NAME

getlogin, **getlogin_r**, **setlogin** - get/set login name

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

```
#include <unistd.h>

char *
getlogin(void);

#include <sys/param.h>

int
getlogin_r(char *name, size_t len);

int
setlogin(const char *name);
```

DESCRIPTION

The **getlogin**() routine returns the login name of the user associated with the current session, as previously set by **setlogin**(). The name is normally associated with a login shell at the time a session is created, and is inherited by all processes descended from the login shell. (This is true even if some of those processes assume another user ID, for example when su(1) is used).

The **getlogin_r**() function provides the same service as **getlogin**() except the caller must provide the buffer *name* with length *len* bytes to hold the result. The buffer should be at least MAXLOGNAME bytes in length.

The **setlogin**() system call sets the login name of the user associated with the current session to *name*. This system call is restricted to the super-user, and is normally used only when a new session is being created on behalf of the named user (for example, at login time, or when a remote shell is invoked).

NOTE: There is only one login name per session.

It is *CRITICALLY* important to ensure that **setlogin**() is only ever called after the process has taken adequate steps to ensure that it is detached from its parent's session. Making a **setsid**() system call is the *ONLY* way to do this. The daemon(3) function calls **setsid**() which is an ideal way of detaching from a controlling terminal and forking into the background.

In particular, doing a **ioctl**(*ttyfd*, *TIOCNOTTY*, ...) or **setpgrp**(...) is *NOT* sufficient.

Once a parent process does a **setsid**() system call, it is acceptable for some child of that process to then do a **setlogin**() even though it is not the session leader, but beware that ALL processes in the session will change their login name at the same time, even the parent.

This is not the same as the traditional UNIX behavior of inheriting privilege.

Since the **setlogin**() system call is restricted to the super-user, it is assumed that (like all other privileged programs) the programmer has taken adequate precautions to prevent security violations.

RETURN VALUES

If a call to **getlogin**() succeeds, it returns a pointer to a null-terminated string in a static buffer, or NULL if the name has not been set. The **getlogin_r**() function returns zero if successful, or the error number upon failure.

The **setlogin**() function returns the value 0 if successful; otherwise the value -1 is returned and the global variable *errno* is set to indicate the error.

ERRORS

The following errors may be returned by these calls:

[EFAULT]	The *name* argument gave an invalid address.
[EINVAL]	The *name* argument pointed to a string that was too long. Login names are limited to MAXLOGNAME (from <*sys/param.h*>) characters, currently 33 including null.
[EPERM]	The caller tried to set the login name and was not the super-user.
[ERANGE]	The size of the buffer is smaller than the result to be returned.

SEE ALSO

setsid(2), daemon(3)

STANDARDS

The **getlogin**() system call and the **getlogin_r**() function conform to ISO/IEC 9945-1:1996 ("POSIX.1").

HISTORY

The **getlogin**() system call first appeared in 4.4BSD. The return value of **getlogin_r**() was changed from

earlier versions of FreeBSD to be conformant with ISO/IEC 9945-1:1996 ("POSIX.1").

BUGS

In earlier versions of the system, **getlogin**() failed unless the process was associated with a login terminal. The current implementation (using **setlogin**()) allows getlogin to succeed even when the process has no controlling terminal. In earlier versions of the system, the value returned by **getlogin**() could not be trusted without checking the user ID. Portable programs should probably still make this check.