



Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 26.04 Manual Pages on command 'herror.3'

\$ man herror.3

gethostbyname(3) Library Functions Manual gethostbyname(3)

NAME

gethostbyname, gethostbyaddr, sethostent, gethostent, endhostent, h_errno, 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(socklen_t size;
                                const void addr[size],
                                socklen_t size, int type);

[[deprecated]] void herror(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(size_t bufsz;
    struct hostent *restrict ret,
    char buf[restrict bufsz], size_t bufsz,
    struct hostent **restrict result,
    int *restrict h_errnop);
```

[[deprecated]]

```
int gethostbyaddr_r(socklen_t size, size_t bufsz;
    const void addr[restrict size], socklen_t size,
    int type,
    struct hostent *restrict ret,
    char buf[restrict bufsz], size_t bufsz,
    struct hostent **restrict result,
    int *restrict h_errnop);
```

[[deprecated]]

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

[[deprecated]]

```
int gethostbyname2_r(size_t bufsz;
    const char *restrict name, int af,
    struct hostent *restrict ret,
    char buf[restrict bufsz], size_t bufsz,
    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 size `size` 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 `de?` pending 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) perror() 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;         /* size 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 size 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.

RETURN VALUE

ERRORS

HOST NOT FOUND

NO DATA

NO RECOVERY

TRY AGAIN

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)`.

[illegible]

?	Interface	?	Attribute	?	Value	?

[illegible]

? gethostname()	? Thread safety ? MT-Unsafe race:hostname env locale	?
-----------------	--	---

[illegible]

? gethostbyaddr() ? Thread safety ? MT-Unsafe race:hostbyaddr env locale ?

Page 5/8

[illegible]

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

sethostent()
endhostent()
gethostent()
POSIX.1-2008.
gethostent_r()
GNU.
Others:
None.

HISTORY

```
sethostent()
endhostent()
gethostent()
    POSIX.1-2001.
gethostbyname()
```

gethostbyaddr()

h_errno

Marked obsolescent in POSIX.1-2001. Removed in POSIX.1-2008, 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 size argument of gethostbyname() was an int. The SUSv2 standard is buggy and declares the size 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 a hostent structure ret which will be filled in on success, and a temporary work buffer buf of size bufsize. 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)

Linux man-pages 6.17

2026-02-08

gethostbyname(3)