## NAME

setfsgid − set group identity used for filesystem checks

## LIBRARY

Standard C library (*libc*, *−lc*)

## SYNOPSIS

**#include <sys/fsuid.h>**

**int setfsgid(gid_t** *fsgid***);**

## DESCRIPTION

On Linux, a process has both a filesystem group ID and an effective group ID. The (Linux-specific) filesystem group ID is used for permissions checking when accessing filesystem objects, while the effective group ID is used for some other kinds of permissions checks (see **credentials**(7)).

Normally, the value of the process's filesystem group ID is the same as the value of its effective group ID. This is so, because whenever a process's effective group ID is changed, the kernel also changes the filesystem group ID to be the same as the new value of the effective group ID. A process can cause the value of its filesystem group ID to diverge from its effective group ID by using **setfsgid**() to change its filesystem group ID to the value given in *fsgid*.

**setfsgid**() will succeed only if the caller is the superuser or if *fsgid* matches either the caller' s real group ID, effective group ID, saved set-group-ID, or current the filesystem user ID.

## RETURN VALUE

On both success and failure, this call returns the previous filesystem group ID of the caller.

## VERSIONS

This system call is present since Linux 1.2.

## STANDARDS

**setfsgid**() is Linux-specific and should not be used in programs intended to be portable.

## NOTES

The filesystem group ID concept and the **setfsgid**() system call were invented for historical reasons that are no longer applicable on modern Linux kernels. See**setfsuid**(2) for a discussion of wh y the use of both **setfsuid**(2) and **setfsgid**() is nowadays unneeded.

The original Linux **setfsgid**() system call supported only 16-bit group IDs. Subsequently, Linux 2.4 added **setfsgid32**() supporting 32-bit IDs. The glibc **setfsgid**() wrapper function transparently deals with the variation across kernel versions.

### C library/kernel differences

In glibc 2.15 and earlier, when the wrapper for this system call determines that the argument can't be passed to the kernel without integer truncation (because the kernel is old and does not support 32-bit group IDs), it will return −1 and set *errno* to **EINVAL** without attempting the system call.

## BUGS

No error indications of any kind are returned to the caller, and the fact that both successful and unsuccessful calls return the same value makes it impossible to directly determine whether the call succeeded or failed. Instead, the caller must resort to looking at the return value from a further call such as *setfsgid(−1)* (which will always fail), in order to determine if a preceding call to **setfsgid**() changed the filesystem group ID. At the very least, **EPERM** should be returned when the call fails (because the caller lacks the **CAP_SETGID** capability).

## SEE ALSO

**kill**(2), **setfsuid**(2), **capabilities**(7), **credentials**(7)