NAME

setuid − set user identity

SYNOPSIS

#include <sys/types.h>
#include <unistd.h>

int setuid(uid_t *uid***);**

DESCRIPTION

setuid() sets the effective user ID of the calling process. If the calling process is privileged (more precisely: if the process has the **CAP_SETUID** capability in its user namespace), the real UID and saved set-user-ID are also set.

Under Linux, **setuid**() is implemented like the POSIX version with the **_POSIX_SAVED_IDS** feature. This allows a set-user-ID (other than root) program to drop all of its user privileges, do some un-privileged work, and then reengage the original effective user ID in a secure manner.

If the user is root or the program is set-user-ID-root, special care must be taken: **setuid**() checks the effective user ID of the caller and if it is the superuser, all process-related user ID's are set to *uid*. After this has occurred, it is impossible for the program to regain root privileges.

Thus, a set-user-ID-root program wishing to temporarily drop root privileges, assume the identity of an unprivileged user, and then regain root privileges afterward cannot use **setuid**(). You can accomplish this with **seteuid**(2).

RETURN VALUE

On success, zero is returned. On error, −1 is returned, and *errno* is set appropriately.

Note: there are cases where **setuid**() can fail even when the caller is UID 0; it is a grave security error to omit checking for a failure return from **setuid**().

ERRORS

EAGAIN
The call would change the caller's real UID (i.e., *uid* does not match the caller's real UID), but there was a temporary failure allocating the necessary kernel data structures.

EAGAIN
uid does not match the real user ID of the caller and this call would bring the number of processes belonging to the real user ID *uid* over the caller's **RLIMIT_NPROC** resource limit. Since Linux 3.1, this error case no longer occurs (but robust applications should check for this error); see the description of **EAGAIN** in **execve**(2).

EINVAL
The user ID specified in *uid* is not valid in this user namespace.

EPERM
The user is not privileged (Linux: does not have the **CAP_SETUID** capability in its user namespace) and *uid* does not match the real UID or saved set-user-ID of the calling process.

CONFORMING TO

POSIX.1-2001, POSIX.1-2008, SVr4. Not quite compatible with the 4.4BSD call, which sets all of the real, saved, and effective user IDs.

NOTES

Linux has the concept of the filesystem user ID, normally equal to the effective user ID. The **setuid**() call also sets the filesystem user ID of the calling process. See **setfsuid**(2).

If *uid* is different from the old effective UID, the process will be forbidden from leaving core dumps.

The original Linux **setuid**() system call supported only 16-bit user IDs. Subsequently, Linux 2.4 added **setuid32**() supporting 32-bit IDs. The glibc **setuid**() wrapper function transparently deals with the variation across kernel versions.

C library/kernel differences

At the kernel level, user IDs and group IDs are a per-thread attribute. However, POSIX requires that all threads in a process share the same credentials. The NPTL threading implementation handles the POSIX requirements by providing wrapper functions for the various system calls that change process UIDs and GIDs. These wrapper functions (including the one for **setuid**()) employ a signal-based technique to ensure that when one thread changes credentials, all of the other threads in the process also change their credentials. For details, see **nptl**(7).

SEE ALSO

getuid(2), **seteuid**(2), **setfsuid**(2), **setreuid**(2), **capabilities**(7), **credentials**(7), **user_namespaces**(7)

COLOPHON

This page is part of release 5.02 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 https://www.kernel.org/doc/man−pages/.