

**NAME**

gethostbyname, gethostbyname\_r, gethostbyaddr, gethostbyaddr\_r, gethostent, gethostent\_r, sethostent, endhostent – get network host entry

**SYNOPSIS**

```
cc [ flag... ] file... -lnsl [ library... ]
#include <netdb.h>
```

```
struct hostent *gethostbyname(const char *name);
```

```
struct hostent *gethostbyname_r(const char *name, struct hostent *result, char *buffer, int buflen, int *h_errnop);
```

```
struct hostent *gethostbyaddr(const char *addr, int len, int type);
```

```
struct hostent *gethostbyaddr_r(const char *addr, int length, int type, struct hostent *result, char *buffer, int buflen, int *h_errnop);
```

```
struct hostent *gethostent(void);
```

```
struct hostent *gethostent_r(struct hostent *result, char *buffer, int buflen, int *h_errnop);
```

```
int sethostent(int stayopen);
```

```
int endhostent(void);
```

**DESCRIPTION**

These functions are used to obtain entries describing hosts. An entry can come from any of the sources for **hosts** specified in the `/etc/nsswitch.conf` file. See `nsswitch.conf(4)`. These functions have been superseded by `getipnodebyname(3SOCKET)`, `getipnodebyaddr(3SOCKET)`, and `getaddrinfo(3SOCKET)`, which provide greater portability to applications when multithreading is performed or technologies such as IPv6 are used. For example, the functions described in the following cannot be used with applications targeted to work with IPv6.

The `gethostbyname()` function searches for information for a host with the hostname specified by the character-string parameter *name*.

The `gethostbyaddr()` function searches for information for a host with a given host address. The parameter *type* specifies the family of the address. This should be one of the address families defined in `<sys/socket.h>`. See the **NOTES** section for more information. Also see the **EXAMPLES** section for information on how to convert an Internet **IP** address notation that is separated by periods (.) into an *addr* parameter. The parameter *len* specifies the length of the buffer indicated by *addr*.

All addresses are returned in network order. In order to interpret the addresses, `byteorder(3SOCKET)` must be used for byte order conversion.

The `sethostent()`, `gethostent()`, and `endhostent()` functions are used to enumerate host entries from the database.

The `sethostent()` function sets or resets the enumeration to the beginning of the set of host entries. This function should be called before the first call to `gethostent()`. Calls to `gethostbyname()` and `gethostbyaddr()` leave the enumeration position in an indeterminate state. If the *stayopen* flag is non-zero, the system can keep allocated resources such as open file descriptors until a subsequent call to `endhostent()`.

Successive calls to the `gethostent()` function return either successive entries or `NULL`, indicating the end of the enumeration.

The `endhostent()` function can be called to indicate that the caller expects to do no further host entry retrieval operations; the system can then deallocate resources it was using. It is still allowed, but possibly less efficient, for the process to call more host retrieval functions after calling `endhostent()`.

**Reentrant Interfaces**

The **gethostbyname()**, **gethostbyaddr()**, and **gethostent()** functions use static storage that is reused in each call, making these functions unsafe for use in multithreaded applications.

The **gethostbyname\_r()**, **gethostbyaddr\_r()**, and **gethostent\_r()** functions provide reentrant interfaces for these operations.

Each reentrant interface performs the same operation as its non-reentrant counterpart, named by removing the **\_r** suffix. The reentrant interfaces, however, use buffers supplied by the caller to store returned results and the interfaces are safe for use in both single-threaded and multithreaded applications.

Each reentrant interface takes the same parameters as its non-reentrant counterpart, as well as the following additional parameters. The parameter *result* must be a pointer to a **struct hostent** structure allocated by the caller. On successful completion, the function returns the host entry in this structure. The parameter *buffer* must be a pointer to a buffer supplied by the caller. This buffer is used as storage space for the host data. All of the pointers within the returned **struct hostent result** point to data stored within this buffer. See the **RETURN VALUES** section for more information. The buffer must be large enough to hold all of the data associated with the host entry. The parameter *buflen* should give the size in bytes of the buffer indicated by *buffer*. The parameter *h\_errnop* should be a pointer to an integer. An integer error status value is stored there on certain error conditions. See the **ERRORS** section for more information.

For enumeration in multithreaded applications, the position within the enumeration is a process-wide property shared by all threads. The **sethostent()** function can be used in a multithreaded application but resets the enumeration position for all threads. If multiple threads interleave calls to **gethostent\_r()**, the threads will enumerate disjoint subsets of the host database.

Like their non-reentrant counterparts, **gethostbyname\_r()** and **gethostbyaddr\_r()** leave the enumeration position in an indeterminate state.

**RETURN VALUES**

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 */
};
```

See the **EXAMPLES** section for information about how to retrieve a “.” separated Internet **IP** address string from the *h\_addr\_list* field of **struct hostent**.

The **gethostbyname()**, **gethostbyname\_r()**, **gethostbyaddr()**, and **gethostbyaddr\_r()** functions each return a pointer to a **struct hostent** if they successfully locate the requested entry; otherwise they return **NULL**.

The **gethostent()** and **gethostent\_r()** functions each return a pointer to a **struct hostent** if they successfully enumerate an entry; otherwise they return **NULL**, indicating the end of the enumeration.

The **gethostbyname()**, **gethostbyaddr()**, and **gethostent()** functions use static storage, so returned data must be copied before a subsequent call to any of these functions if the data is to be saved.

When the pointer returned by the reentrant functions **gethostbyname\_r()**, **gethostbyaddr\_r()**, and **gethostent\_r()** is not **NULL**, it is always equal to the *result* pointer that was supplied by the caller.

The **sethostent()** and **endhostent()** functions return **0** on success.

**ERRORS**

The reentrant functions **gethostbyname\_r()**, **gethostbyaddr\_r()**, and **gethostent\_r()** will return *NULL* and set *errno* to **ERANGE** if the length of the buffer supplied by caller is not large enough to store the result. See **Intro(2)** for the proper usage and interpretation of **errno** in multithreaded applications.

The reentrant functions **gethostbyname\_r()** and **gethostbyaddr\_r()** set the integer pointed to by *h\_errnop* to one of these values in case of error.

On failures, the non-reentrant functions **gethostbyname()** and **gethostbyaddr()** set a global integer *h\_errno* to indicate one of these error codes (defined in `<netdb.h>`): **HOST\_NOT\_FOUND**, **TRY\_AGAIN**, **NO\_RECOVERY**, **NO\_DATA**, and **NO\_ADDRESS**.

If a resolver is provided with a malformed address, or if any other error occurs before **gethostbyname()** is resolved, then **gethostbyname()** returns an internal error with a value of `-1`.

The **gethostbyname()** function will set *h\_errno* to **NETDB\_INTERNAL** when it returns a *NULL* value.

**EXAMPLES****Example 1: Using gethostbyname()**

Here is a sample program that gets the canonical name, aliases, and “.” separated Internet **IP** addresses for a given “.” separated **IP** address:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
int main(int argc, const char **argv)
{
    in_addr_t addr;
    struct hostent *hp;
    char **p;
    if (argc != 2) {
        (void) printf("usage: %s IP-address0, argv[0]);
        exit (1);
    }
    if ((int)(addr = inet_addr(argv[1])) == -1) {
        (void) printf("IP-address must be of the form a.b.c.d0);
        exit (2);
    }
    hp = gethostbyaddr((char *)&addr, 4, AF_INET);
    if (hp == NULL) {
        (void) printf("host information for %s not found0, argv[1]);
        exit (3);
    }
    for (p = hp->h_addr_list; *p != 0; p++) {
        struct in_addr in;
        char **q;
        (void) memcpy(&in.s_addr, *p, sizeof (in.s_addr));
        (void) printf("%s%s", inet_ntoa(in), hp->h_name);
        for (q = hp->h_aliases; *q != 0; q++)
            (void) printf(" %s", *q);
        (void) putchar('0');
```

```

    }
    exit (0);
}

```

Note that the preceding sample program is unsafe for use in multithreaded applications.

## FILES

**/etc/hosts**

**/etc/netconfig**

**/etc/nsswitch.conf**

## ATTRIBUTES

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

```

tab() allbox; cw(2.750000i)| cw(2.750000i) lw(2.750000i)| lw(2.750000i).  ATTRIBUTE
TYPEATTRIBUTE VALUE MT-LevelT{ See Reentrant Interfaces in the DESCRIPTION section.
T}

```

## SEE ALSO

**Intro(2)**, **Intro(3)**, **byteorder(3SOCKET)**, **inet(3SOCKET)**, **netdb.h(3HEAD)**, **netdir(3NSL)**, **hosts(4)**, **netconfig(4)**, **nss(4)**, **nsswitch.conf(4)**, **attributes(5)**

## WARNINGS

The reentrant interfaces **gethostbyname\_r()**, **gethostbyaddr\_r()**, and **gethostent\_r()** are included in this release on an uncommitted basis only and are subject to change or removal in future minor releases.

## NOTES

To ensure that they all return consistent results, **gethostbyname()**, **gethostbyname\_r()**, and **netdir\_getbyname()** are implemented in terms of the same internal library function. This function obtains the system-wide source lookup policy based on the **inet** family entries in **netconfig(4)** and the **hosts:** entry in **nsswitch.conf(4)**. Similarly, **gethostbyaddr()**, **gethostbyaddr\_r()**, and **netdir\_getbyaddr()** are implemented in terms of the same internal library function. If the **inet** family entries in **netconfig(4)** have a “-” in the last column for **nametoaddr** libraries, then the entry for **hosts** in **nsswitch.conf** will be used; **nametoaddr** libraries in that column will be used, and **nsswitch.conf** will not be consulted.

There is no analogue of **gethostent()** and **gethostent\_r()** in the **netdir** functions, so these enumeration functions go straight to the **hosts** entry in **nsswitch.conf**. Thus enumeration can return results from a different source than that used by **gethostbyname()**, **gethostbyname\_r()**, **gethostbyaddr()**, and **gethostbyaddr\_r()**.

All the functions that return a **struct hostent** must always return the *canonical name* 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 do that in different ways. If there is more than one alias and more than one address in **hostent**, no pairing is implied between them.

The system attempts to put those addresses that are on the same subnet as the caller before addresses that are on different subnets. However, if address sorting is disabled by setting **SORT\_ADDRS** to **FALSE** in the **/etc/default/nss** file, the system does not put the local subnet addresses first. See **nss(4)** for more information.

When compiling multithreaded applications, see **Intro(3)**, **MULTITHREADED APPLICATIONS**, for information about the use of the **\_REENTRANT** flag.

Use of the enumeration interfaces **gethostent()** and **gethostent\_r()** is discouraged; enumeration might not be supported for all database sources. The semantics of enumeration are discussed further in **nsswitch.conf(4)**.

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

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