NAME

getnetbyname, getnetbyname_r, getnetbyaddr, getnetbyaddr_r, getnetent, getnetent_r, setnetent, endnetent – get network entry

SYNOPSIS

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

#include <netdb.h>

struct netent ∗**getnetbyname(const char** ∗*name*);

struct netent ∗**getnetbyname_r(const char** ∗*name*, **struct netent** ∗*result*, **char** ∗*buffer*, **int** *buflen*);

struct netent ∗**getnetbyaddr(long** *net*, **int** *type*);

struct netent ∗**getnetbyaddr_r(long** *net*, **int** *type*, **struct netent** ∗*result*, **char** ∗*buffer*, **int** *buflen*);

struct netent ∗**getnetent(void);**

struct netent ∗**getnetent_r(struct netent** ∗*result*, **char** ∗*buffer*, **int** *buflen*);

int setnetent(int *stayopen*);

int endnetent(void);

MT-LEVEL

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

DESCRIPTION

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

getnetbyname( ) searches for a network entry with the network name specified by the character string parameter *name*.

getnetbyaddr( ) searches for a network entry with the network address specified by *net*. 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 below for more information.

The functions **setnetent( )**, **getnetent( )**, and **endnetent( )** are used to enumerate network entries from the database.

setnetent( ) sets (or resets) the enumeration to the beginning of the set of network entries. This function should be called before the first call to **getnetent( )**. Calls to **getnetbyname( )** and **getnetbyaddr( )** 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 **endnetent( )**.

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

endnetent( ) may be called to indicate that the caller expects to do no further network 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 network entry retrieval functions after calling **endnetent( )**.

Reentrant Interfaces

The functions **getnetbyname( )**, **getnetbyaddr( )**, and **getnetent( )** use static storage that is re-used in each call, making these routines unsafe for use in multithreaded applications.

The functions:

getnetbyname_r( ),
getnetbyaddr_r( ),

and

getnetent_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 netent** structure allocated by the caller. On successful completion, the function returns the network 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 network entry data. All of the pointers within the returned **struct netent** *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 network entry. The parameter *buflen* should give the size in bytes of the buffer indicated by *buffer* .

For enumeration in multithreaded applications, the position within the enumeration is a process-wide property shared by all threads. **setnetent( )** may be used in a multithreaded application but resets the enumeration position for all threads. If multiple threads interleave calls to **getnetent_r( )**, the threads will enumerate disjoint subsets of the network database.

Like their non-reentrant counterparts, **getnetbyname_r( )** and **getnetbyaddr_r( )** leave the enumeration position in an indeterminate state.

RETURN VALUES

Network entries are represented by the **struct netent** structure defined in **<netdb.h>**:

```
struct netent {
        char    *n_name;
        char    **n_aliases;
        int     n_addrtype;
        long    n_net;
};
```

The functions **getnetbyname( )**, **getnetbyname_r( )**, **getnetbyaddr( )**, and **getnetbyaddr_r( )** each return a pointer to a **struct netent** if they successfully locate the requested entry; otherwise they return NULL.

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

The functions **getnetbyname( )**, **getnetbyaddr( )**, and **getnetent( )** 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 **getnetbyname_r( )**, **getnetbyaddr_r( )**, and **getnetent_r( )** is non-NULL, it is always equal to the *result* pointer that was supplied by the caller.

The functions **setnetent( )** and **endnetent( )** return **0** on success.

ERRORS

The reentrant functions **getnetbyname_r( )**, **getnetbyaddr_r( )** and **getnetent_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.

FILES

/etc/networks
/etc/nsswitch.conf

SEE ALSO

inet(3N), **networks**(4), **nsswitch.conf**(4)

WARNINGS

The reentrant interfaces **getnetbyname_r( )**, **getnetbyaddr_r( )**, and **getnetent_r( )** are included in this release on an uncommitted basis only, and are subject to change or removal in future minor releases.

NOTES

The current implementation of these functions only return or accept network numbers for the Internet address family (type **AF_INET).** The functions described in **inet**(3N) may be helpful in constructing and manipulating addresses and network numbers in this form.

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.

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 **getnetent( )** and **getnetent_r( )** is discouraged; enumeration may not be supported for all database sources. The semantics of enumeration are discussed further in **nsswitch.conf**(4).