



***Rocky Enterprise Linux 9.2 Manual Pages on command 'endspent.3'***

***\$ man endspent.3***

GETSPNAM(3)           Linux Programmer's Manual           GETSPNAM(3)

NAME

getspnam, getspnam\_r, getspent, getspent\_r, setspent, endspent, fget?  
spent, fgetspent\_r, sgetspent, sgetspent\_r, putspent, lckpword, ulckpword  
- get shadow password file entry

SYNOPSIS

```
/* General shadow password file API */  
  
#include <shadow.h>  
  
struct spwd *getspnam(const char *name);  
struct spwd *getspent(void);  
void setspent(void);  
void endspent(void);  
  
struct spwd *fgetspent(FILE *stream);  
struct spwd *sgetspent(const char *s);  
int putspent(const struct spwd *p, FILE *stream);  
int lckpword(void);  
int ulckpword(void);  
  
/* GNU extension */
```

```
#include <shadow.h>

int getspent_r(struct spwd *spbuf,
               char *buf, size_t buflen, struct spwd **spbufp);

int getspnam_r(const char *name, struct spwd *spbuf,
               char *buf, size_t buflen, struct spwd **spbufp);

int fgetspent_r(FILE *stream, struct spwd *spbuf,
                char *buf, size_t buflen, struct spwd **spbufp);

int sgetspent_r(const char *s, struct spwd *spbuf,
                char *buf, size_t buflen, struct spwd **spbufp);
```

Feature Test Macro Requirements for glibc (see `feature_test_macros(7)`):

`getspent_r()`, `getspnam_r()`, `fgetspent_r()`, `sgetspent_r()`:

Since glibc 2.19:

```
_DEFAULT_SOURCE
```

Glibc 2.19 and earlier:

```
_BSD_SOURCE || _SVID_SOURCE
```

## DESCRIPTION

Long ago it was considered safe to have encrypted passwords openly visible in the password file. When computers got faster and people got more security-conscious, this was no longer acceptable. Julianne Frances Haugh implemented the shadow password suite that keeps the encrypted passwords in the shadow password database (e.g., the local shadow password file `/etc/shadow`, NIS, and LDAP), readable only by root.

The functions described below resemble those for the traditional password database (e.g., see `getpwnam(3)` and `getpwent(3)`).

The `getspnam()` function returns a pointer to a structure containing the broken-out fields of the record in the shadow password database that matches the username name.

The `getspent()` function returns a pointer to the next entry in the shadow password database. The position in the input stream is initialized by `setspent()`. When done reading, the program may call `endspent()` so that resources can be deallocated.

The `fgetspent()` function is similar to `getspent()` but uses the supplied

stream instead of the one implicitly opened by `setspent()`.

The `sgetspent()` function parses the supplied string `s` into a struct `spwd`.

The `putspent()` function writes the contents of the supplied struct `spwd *p` as a text line in the shadow password file format to stream. String entries with value `NULL` and numerical entries with value `-1` are written as an empty string.

The `lckpwnf()` function is intended to protect against multiple simultaneous accesses of the shadow password database. It tries to acquire a lock, and returns `0` on success, or `-1` on failure (lock not obtained within `15` seconds). The `ulckpwnf()` function releases the lock again.

Note that there is no protection against direct access of the shadow password file. Only programs that use `lckpwnf()` will notice the lock.

These were the functions that formed the original shadow API. They are widely available.

#### Reentrant versions

Analogous to the reentrant functions for the password database, `glibc` also has reentrant functions for the shadow password database. The `getspnam_r()` function is like `getspnam()` but stores the retrieved shadow password structure in the space pointed to by `spbuf`. This shadow password structure contains pointers to strings, and these strings are stored in the buffer `buf` of size `buflen`. A pointer to the result (in case of success) or `NULL` (in case no entry was found or an error occurred) is stored in `*spbufp`.

The functions `getspent_r()`, `fgetspent_r()`, and `sgetspent_r()` are similarly analogous to their nonreentrant counterparts.

Some non-`glibc` systems also have functions with these names, often with different prototypes.

#### Structure

The shadow password structure is defined in `<shadow.h>` as follows:

```
struct spwd {
    char *sp_namp; /* Login name */
    char *sp_pwdp; /* Encrypted password */
```

```

long sp_lstchg; /* Date of last change
                (measured in days since
                1970-01-01 00:00:00 +0000 (UTC)) */
long sp_min; /* Min # of days between changes */
long sp_max; /* Max # of days between changes */
long sp_warn; /* # of days before password expires
              to warn user to change it */
long sp_inact; /* # of days after password expires
              until account is disabled */
long sp_expire; /* Date when account expires
                (measured in days since
                1970-01-01 00:00:00 +0000 (UTC)) */
unsigned long sp_flag; /* Reserved */
};

```

## RETURN VALUE

The functions that return a pointer return NULL if no more entries are available or if an error occurs during processing. The functions which have `int` as the return value return 0 for success and -1 for failure, with `errno` set to indicate the cause of the error.

For the nonreentrant functions, the return value may point to static area, and may be overwritten by subsequent calls to these functions. The reentrant functions return zero on success. In case of error, an error number is returned.

## ERRORS

**EACCES** The caller does not have permission to access the `shadow` password file.

**ERANGE** Supplied buffer is too small.

## FILES

`/etc/shadow`

local shadow password database file

`/etc/.pwd.lock`

lock file

The include file `<paths.h>` defines the constant `_PATH_SHADOW` to the

pathname of the shadow password file.

## ATTRIBUTES

For an explanation of the terms used in this section, see at?

tributes(7).

??

?Interface ? Attribute ? Value ?

??

?getspnam() ? Thread safety ? MT-Unsafe race:getspnam locale ?

??

?getspent() ? Thread safety ? MT-Unsafe race:spent ?

? ? ? race:spentbuf locale ?

??

?setspent(), ? Thread safety ? MT-Unsafe race:setspent locale ?

?endspent(), ? ? ?

?getspent\_r() ? ? ?

??

?fgetspent() ? Thread safety ? MT-Unsafe race:fgetspent ?

??

?sgetspent() ? Thread safety ? MT-Unsafe race:sgetspent ?

??

?putspent(), ? Thread safety ? MT-Safe locale ?

?getspnam\_r(), ? ? ?

?sgetspent\_r() ? ? ?

??

?lckpwn(), ? Thread safety ? MT-Safe ?

?ulckpwn(), ? ? ?

?fgetspent\_r() ? ? ?

??

In the above table, getspent in race:spent signifies that if any of the functions setspent(), getspent(), getspent\_r(), or endspent() are used in parallel in different threads of a program, then data races could occur.

The shadow password database and its associated API are not specified in POSIX.1. However, many other systems provide a similar API.

#### SEE ALSO

`getgrnam(3)`, `getpwnam(3)`, `getpwnam_r(3)`, `shadow(5)`

#### COLOPHON

This page is part of release 5.10 of the Linux man-pages project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at <https://www.kernel.org/doc/man-pages/>.

GNU

2017-09-15

GETSPNAM(3)