NAME

getpwnam, getpwnam_r, getpwuid, getpwuid_r − get password file entry

LIBRARY

Standard C library (*libc*, *−lc*)

SYNOPSIS

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

struct passwd *getpwnam(const char *name);
struct passwd *getpwuid(uid_t uid);

int getpwnam_r(const char *restrict name, struct passwd *restrict pwd,
        char buf[restrict .buflen], size_t buflen,
        struct passwd **restrict result);
int getpwuid_r(uid_t uid, struct passwd *restrict pwd,
        char buf[restrict .buflen], size_t buflen,
        struct passwd **restrict result);
```

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

getpwnam_r(), **getpwuid_r**():
_POSIX_C_SOURCE
|| /* glibc <= 2.19: */ _BSD_SOURCE || _SVID_SOURCE

DESCRIPTION

The **getpwnam**() function returns a pointer to a structure containing the broken-out fields of the record in the password database (e.g., the local password file */etc/passwd*, NIS, and LDAP) that matches the username *name*.

The **getpwuid**() function returns a pointer to a structure containing the broken-out fields of the record in the password database that matches the user ID *uid*.

The *passwd* structure is defined in <*pwd.h*> as follows:

```
struct passwd {
    char   *pw_name;       /* username */
    char   *pw_passwd;     /* user password */
    uid_t   pw_uid;        /* user ID */
    gid_t   pw_gid;        /* group ID */
    char   *pw_gecos;      /* user information */
    char   *pw_dir;        /* home directory */
    char   *pw_shell;      /* shell program */
};
```

See **passwd**(5) for more information about these fields.

The **getpwnam_r**() and **getpwuid_r**() functions obtain the same information as **getpwnam**() and **getpwuid**(), but store the retrieved *passwd* structure in the space pointed to by *pwd*. The string fields pointed to by the members of the *passwd* structure 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 **result*.

The call

```
sysconf(_SC_GETPW_R_SIZE_MAX)
```

returns either −1, without changing *errno*, or an initial suggested size for *buf*. (If this size is too small, the call fails with **ERANGE**, in which case the caller can retry with a larger buffer.)

RETURN VALUE

The **getpwnam**() and **getpwuid**() functions return a pointer to a *passwd* structure, or NULL if the matching entry is not found or an error occurs. If an error occurs, *errno* is set to indicate the error. If one wants to check *errno* after the call, it should be set to zero before the call.

The return value may point to a static area, and may be overwritten by subsequent calls to **getpwent**(3), **getpwnam**(), or **getpwuid**(). (Do not pass the returned pointer to**fr ee**(3).)

On success, **getpwnam_r**() and **getpwuid_r**() return zero, and set*r esult* to *pwd*. If no matching password record was found, these functions return 0 and store NULL in*r esult*. In case of error, an error number is returned, and NULL is stored in*r esult*.

ERRORS

0 or **ENOENT** or **ESRCH** or **EBADF** or **EPERM** or ...
The given *name* or *uid* was not found.

EINTR
A signal was caught; see **signal**(7).

EIO I/O error.

EMFILE
The per-process limit on the number of open file descriptors has been reached.

ENFILE
The system-wide limit on the total number of open files has been reached.

ENOMEM
Insufficient memory to allocate *passwd* structure.

ERANGE
Insufficient buffer space supplied.

FILES

/etc/passwd
local password database file

ATTRIBUTES

For an explanation of the terms used in this section, see **attributes**(7).

Interface	Attribute	Value
getpwnam()	Thread safety	MT-Unsafe race:pwnam locale
getpwuid()	Thread safety	MT-Unsafe race:pwuid locale
getpwnam_r(), **getpwuid_r**()	Thread safety	MT-Safe locale

STANDARDS

POSIX.1-2001, POSIX.1-2008, SVr4, 4.3BSD. The *pw_gecos* field is not specified in POSIX, but is present on most implementations.

NOTES

The formulation given above under "RETURN VALUE" is from POSIX.1-2001. It does not call "not found" an error, and hence does not specify what value *errno* might have in this situation. But that makes it impossible to recognize errors. One might argue that according to POSIX *errno* should be left unchanged if an entry is not found. Experiments on various UNIX-like systems show that lots of different values occur in this situation: 0, ENOENT, EBADF, ESRCH, EWOULDBLOCK, EPERM, and probably others.

The *pw_dir* field contains the name of the initial working directory of the user. Login programs use the value of this field to initialize the **HOME** environment variable for the login shell. An application that wants to determine its user's home directory should inspect the value of **HOME** (rather than the value *getpwuid(getuid())−>pw_dir*) since this allows the user to modify their notion of "the home directory" during a login session. To determine the (initial) home directory of another user, it is necessary to use *getpwnam("username")−>pw_dir* or similar.

EXAMPLES

The program below demonstrates the use of **getpwnam_r**() to find the full username and user ID for the username supplied as a command-line argument.

```
#include <errno.h>
#include <pwd.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int
main(int argc, char *argv[])
{
    struct passwd pwd;
    struct passwd *result;
    char *buf;
    long bufsize;
    int s;

    if (argc != 2) {
        fprintf(stderr, "Usage: %s username\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    bufsize = sysconf(_SC_GETPW_R_SIZE_MAX);
    if (bufsize == -1)          /* Value was indeterminate */
        bufsize = 16384;        /* Should be more than enough */

    buf = malloc(bufsize);
    if (buf == NULL) {
        perror("malloc");
        exit(EXIT_FAILURE);
    }

    s = getpwnam_r(argv[1], &pwd, buf, bufsize, &result);
    if (result == NULL) {
        if (s == 0)
            printf("Not found\n");
        else {
            errno = s;
            perror("getpwnam_r");
        }
        exit(EXIT_FAILURE);
    }

    printf("Name: %s; UID: %jd\n", pwd.pw_gecos,
           (intmax_t) pwd.pw_uid);
    exit(EXIT_SUCCESS);
}
```

SEE ALSO

endpwent(3), **fgetpwent**(3), **getgrnam**(3), **getpw**(3), **getpwent**(3), **getspnam**(3), **putpwent**(3), **setpwent**(3), **passwd**(5)