

NAME

getlogin, getlogin_r, cuserid - get username

SYNOPSIS

```
#include <unistd.h>

char *getlogin(void);
int getlogin_r(char *buf, size_t bufsize);

#include <stdio.h>

char *cuserid(char *string);
```

Feature Test Macro Requirements for glibc (see [feature_test_macros\(7\)](#)):

```
getlogin_r(): _REENTRANT || _POSIX_C_SOURCE >= 199506L
cuserid(): _XOPEN_SOURCE
```

DESCRIPTION

getlogin() returns a pointer to a string containing the name of the user logged in on the controlling terminal of the process, or a null pointer if this information cannot be determined. The string is statically allocated and might be overwritten on subsequent calls to this function or to **cuserid()**.

getlogin_r() returns this same username in the array *buf* of size *bufsize*.

cuserid() returns a pointer to a string containing a username associated with the effective user ID of the process. If *string* is not a null pointer, it should be an array that can hold at least **L_cuserid** characters; the string is returned in this array. Otherwise, a pointer to a string in a static area is returned. This string is statically allocated and might be overwritten on subsequent calls to this function or to **getlogin()**.

The macro **L_cuserid** is an integer constant that indicates how long an array you might need to store a username. **L_cuserid** is declared in *<stdio.h>*.

These functions let your program identify positively the user who is running (**cuserid()**) or the user who logged in this session (**getlogin()**). (These can differ when set-user-ID programs are involved.)

For most purposes, it is more useful to use the environment variable **LOGNAME** to find out who the user is. This is more flexible precisely because the user can set **LOGNAME** arbitrarily.

RETURN VALUE

getlogin() returns a pointer to the username when successful, and NULL on failure, with *errno* set to indicate the cause of the error. **getlogin_r()** returns 0 when successful, and nonzero on failure.

ERRORS

POSIX specifies

EMFILE

The calling process already has the maximum allowed number of open files.

ENFILE

The system already has the maximum allowed number of open files.

ENXIO

The calling process has no controlling terminal.

ERANGE

(*getlogin_r*) The length of the username, including the terminating null byte (0), is larger than *bufsize*.

Linux/glibc also has

ENOENT

There was no corresponding entry in the utmp-file.

ENOMEM

Insufficient memory to allocate passwd structure.

ENOTTY

Standard input didn't refer to a terminal. (See BUGS.)

FILES

/etc/passwd

password database file

/var/run/utmp

(traditionally */etc/utmp*; some libc versions used */var/adm/utmp*)

ATTRIBUTES**Multithreading (see pthreads(7))**

The **getlogin()** function is not thread-safe.

The **getlogin_r()** function is thread-safe.

The **cuserid()** function is thread-safe with exceptions. It is not thread-safe if called with a NULL parameter.

CONFORMING TO

getlogin() and **getlogin_r()** specified in POSIX.1-2001.

System V has a **cuserid()** function which uses the real user ID rather than the effective user ID. The **cuserid()** function was included in the 1988 version of POSIX, but removed from the 1990 version. It was present in SUSv2, but removed in POSIX.1-2001.

OpenBSD has **getlogin()** and **setlogin()**, and a username associated with a session, even if it has no controlling terminal.

BUGS

Unfortunately, it is often rather easy to fool **getlogin()**. Sometimes it does not work at all, because some program messed up the utmp file. Often, it gives only the first 8 characters of the login name. The user currently logged in on the controlling terminal of our program need not be the user who started it. Avoid **getlogin()** for security-related purposes.

Note that glibc does not follow the POSIX specification and uses *stdin* instead of */dev/tty*. A bug. (Other recent systems, like SunOS 5.8 and HP-UX 11.11 and FreeBSD 4.8 all return the login name also when *stdin* is redirected.)

Nobody knows precisely what **cuserid()** does; avoid it in portable programs. Or avoid it altogether: use *getpwuid(geteuid())* instead, if that is what you meant. **Do not use cuserid()**.

SEE ALSO

[geteuid\(2\)](#), [getuid\(2\)](#), [utmp\(5\)](#)

COLOPHON

This page is part of release 3.74 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 <http://www.kernel.org/doc/man-pages/>.