NAME

getrandom − obtain a series of random bytes

LIBRARY

Standard C library (*libc*, *−lc*)

SYNOPSIS

#include <sys/random.h>

ssize_t getrandom(void *buf* **[.***buflen***], size_t** *buflen***, unsigned int** *fla gs***);**

DESCRIPTION

The **getrandom**() system call fills the buffer pointed to by *buf* with up to *buflen* random bytes. These bytes can be used to seed user-space random number generators or for cryptographic purposes.

By default, **getrandom**() draws entropy from the *urandom* source (i.e., the same source as the */dev/urandom* device). This behavior can be changed via the *fla gs* argument.

If the *urandom* source has been initialized, reads of up to 256 bytes will always return as many bytes as requested and will not be interrupted by signals. No such guarantees apply for larger buffer sizes. For example, if the call is interrupted by a signal handler, it may return a partially filled buffer, or fail with the error **EINTR**.

If the *urandom* source has not yet been initialized, then **getrandom**() will block, unless **GRND_NONBLOCK** is specified in *flags*.

The *flags* argument is a bit mask that can contain zero or more of the following values ORed together:

GRND_RANDOM

If this bit is set, then random bytes are drawn from the *random* source (i.e., the same source as the */dev/random* device) instead of the *urandom* source. The *random* source is limited based on the entropy that can be obtained from environmental noise. If the number of available bytes in the *random* source is less than requested in *buflen*, the call returns just the available random bytes. If no random bytes are available, the behavior depends on the presence of **GRND_NONBLOCK** in the *flags* argument.

GRND_NONBLOCK

By default, when reading from the *random* source, **getrandom**() blocks if no random bytes are available, and when reading from the *urandom* source, it blocks if the entropy pool has not yet been initialized. If the **GRND_NONBLOCK** flag is set, then **getrandom**() does not block in these cases, but instead immediately returns −1 with *errno* set to **EAGAIN**.

RETURN VALUE

On success, **getrandom**() returns the number of bytes that were copied to the buffer *buf* . This may be less than the number of bytes requested via *buflen* if either **GRND_RANDOM** was specified in *fla gs* and insufficient entropy was present in the *random* source or the system call was interrupted by a signal.

On error, −1 is returned, and *errno* is set to indicate the error.

ERRORS

EAGAIN

The requested entropy was not available, and **getrandom**() would have blocked if the **GRND_NONBLOCK** flag was not set.

EFAULT

The address referred to by *buf* is outside the accessible address space.

EINTR

The call was interrupted by a signal handler; see the description of how interrupted **read**(2) calls on "slow" devices are handled with and without the **SA_RESTART** flag in the **signal**(7) man page.

EINVAL
An invalid flag was specified in *fla gs*.

ENOSYS
The glibc wrapper function for **getrandom**() determined that the underlying kernel does not implement this system call.

VERSIONS

getrandom() was introduced in Linux 3.17. Support was added in glibc 2.25.

STANDARDS

This system call is Linux-specific.

NOTES

For an overview and comparison of the various interfaces that can be used to obtain randomness, see **random**(7).

Unlike */dev/random* and */dev/urandom*, **getrandom**() does not involve the use of pathnames or file descriptors. Thus,**getrandom**() can be useful in cases where **chr oot**(2) makes */dev* pathnames invisible, and where an application (e.g., a daemon during start-up) closes a file descriptor for one of these files that was opened by a library.

Maximum number of bytes returned

As of Linux 3.19 the following limits apply:

- When reading from the *urandom* source, a maximum of 32Mi-1 bytes is returned by a single call to **getrandom**() on systems where *int* has a size of 32 bits.
- When reading from the *random* source, a maximum of 512 bytes is returned.

Interruption by a signal handler

When reading from the *urandom* source (**GRND_RANDOM** is not set), **getrandom**() will block until the entropy pool has been initialized (unless the **GRND_NONBLOCK** flag was specified). If a request is made to read a large number of bytes (more than 256), **getrandom**() will block until those bytes have been generated and transferred from kernel memory to *buf*. When reading from the*r andom* source (**GRND_RANDOM** is set), **getrandom**() will block until some random bytes become available (unless the **GRND_NONBLOCK** flag was specified).

The behavior when a call to **getrandom**() that is blocked while reading from the *urandom* source is interrupted by a signal handler depends on the initialization state of the entropy buffer and on the request size, *buflen*. If the entropy is not yet initialized, then the call fails with the **EINTR** error. If the entropy pool has been initialized and the request size is large (*buflen* > 256), the call either succeeds, returning a partially filled buffer, or fails with the error **EINTR**. If the entropy pool has been initialized and the request size is small (*buflen* <= 256), then **getrandom**() will not fail with **EINTR**. Instead, it will return all of the bytes that have been requested.

When reading from the *random* source, blocking requests of any size can be interrupted by a signal handler (the call fails with the error **EINTR**).

Using **getrandom**() to read small buffers (<= 256 bytes) from the *urandom* source is the preferred mode of usage.

The special treatment of small values of *buflen* was designed for compatibility with OpenBSD's **getentropy**(3), which is nowadays supported by glibc.

The user of **getrandom**() *must* always check the return value, to determine whether either an error occurred or fewer bytes than requested were returned. In the case where **GRND_RANDOM** is not specified and *buflen* is less than or equal to 256, a return of fewer bytes than requested should never happen, but the careful programmer will check for this anyway!

BUGS

As of Linux 3.19, the following bug exists:

- Depending on CPU load, **getrandom**() does not react to interrupts before reading all bytes requested.

SEE ALSO

getentropy(3), **random**(4), **urandom**(4), **random**(7), **signal**(7)