
*Full credit is given to the above companies including the OS that this PDF file was generated!*

## *Red Hat Enterprise Linux Release 9.2 Manual Pages on 'getlogin.3p' command*

***$ man getlogin.3p***

GETLOGIN(3P) POSIX Programmer's Manual GETLOGIN(3P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

getlogin, getlogin_r ? get login name

SYNOPSIS

```
#include <unistd.h>
char *getlogin(void);
int getlogin_r(char *name, size_t namesize);
```

DESCRIPTION

The getlogin() function shall return a pointer to a string containing the user name associated by the login activity with the controlling terminal of the current process. If getlogin() returns a non-null pointer, then that pointer points to the name that the user logged in under, even if there are several login names with the same user ID.

The getlogin() function need not be thread-safe.

The getlogin_r() function shall put the name associated by the login activity with the controlling terminal of the current process in the character array pointed to by name. The array is namesize characters long and should have space for the name and the terminating null char?

acter. The maximum size of the login name is {LOGIN_NAME_MAX}.

If getlogin_r() is successful, name points to the name the user used at login, even if there are several login names with the same user ID.

The getlogin() and getlogin_r() functions may make use of file descrip? tors 0, 1, and 2 to find the controlling terminal of the current process, examining each in turn until the terminal is found. If in this case none of these three file descriptors is open to the controlling terminal, these functions may fail. The method used to find the termi? nal associated with a file descriptor may depend on the file descriptor being open to the actual terminal device, not /dev/tty.

RETURN VALUE

Upon successful completion, getlogin() shall return a pointer to the login name or a null pointer if the user's login name cannot be found. Otherwise, it shall return a null pointer and set errno to indicate the error.

The application shall not modify the string returned. The returned pointer might be invalidated or the string content might be overwritten by a subsequent call to getlogin(). The returned pointer and the string content might also be invalidated if the calling thread is ter? minated.

If successful, the getlogin_r() function shall return zero; otherwise, an error number shall be returned to indicate the error.

ERRORS

These functions may fail if:

EMFILE All file descriptors available to the process are currently open.

ENFILE The maximum allowable number of files is currently open in the system.

ENOTTY None of the file descriptors 0, 1, or 2 is open to the control? ling terminal of the current process.

ENXIO The calling process has no controlling terminal.

The getlogin_r() function may fail if:

ERANGE The value of namesize is smaller than the length of the string

to be returned including the terminating null character.

The following sections are informative.

## EXAMPLES

### Getting the User Login Name S

The following example calls the getlogin() function to obtain the name of the user associated with the calling process, and passes this infor? mation to the getpwnam() function to get the associated user database information.

```
#include <unistd.h>
#include <sys/types.h>
#include <pwd.h>
#include <stdio.h>
...
char *lgn;
struct passwd *pw;
...
if ((lgn = getlogin()) == NULL || (pw = getpwnam(lgn)) == NULL) {
    fprintf(stderr, "Get of user information failed.\n"); exit(1);
    }
```

## APPLICATION USAGE

Three names associated with the current process can be determined: get? pwuid(geteuid()) shall return the name associated with the effective user ID of the process; getlogin() shall return the name associated with the current login activity; and getpwuid(getuid()) shall return the name associated with the real user ID of the process.

The getlogin_r() function is thread-safe and returns values in a user-supplied buffer instead of possibly using a static data area that may be overwritten by each call.

## RATIONALE

The getlogin() function returns a pointer to the user's login name. The same user ID may be shared by several login names. If it is desired to get the user database entry that is used during login, the result of getlogin() should be used to provide the argument to the getpwnam()

function. (This might be used to determine the user's login shell, par?
ticularly where a single user has multiple login shells with distinct
login names, but the same user ID.)

The information provided by the cuserid() function, which was origi?
nally defined in the POSIX.1?1988 standard and subsequently removed,
can be obtained by the following:

```
getpwuid(geteuid())
```

while the information provided by historical implementations of
cuserid() can be obtained by:

```
getpwuid(getuid())
```

The thread-safe version of this function places the user name in a
user-supplied buffer and returns a non-zero value if it fails. The non-
thread-safe version may return the name in a static data area that may
be overwritten by each call.

## FUTURE DIRECTIONS

None.

## SEE ALSO

getpwnam(), getpwuid(), geteuid(), getuid()

The Base Definitions volume of POSIX.1?2017, <limits.h>, <unistd.h>

## COPYRIGHT