NAME

getpwnam, getpwnam_r, getpwent, getpwent_r, getpwuid, getpwuid_r, setpwent, endpwent, fgetpwent, fgetpwent_r – get password entry

SYNOPSIS

```
#include <pwd.h>

struct passwd *getpwnam(const char *name);

struct passwd *getpwnam_r(const char *name, struct passwd *result,
        char *buffer, int buflen);

struct passwd *getpwent(void);

struct passwd *getpwent_r(struct passwd *result, char *buffer, int buflen);

struct passwd *getpwuid(uid_t uid);

struct passwd *getpwuid_r(uid_t uid, struct passwd * result, char *buffer, int buflen);

void setpwent(void);

void endpwent(void);

struct passwd *fgetpwent(FILE *f);

struct passwd *fgetpwent_r(FILE *f, struct passwd *result, char *buffer, int buflen);
```

POSIX

```
cc [ flag... ] file ... –D_POSIX_PTHREAD_SEMANTICS [ library... ]

int getpwnam_r(const char *name,struct passwd *pwd, char *buffer, size_t bufsize
        struct passwd **result);

int getpwuid_r(uid_t uid, struct passwd *pwd, char *buffer, size_t bufsize
        struct passwd **result);
```

MT-LEVEL

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

DESCRIPTION

These functions are used to obtain password entries. Entries can come from any of the sources for **passwd** specified in the **/etc/nsswitch.conf** file (see **nsswitch.conf**(4)).

getpwnam( ) searches for a password entry with the login name specified by the character string parameter *name*.

getpwuid( ) searches for a password entry with the (numeric) user id specified by the parameter *uid*.

The functions **setpwent( )**, **getpwent( )**, and **endpwent( )** are used to enumerate password entries from the database. **setpwent( )** sets (or resets) the enumeration to the beginning of the set of password entries. This function should be called before the first call to **getpwent( )**. Calls to **getpwnam( )** and **getpwuid( )** leave the enumeration position in an indeterminate state. Successive calls to **getpwent( )** return either successive entries or NULL, indicating the end of the enumeration.

endpwent( ) may be called to indicate that the caller expects to do no further password retrieval operations; the system may then close the password file, deallocate resources it was using, and so forth. It is still allowed, but possibly less efficient, for the process to call more password functions after calling **endpwent( )**.

fgetpwent( ), unlike the other functions above, does not use **nsswitch.conf**; it reads and parses the next line from the stream *f*, which is assumed to have the format of the **passwd** file. See **passwd**(4).

Reentrant Interfaces

The functions **getpwnam( )**, **getpwuid( )**, **getpwent( )**, and **fgetpwent( )** use static storage that is re-used in each call, making these routines unsafe for use in multithreaded applications.

The parallel functions **getpwnam_r( )**, **getpwuid_r( )**, **getpwent_r( )**, and **fgetpwent_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* (or *pwd* for the POSIX versions) must be a pointer to a **struct passwd** structure allocated by the caller. On successful completion, the function returns the password entry in this structure. The parameter *buffer* is a pointer to a buffer supplied by the caller, used as storage space for the password data. All of the pointers within the returned **struct passwd** *result* (or *grp*) point to data stored within this buffer; see **RETURN VALUES**. The buffer must be large enough to hold all the data associated with the password entry. The parameter *buflen* (or *bufsize* for the POSIX versions) should give the size in bytes of *buffer*. The POSIX versions place a pointer to the modified *grp* structure in the *result* parameter, instead of returning a pointer to this structure.

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

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

RETURN VALUES

Password entries are represented by the **struct passwd** structure defined in **<pwd.h>**:

```
struct passwd {
   char *pw_name;    /* user's login name */
   char *pw_passwd;  /* no longer used */
   uid_t pw_uid;       /* user's uid */
   gid_t pw_gid;       /* user's gid */
   char *pw_age;     /* not used */
   char *pw_comment; /* not used */
   char *pw_gecos;   /* typically user's full name */
   char *pw_dir;     /* user's home dir */
   char *pw_shell;   /* user's login shell */
};
```

The functions **getpwnam( )**, **getpwnam_r( )**, **getpwuid( )**, and **getpwuid_r( )** each return a pointer to a **struct passwd** if they successfully locate the requested entry; otherwise they return NULL. The POSIX functions **getpwnam_r( )** and **getpwuid_r( )** return zero upon success, or the error number in case of failure.

The functions **getpwent( )**, **getpwent_r( )**, **fgetpwent( )**, and **fgetpwent_r( )** each return a pointer to a **struct passwd** if they successfully enumerate an entry; otherwise they return NULL, indicating the end of the enumeration.

The functions **getpwnam( )**, **getpwuid( )**, **getpwent( )**, and **fgetpwent( )** 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 **getpwnam_r( )**, **getpwuid_r( )**, **getpwent_r( )**, and **fgetpwent_r( )** is non-NULL, it is always equal to the *result* pointer that was supplied by the caller.

ERRORS

The reentrant functions **getpwnam_r( )**, **getpwuid_r( )**, **getpwent_r( )**, and **fgetpwent_r( )** will return NULL and set **errno** to **ERANGE** (or in the case of POSIX functions **getpwnam_r( )** and **getpwuid_r( )** return the **ERANGE** error) 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/passwd
/etc/shadow
/etc/nsswitch.conf

SEE ALSO

nispasswd(1), **passwd**(1), **yppasswd**(1), **Intro**(2), **Intro**(3), **cuserid**(3S), **getgrnam**(3C), **getlogin**(3C), **getspnam**(3C), **nsswitch.conf**(4), **passwd**(4), **shadow**(4)

NOTES

The **pw_passwd** field in the **passwd** structure should not be used as the encrypted password for the user; use **getspnam( )** or **getspnam_r( )** instead. See **getspnam**(3C).

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 **getpwent( )** and **getpwent_r( )** is discouraged; enumeration is supported for the passwd file, NIS, and NIS+, but in general is not efficient and may not be supported for all database sources. The semantics of enumeration are discussed further in **nsswitch.conf**(4).

Previous releases allowed the use of '+' and '-' entries in **/etc/passwd** to selectively include and exclude NIS entries. The primary usage of these '+/-' entries is superseded by the name service switch, so the '+/-' form may not be supported in future releases.

If required, the '+/-' functionality can still be obtained for NIS by specifying **compat** as the source for **passwd**.

If the '+/-' functionality is required in conjunction with NIS+, specify both **compat** as the source for **passwd** and **nisplus** as the source for the pseudo-database **passwd_compat**. See **passwd**(4), **shadow**(4), and **nsswitch.conf**(4) for details.

If the '+/-' is used, both **/etc/shadow** and **/etc/passwd** should have the same '+' and '-' entries to ensure consistency between the password and shadow databases.

If a password entry from any of the sources contains an empty *uid* or *gid* field, that entry will be ignored by the files, NIS, and NIS+ name service switch backends. This will cause the user to appear unknown to the system.

If a password entry contains an empty *gecos*, *home directory*, or *shell* field, **getpwnam( )** and **getpwnam_r( )** return a pointer to a null string in the respective field of the **passwd** structure.

If the shell field is empty, **login**(1) automatically assigns the default shell. See **login**(1).

Reentrant interfaces **getpwnam_r( )**, **getpwent_r( )**, **getpwuid_r( )**, and **fgetpwent_r( )** are as specified in POSIX 1003.1c Draft #10.