NAME

gethostbyname, gethostbyname_r, gethostbyaddr, gethostbyaddr_r, gethostent, gethostent_r, sethostent, endhostent – get network host entry

SYNOPSIS

cc [*flag* ...] *file* ... –**lnsl** [*library* ...]

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#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);
```

MT-LEVEL

See the subsection "Reentrant Interfaces" in the **DESCRIPTION** section of this page.

DESCRIPTION

These functions are used to obtain entries describing hosts. An entry may come from any of the sources for **hosts** specified in the **/etc/nsswitch.conf** file (see **nsswitch.conf**(4)).

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

gethostbyaddr( ) 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>**. The parameter *addr* must be a pointer to a buffer containing the address. The address is given in a form specific to the address family. See the **NOTES** section below for more information. Also see the **EXAMPLES** section below on how to convert a ''.'' separated Internet IP address notation into the *addr* parameter. The parameter *len* specifies the length of the buffer indicated by *addr*.

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

sethostent( ) 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 may keep allocated resources such as open file descriptors until a subsequent call to **endhostent( )**.

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

endhostent() may be called to indicate that the caller expects to do no further host entry retrieval operations; the system may 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 functions **gethostbyname()**, **gethostbyaddr()**, and **gethostent()** use static storage that is re-used in each call, making these functions unsafe for use in multithreaded applications.

The functions:

gethostbyname_r(),
gethostbyaddr_r(),

and

gethostent_r()

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 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 **RETURN VALUES**). 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 **ERRORS**).

For enumeration in multithreaded applications, the position within the enumeration is a process-wide property shared by all threads. **sethostent()** may 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 below on how to retrieve a ''.'' separated Internet IP address string from the *h_addr_list* field of **struct hostent .**

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

The functions **gethostent( )** and **gethostent_r( )** 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 functions **gethostbyname( )**, **gethostbyaddr( )**, and **gethostent( )** 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 functions **sethostent( )** and **endhostent( )** 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.

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**. 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.

EXAMPLES

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 <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)
{
      u_long addr;
      struct hostent *hp;
      char **p;

      if (argc != 2) {
         (void) printf("usage: %s IP-address\n", argv[0]);
         exit (1);
      }
      if ((int)(addr = inet_addr(argv[1])) == -1) {
         (void) printf("IP-address must be of the form a.b.c.d\n");
         exit (2);
      }

      hp = gethostbyaddr((char *)&addr, sizeof (addr), AF_INET);
      if (hp == NULL) {
         (void) printf("host information for %s not found\n", 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\t%s", inet_ntoa(in), hp−>h_name);
         for (q = hp->h_aliases; *q != 0; q++)
            (void) printf(" %s", *q);
         (void) putchar('\n');
      }
      exit (0);
}
```

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

FILES

/etc/hosts
/etc/netconfig
/etc/nsswitch.conf

SEE ALSO

inet(3N), **netdir**(3N), **hosts**(4), **netconfig**(4), **nsswitch.conf**(4)

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

Programs that use the interfaces described in this manual page cannot be linked statically since the implementations of these functions employ dynamic loading and linking of shared objects at run time.

In order 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; otherwise the 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 may 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 will strive to put the addresses on the same subnet as that of the caller first.

When compiling multithreaded applications, see **Intro**(3), *Notes On Multithread Applications*, for information about the use of the **_REENTRANT** flag.

Use of the enumeration interfaces **gethostent()** and **gethostent_r()** is discouraged; enumeration may 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**(3N), and illustrated in the **EXAMPLES** section above, are helpful in constructing and manipulating addresses in this form.