct in\_addr* equivalent into the *h\_addr\_list[0]* field of the returned *hostent* structure. If *name* doesn't end in a dot and the environment variable **HOSTALIASES** is set, the alias file pointed to by **HOSTALIASES** will first be searched for *name* (see **hostname(7)** for the file format). The current domain and its parents are searched unless *name* ends in a dot.

The **gethostbyaddr()** function returns a structure of type *hostent* for the given host address *addr* of length *len* and address type *type*. Valid address types are **AF\_INET** and **AF\_INET6** (defined in `<sys/socket.h>`). The host address argument is a pointer to a struct of a type depending on the address type, for example a *struct in\_addr \** (probably obtained via a call to **inet\_addr(3)**) for address type **AF\_INET**.

The (obsolete) **herror()** function prints the error message associated with the current value of *h\_errno* on *stderr*.

The (obsolete) **hstrerror()** function takes an error number (typically *h\_errno*) and returns the corresponding message string.

The domain name queries carried out by **gethostbyname()** and **gethostbyaddr()** rely on the Name Service Switch (**nsswitch.conf(5)**) configured sources or a local name server (**named(8)**). The default action is to query the Name Service Switch (**nsswitch.conf(5)**) configured sources, failing that, a local name server (**named(8)**).

**Historical**

The **nsswitch.conf(5)** file is the modern way of controlling the order of host lookups.

In glibc 2.4 and earlier, the *order* keyword was used to control the order of host lookups as defined in `/etc/host.conf` (**host.conf(5)**).

The *hostent* structure is defined in `<netdb.h>` as follows:

```
struct hostent {
    char    *h_name;                /* official 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 */
    }
    #define h_addr h_addr_list[0] /* for backward compatibility */

```

The members of the *hostent* structure are:

*h\_name*

The official name of the host.

*h\_aliases*

An array of alternative names for the host, terminated by a null pointer.

*h\_addrtype*

The type of address; always **AF\_INET** or **AF\_INET6** at present.

*h\_length*

The length of the address in bytes.

*h\_addr\_list*

An array of pointers to network addresses for the host (in network byte order), terminated by a null pointer.

*h\_addr* The first address in *h\_addr\_list* for backward compatibility.

## RETURN VALUE

The **gethostbyname()** and **gethostbyaddr()** functions return the *hostent* structure or a null pointer if an error occurs. On error, the *h\_errno* variable holds an error number. When non-NULL, the return value may point at static data, see the notes below.

## ERRORS

The variable *h\_errno* can have the following values:

### HOST\_NOT\_FOUND

The specified host is unknown.

### NO\_DATA

The requested name is valid but does not have an IP address. Another type of request to the name server for this domain may return an answer. The constant **NO\_ADDRESS** is a synonym for **NO\_DATA**.

### NO\_RECOVERY

A nonrecoverable name server error occurred.

### TRY\_AGAIN

A temporary error occurred on an authoritative name server. Try again later.

## FILES

*/etc/host.conf*

resolver configuration file

*/etc/hosts*

host database file

*/etc/nsswitch.conf*

name service switch configuration

## ATTRIBUTES

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

| Interface   | Attribute     | Value   |
|---|---------------|---|
| <b>gethostbyname()</b>  | Thread safety | MT-Unsafe race:hostbyname env locale              |
| <b>gethostbyaddr()</b>  | Thread safety | MT-Unsafe race:hostbyaddr env locale              |
| <b>sethostent()</b> , <b>endhostent()</b> ,<br><b>gethostent_r()</b>                  | Thread safety | MT-Unsafe race:hostent env locale                 |
| <b>herror()</b> , <b>hstrerror()</b>  | Thread safety | MT-Safe   |
| <b>gethostent()</b>   | Thread safety | MT-Unsafe race:hostent race:hostentbuf env locale |
| <b>gethostbyname2()</b>   | Thread safety | MT-Unsafe race:hostbyname2 env locale             |
| <b>gethostbyaddr_r()</b> ,<br><b>gethostbyname_r()</b> ,<br><b>gethostbyname2_r()</b> | Thread safety | MT-Safe env locale                                |

In the above table, *hostent* in *race:hostent* signifies that if any of the functions **sethostent()**, **gethostent()**, **gethostent\_r()**, or **endhostent()** are used in parallel in different threads of a program, then data races could occur.

## STANDARDS

POSIX.1-2001 specifies **gethostbyname()**, **gethostbyaddr()**, **sethostent()**, **endhostent()**, **gethostent()**, and *h\_errno*; **gethostbyname()**, **gethostbyaddr()**, and *h\_errno* are marked obsolescent in that standard. POSIX.1-2008 removes the specifications of **gethostbyname()**, **gethostbyaddr()**, and *h\_errno*, recommending the use of **getaddrinfo(3)** and **getnameinfo(3)** instead.

## NOTES

The functions **gethostbyname()** and **gethostbyaddr()** may return pointers to static data, which may be overwritten by later calls. Copying the *struct hostent* does not suffice, since it contains pointers; a deep copy is required.

In the original BSD implementation the *len* argument of **gethostbyname()** was an *int*. The SUSv2 standard is buggy and declares the *len* argument of **gethostbyaddr()** to be of type *size\_t*. (That is wrong, because it has to be *int*, and *size\_t* is not. POSIX.1-2001 makes it *socklen\_t*, which is OK.) See also **accept(2)**.

The BSD prototype for **gethostbyaddr()** uses *const char \** for the first argument.

## System V/POSIX extension

POSIX requires the **gethostent()** call, which should return the next entry in the host data base. When using DNS/BIND this does not make much sense, but it may be reasonable if the host data base is a file that can be read line by line. On many systems, a routine of this name reads from the file */etc/hosts*. It may be available only when the library was built without DNS support. The glibc version will ignore ipv6 entries. This function is not reentrant, and glibc adds a reentrant version **gethostent\_r()**.

## GNU extensions

glibc2 also has a **gethostbyname2()** that works like **gethostbyname()**, but permits to specify the address family to which the address must belong.

glibc2 also has reentrant versions **gethostent\_r()**, **gethostbyaddr\_r()**, **gethostbyname\_r()**, and **gethostbyname2\_r()**. The caller supplies *ahostent* structure *r* et which will be filled in on success, and a temporary work buffer *buf* of size *buflen*. After the call, *result* will point to the result on success. In case of an error or if no entry is found *result* will be NULL. The functions return 0 on success and a nonzero error number on failure. In addition to the errors returned by the nonreentrant versions of these functions, if *buf* is too small, the functions will return **ERANGE**, and the call should be retried with a larger buffer. The global variable *h\_errno* is not modified, but the address of a variable in which to store error numbers is passed in *h\_errnop*.

## BUGS

**gethostbyname()** does not recognize components of a dotted IPv4 address string that are expressed in hexadecimal.

**SEE ALSO**

**getaddrinfo(3), getnameinfo(3), inet(3), inet\_ntop(3), inet\_pton(3), resolver(3), hosts(5), nsswitch.conf(5), hostname(7), named(8)**