NAME

issetugid - is current process tainted by uid or gid changes

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <unistd.h>

int
issetugid(*void*);

DESCRIPTION

The **issetugid**() system call returns 1 if the process environment or memory address space is considered "tainted", and returns 0 otherwise.

A process is tainted if it was created as a result of an execve(2) system call which had either of the setuid or setgid bits set (and extra privileges were given as a result) or if it has changed any of its real, effective or saved user or group ID’s since it began execution.

This system call exists so that library routines (eg: libc, libtermcap) can reliably determine if it is safe to use information that was obtained from the user, in particular the results from getenv(3) should be viewed with suspicion if it is used to control operation.

A "tainted" status is inherited by child processes as a result of the fork(2) system call (or other library code that calls fork, such as popen(3)).

It is assumed that a program that clears all privileges as it prepares to execute another will also reset the environment, hence the "tainted" status will not be passed on. This is important for programs such as su(1) which begin setuid but need to be able to create an untainted process.

ERRORS

The **issetugid**() system call is always successful, and no return value is reserved to indicate an error.

SEE ALSO

execve(2), fork(2), setegid(2), seteuid(2), setgid(2), setregid(2), setreuid(2), setuid(2)

HISTORY

The **issetugid**() system call first appeared in OpenBSD 2.0 and was also implemented in FreeBSD 3.0.