NAME

getrpcbyname, getrpcbyname_r, getrpcbynumber, getrpcbynumber_r, getrpcent, getrpcent_r, setrpcent, endrpcent – get RPC entry

SYNOPSIS

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

#include <rpc/rpcent.h>

struct rpcent *getrpcbyname(const char * *name***);**

struct rpcent *getrpcbyname_r(const char * *name*, **struct rpcent ****result*, **char ****buffer*, **int** *buflen***);**

struct rpcent *getrpcbynumber(const int *number***);**

struct rpcent *getrpcbynumber_r(const int *number*, **struct rpcent ****result*, **char ****buffer*, **int** *buflen***);**

struct rpcent *getrpcent(void);

struct rpcent *getrpcent_r(struct rpcent ****result*, **char ****buffer*, **int *buflen***);**

void setrpcent(const int *stayopen***);**

void endrpcent(void);

MT-LEVEL

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

DESCRIPTION

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

getrpcbyname( ) searches for an entry with the RPC service name specified by the parameter *name*.

getrpcbynumber( ) searches for an entry with the RPC program number *number*.

The functions **setrpcent( )**, **getrpcent( )**, and **endrpcent( )** are used to enumerate RPC entries from the database.

setrpcent( ) sets (or resets) the enumeration to the beginning of the set of RPC entries. This function should be called before the first call to **getrpcent( )**. Calls to **getrpcbyname( )** and **getrpcbynumber( )** 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 **endrpcent( )**.

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

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

Reentrant Interfaces

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

The functions:

getrpcbyname_r( ),
getrpcbynumber_r( ),

and

getrpcent_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 rpcent** structure allocated by the caller. On successful completion, the function returns the RPC 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 RPC entry data. All of the pointers within the returned **struct rpcent** *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 RPC 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. **setrpcent( )** may be used in a multithreaded application but resets the enumeration position for all threads. If multiple threads interleave calls to **getrpcent_r( )**, the threads will enumerate disjoint subsets of the RPC entry database.

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

RETURN VALUES

RPC entries are represented by the **struct rpcent** structure defined in **<rpc/rpcent.h>**:

```
struct rpcent {
        char *r_name;       /* name of this rpc service */
        char **r_aliases;   /* zero-terminated list of
                               alternate names */
        long r_number;      /* rpc program number */
};
```

The functions **getrpcbyname( )**, **getrpcbyname_r( )**, **getrpcbynumber( )**, and **getrpcbynumber_r( )** each return a pointer to a **struct rpcent** if they successfully locate the requested entry; otherwise they return NULL.

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

The functions **getrpcbyname( )**, **getrpcbynumber( )**, and **getrpcent( )** 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 **getrpcbyname_r( )**, **getrpcbynumber_r( )**, and **getrpcent_r( )** is non-NULL, it is always equal to the *result* pointer that was supplied by the caller.

ERRORS

The reentrant functions **getrpcyname_r( )**, **getrpcbynumber_r( )** and **getrpcent_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/rpc
/etc/nsswitch.conf

SEE ALSO

rpcinfo(1M), **rpc**(3N), **nsswitch.conf**(4), **rpc**(4)

WARNINGS

The reentrant interfaces **getrpcbyname_r( )**, **getrpcbynumber_r( )**, and **getrpcent_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.

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