NAME

getpwent, **getpwent_r**, **getpwnam**, **getpwnam_r**, **getpwuid**, **getpwuid_r**, **setpassent**, **setpwent**, **endpwent** - password database operations

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

```
#include <sys/types.h>
#include <pwd.h>

struct passwd *
getpwent(void);

int
getpwent_r(struct passwd *pwd, char *buffer, size_t bufsize, struct passwd **result);

struct passwd *
getpwnam(const char *login);

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

struct passwd *
getpwuid(uid_t uid);

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

int
setpassent(int stayopen);

void
setpwent(void);

void
endpwent(void);
```

DESCRIPTION

These functions operate on the password database file which is described in passwd(5). Each entry in the database is defined by the structure *passwd* found in the include file <*pwd.h*>:

```
struct passwd {
        char *pw_name;     /* user name */
        char *pw_passwd;  /* encrypted password */
        uid_t       pw_uid;         /* user uid */
        gid_t       pw_gid;         /* user gid */
        time_t      pw_change;    /* password change time */
        char *pw_class;      /* user access class */
        char *pw_gecos;     /* Honeywell login info */
        char *pw_dir;  /* home directory */
        char *pw_shell;      /* default shell */
        time_t      pw_expire;      /* account expiration */
        int    pw_fields;       /* internal: fields filled in */
};
```

The functions **getpwnam**() and **getpwuid**() search the password database for the given login name or user uid, respectively, always returning the first one encountered.

The **getpwent**() function sequentially reads the password database and is intended for programs that wish to process the complete list of users.

The functions **getpwent_r**(), **getpwnam_r**(), and **getpwuid_r**() are thread-safe versions of **getpwent**(), **getpwnam**(), and **getpwuid**(), respectively. The caller must provide storage for the results of the search in the *pwd*, *buffer*, *bufsize*, and *result* arguments. When these functions are successful, the *pwd* argument will be filled-in, and a pointer to that argument will be stored in *result*. If an entry is not found or an error occurs, *result* will be set to NULL.

The **setpassent**() function accomplishes two purposes. First, it causes **getpwent**() to ‘‘rewind’’ to the beginning of the database. Additionally, if *stayopen* is non-zero, file descriptors are left open, significantly speeding up subsequent accesses for all of the routines. (This latter functionality is unnecessary for **getpwent**() as it does not close its file descriptors by default.)

It is dangerous for long-running programs to keep the file descriptors open as the database will become out of date if it is updated while the program is running.

The **setpwent**() function is identical to **setpassent**() with an argument of zero.

The **endpwent**() function closes any open files.

These routines have been written to ‘‘shadow’’ the password file, e.g. allow only certain programs to have access to the encrypted password. If the process which calls them has an effective uid of 0, the encrypted password will be returned, otherwise, the password field of the returned structure will point to the string ‘*’.

RETURN VALUES

The functions **getpwent**(), **getpwnam**(), and **getpwuid**() return a valid pointer to a passwd structure on success or NULL if the entry is not found or if an error occurs. If an error does occur, *errno* will be set. Note that programs must explicitly set *errno* to zero before calling any of these functions if they need to distinguish between a non-existent entry and an error. The functions **getpwent_r**(), **getpwnam_r**(), and **getpwuid_r**() return 0 if no error occurred, or an error number to indicate failure. It is not an error if a matching entry is not found. (Thus, if *result* is NULL and the return value is 0, no matching entry exists.)

The **setpassent**() function returns 0 on failure and 1 on success. The **endpwent**() and **setpwent**() functions have no return value.

FILES

/etc/pwd.db	The insecure password database file
/etc/spwd.db	The secure password database file
/etc/master.passwd	The current password file
/etc/passwd	A Version 7 format password file

COMPATIBILITY

The historic function setpwfile(3), which allowed the specification of alternate password databases, has been deprecated and is no longer available.

ERRORS

These routines may fail for any of the errors specified in open(2), dbopen(3), socket(2), and connect(2), in addition to the following:

[ERANGE] The buffer specified by the *buffer* and *bufsize* arguments was insufficiently sized to store the result. The caller should retry with a larger buffer.

SEE ALSO

getlogin(2), getgrent(3), nsswitch.conf(5), passwd(5), pwd_mkdb(8), vipw(8), yp(8)

STANDARDS

The **getpwent**(), **getpwnam**(), **getpwnam_r**(), **getpwuid**(), **getpwuid_r**(), **setpwent**(), and **endpwent**() functions conform to ISO/IEC 9945-1:1996 (‘‘POSIX.1’’).

HISTORY

The **getpwent**(), **getpwnam**(), **getpwuid**(), **setpwent**(), and **endpwent**() functions appeared in Version 7 AT&T UNIX. The **setpassent**() function appeared in 4.3BSD-Reno. The **getpwent_r**(), **getpwnam_r**(), and **getpwuid_r**() functions appeared in FreeBSD 5.1.

BUGS

The functions **getpwent**(), **getpwnam**(), and **getpwuid**(), leave their results in an internal static object and return a pointer to that object. Subsequent calls to the same function will modify the same object.

The functions **getpwent**(), **getpwent_r**(), **endpwent**(), **setpassent**(), and **setpwent**() are fairly useless in a networked environment and should be avoided, if possible. The **getpwent**() and **getpwent_r**() functions make no attempt to suppress duplicate information if multiple sources are specified in nsswitch.conf(5).