**NAME**

getnetgrent, getnetgrent_r, setnetgrent, endnetgrent, innetgr – get network group entry

**SYNOPSIS**

**int getnetgrent(char** ∗∗*machinep*, **char** ∗∗*userp*, **char** ∗∗*domainp*);

**int getnetgrent_r(char** ∗∗*machinep*, **char** ∗∗*userp*, **char** ∗∗*domainp*, **char** ∗*buffer*, **int** *buflen*);

**void setnetgrent(const char** ∗*netgroup*);

**void endnetgrent(void);**

**int innetgr(const char** ∗*netgroup*, **const char** ∗*machine*, **const char** ∗*user*, **const char** ∗*domain*);

**MT-LEVEL**

See the **DESCRIPTION** section of this page.

**DESCRIPTION**

These functions are used to test membership in and enumerate members of ‘‘netgroup’’ network groups defined in a system database. Netgroups are sets of (machine,user,domain) triples (see **netgroup**(4)).

These functions consult the source specified for **netgroup** in the **/etc/nsswitch.conf** file (see **nsswitch.conf**(4)).

The function **innetgr( )** returns 1 if there is a netgroup *netgroup* that contains the specified *machine, user, domain* triple as a member; otherwise it returns 0. Any of the supplied pointers *machine, user,* and *domain* may be **NULL,** signifying a "wild card" that matches all values in that position of the triple.

The **innetgr( )** function is safe for use in single-threaded and multi-threaded applications.

The functions **setnetgrent( )**, **getnetgrent( )**, and **endnetgrent( )** are used to enumerate the members of a given network group.

The function **setnetgrent( )** establishes the network group specified in the parameter *netgroup* as the current group whose members are to be enumerated.

Successive calls to the function **getnetgrent( )** will enumerate the members of the group established by calling **setnetgrent( )**; each call returns 1 if it succeeds in obtaining another member of the network group, or 0 if there are no further members of the group.

When calling either **getnetgrent( )** or **getnetgrent_r( ) ,** addresses of the three character pointers are used as arguments; i.e.:

**char** ∗*mp*, ∗*up*, ∗*dp*;

**getnetgrent(**&*mp*, &*up*, &*dp***);**

Upon successful return from **getnetgrent( ),** the pointer *mp* points to a string containing the name of the machine part of the member triple, *up* points to a string containing the user name and *dp* points to a string containing the domain name. If the pointer returned for *mp, up,* or *dp* is **NULL,** it signifies that the element of the netgroup contains wild card specifier in that position of the triple.

The pointers returned by **getnetgrent( )** point into a buffer allocated by **setnetgrent( )** that is re-used by in each call. This space is released when an **endnetgrent( )** call is made, and should not be released by the caller. This implementation is not safe for use in multi-threaded applications.

The function **getnetgrent_r( )** is similar to **getnetgrent( )** but uses a buffer supplied by the caller for the space needed to store the results. The parameter *buffer* should be a pointer to a buffer allocated by the caller and the length of this buffer should be specified by the parameter *buflen.* The buffer must be large enough to hold the data associated with the triple. The **getnetgrent_r( )** function is safe for use both in single-threaded and multi-threaded applications.

The function **endnetgrent( )** frees the space allocated by the previous **setnetgrent( )** call. The equivalent of an **endnetgrent( )** implicitly performed whenever a **setnetgrent( )** call is made to a new network group.

Note that while **setnetgrent( )** and **endnetgrent( )** are safe for use in multi-threaded applications, the effect of each is process-wide. Calling **setnetgrent( )** resets the enumeration position for all threads. If multiple threads interleave calls to **getnetgrent_r( )** each will enumerate a disjoint subset of the netgroup. Thus the effective use of these functions in multi-threaded applications may require coordination by the caller.

## ERRORS

The function **getnetgrent_r( )** will return 0 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 multi-threaded applications.

## FILES

**/etc/nsswitch.conf**

## SEE ALSO

**netgroup**(4) **nsswitch.conf**(4),

## WARNINGS

The function **getnetgrent_r( )** is included in this release on an uncommitted basis only, and is subject to change or removal in future minor releases.

## NOTES

Only the Network Information Services, NIS and NIS+, are supported as sources for the **netgroup** database.

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 multi-threaded applications, see **Intro**(3), *Notes On Multithread Applications*, for information about the use of the **_REENTRANT** flag.