NAME

getservbyname, getservbyname_r, getservbyport, getservbyport_r, getservent, getservent_r, setservent, endservent – get service entry

SYNOPSIS

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

#include <netdb.h>

struct servent ∗**getservbyname(const char** ∗*name*, **const char** ∗*proto*);

struct servent ∗**getservbyname_r(const char** ∗*name*, **const char** ∗*proto*, **struct servent** ∗*result*, **char** ∗*buffer*, **int** *buflen*);

struct servent ∗**getservbyport(int** *port*, **const char** ∗*proto*);

struct servent ∗**getservbyport_r(int** *port*, **const char** ∗*proto*, **struct servent** ∗*result*, **char** ∗*buffer*, **int** *buflen*);

struct servent ∗**getservent(void);**

struct servent ∗**getservent_r(struct servent** ∗*result*, **char** ∗*buffer*, **int** *buflen*);

int setservent(int *stayopen*);

int endservent(void);

MT-LEVEL

See the subsection “Reentrant Interfaces” in the **DESCRIPTION** section of this man page.

DESCRIPTION

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

getservbyname( ) and **getservbyport( )** sequentially search from the beginning of the file until a matching protocol name or port number is found, or until end-of-file is encountered. If a protocol name is also supplied (non-NULL), searches must also match the protocol.

getservbyname( ) searches for an entry with the Internet service name specified by the parameter *name*.

getservbyport( ) searches for an entry with the Internet port number *port*.

The string *proto* is used by both **getservbyname( )** and **getservbyport( )** to restrict the search to entries with the specified protocol. If *proto* is **NULL,** entries with any protocol may be returned.

The functions **setservent( )**, **getservent( )**, and **endservent( )** are used to enumerate entries from the services database.

setservent( ) sets (or resets) the enumeration to the beginning of the set of service entries. This function should be called before the first call to **getservent( )**. Calls to the functions **getservbyname( )** and **getservbyport( )** 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 **endservent( )**.

getservent( ) reads the next line of the file, opening the file if necessary. **getservent( )** opens and rewinds the file. If the *stayopen* flag is non-zero, the net data base will not be closed after each call to **getservent( )** (either directly, or indirectly through one of the other "getserv" calls).

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

endservent( ) closes the file. **endservent( )** may be called to indicate that the caller expects to do no further service 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 service entry retrieval functions after calling **endservent( )**.

Reentrant Interfaces

The functions **getservbyname( )**, **getservbyport( )**, and **getservent( )** use static storage that is re-used in each call, making these functions unsafe for use in multithreaded applications.

The functions:

getservbyname_r( ),
getservbyport_r( ),

and

getservent_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 servent** structure allocated by the caller. On successful completion, the function returns the service 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 service entry data. All of the pointers within the returned **struct servent** *result* point to data stored within this buffer. See the **RETURN VALUES** section of this man page. The buffer must be large enough to hold all of the data associated with the service 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. **setservent( )** may be used in a multithreaded application but resets the enumeration position for all threads. If multiple threads interleave calls to **getservent_r( )**, the threads will enumerate disjoint subsets of the service database.

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

RETURN VALUES

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

```
struct servent {
        char    *s_name;        /* official name of service */
        char    **s_aliases;    /* alias list */
        int     s_port;         /* port service resides at */
        char    *s_proto;       /* protocol to use */
};
```

The members of this structure are:

s_name	The official name of the service.
s_aliases	A zero terminated list of alternate names for the service.
s_port	The port number at which the service resides. Port numbers are returned in network byte order.
s_proto	The name of the protocol to use when contacting the service.

The functions **getservbyname( )**, **getservbyname_r( )**, **getservbyport( )**, and **getservbyport_r( )** each return a pointer to a **struct servent** if they successfully locate the requested entry; otherwise they return **NULL.**

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

The functions **getservbyname( )**, **getservbyport( )**, and **getservent( )** 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 **getservbyname_r( )**, **getservbyport_r( )**, and **getservent_r( )** is non-null, it is always equal to the *result* pointer that was supplied by the caller.

ERRORS

The reentrant functions **getservbyname_r( )**, **getservbyport_r( )** and **getservent_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/services	Internet network services
/etc/netconfig	network configuration file
/etc/nsswitch.conf	configuration file for the name-service switch

SEE ALSO

intro(2), **intro**(3), **netdir**(3N), **netconfig**(4), **nsswitch.conf**(4), **services**(4)

WARNINGS

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

NOTES

The functions that return **struct servent** return the least significant 16-bits of the *s_port* field in *network byte order.* **getservbyport( )** and **getservbyport_r( )** also expect the input parameter *port* in the *network byte order* . See **htons**(3N) for more details on converting between host and network byte orders.

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, **getservbyname( )**, **getservbyname_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 **services:** entry in **nsswitch.conf**(4). Similarly, **getservbyport( )**, **getservbyport_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 **services** in **nsswitch.conf** will be used; otherwise the name-toaddr libraries in that column will be used, and **nsswitch.conf** will not be consulted.

There is no analogue of **getservent( )** and **getservent_r( )** in the netdir functions, so these enumeration functions go straight to the **services** entry in **nsswitch.conf**. Thus enumeration may return results from a different source than that used by **getservbyname( )**, **getservbyname_r( )**, **getservbyport( )**, and **getservbyport_r( )**.

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