



## ***Linux Ubuntu 26.04 Manual Pages on command 'setstate.3'***

### ***\$ man setstate.3***

random(3)                      Library Functions Manual                      random(3)

#### NAME

random, srand, initstate, setstate - random number generator

#### LIBRARY

Standard C library (libc, -lc)

#### SYNOPSIS

```
#include <stdlib.h>

long random(void);

void srand(unsigned int seed);

char *initstate(size_t n;
               unsigned int seed, char state[n], size_t n);

char *setstate(char *state);
```

Feature Test Macro Requirements for glibc (see feature\_test\_macros(7)):

random(), srand(), initstate(), setstate():

```
_XOPEN_SOURCE >= 500

/* glibc >= 2.19: */ _DEFAULT_SOURCE

/* glibc <= 2.19: */ _SVID_SOURCE || _BSD_SOURCE
```

#### DESCRIPTION

The random() function uses a nonlinear additive feedback random number generator employing a default table of size 31 long integers to return successive pseudo-random numbers in the range from 0 to  $2^{31} - 1$ . The period of this random number generator is very large, approximately  $16 * ((2^{31}) - 1)$ .

The srand() function sets its argument as the seed for a new sequence of pseudo-random in?

tegers to be returned by `random()`. These sequences are repeatable by calling `srandom()` with the same seed value. If no seed value is provided, the `random()` function is automatically seeded with a value of 1.

The `initstate()` function allows a state array `state` to be initialized for use by `random()`.

The size of the state array `n` is used by `initstate()` to decide how sophisticated a random number generator it should use; the larger the state array, the better the random numbers will be. Current "optimal" values for the size of the state array `n` are 8, 32, 64, 128, and 256 bytes; other amounts will be rounded down to the nearest known amount. Using less than 8 bytes results in an error. `seed` is the seed for the initialization, which specifies a starting point for the random number sequence, and provides for restarting at the same point.

The `setstate()` function changes the state array used by the `random()` function. The state array `state` is used for random number generation until the next call to `initstate()` or `setstate()`. `state` must first have been initialized using `initstate()` or be the result of a previous call of `setstate()`.

## RETURN VALUE

The random() function returns a value between 0 and  $(2^{31}) - 1$ . The srand() function re-initializes the random number generator with the seed value. If the seed value is not provided, it turns no value.

The `initstate()` function returns a pointer to the previous state array. On failure, it returns `NULL`, and `errno` is set to indicate the error.

On success, `setstate()` returns a pointer to the previous state array. On failure, it returns `NULL`, and `errno` is set to indicate the error.

## ERRORS

**EINVAL** The state argument given to `setstate()` was `NULL`.

**EINVAL** A state array of less than 8 bytes was specified to `initstate()`.

## ATTRIBUTES

For an explanation of the terms used in this section, see `attributes(7)`.

[illegible]

| ? | Interface | ? | Attribute | ? | Value | ? |
|---|-----------|---|-----------|---|-------|---|
|   |           |   |           |   |       |   |

[illegible]

? random(), srandom(), initstate(), setstate()      ? Thread safety ? MT-Safe ?

[illegible]

## STANDARDS

POSIX.1-2008.

## HISTORY

POSIX.1-2001, 4.3BSD.

## NOTES

Random-number generation is a complex topic. Numerical Recipes in C: The Art of Scientific Computing (William H. Press, Brian P. Flannery, Saul A. Teukolsky, William T. Vetterling; New York: Cambridge University Press, 2007, 3rd ed.) provides an excellent discussion of practical random-number generation issues in Chapter 7 (Random Numbers).

For a more theoretical discussion which also covers many practical issues in depth, see Chapter 3 (Random Numbers) in Donald E. Knuth's The Art of Computer Programming, volume 2 (Seminumerical Algorithms), 2nd ed.; Reading, Massachusetts: Addison-Wesley Publishing Company, 1981.

## CAVEATS

The `random()` function should not be used in multithreaded programs where reproducible behavior is required. Use `random_r(3)` for that purpose.

## BUGS

According to POSIX, `initstate()` should return NULL on error. In the glibc implementation, `errno` is (as specified) set on error, but the function does not return NULL.

## SEE ALSO

`getrandom(2)`, `drand48(3)`, `rand(3)`, `random_r(3)`, `srand(3)`

Linux man-pages 6.17

2026-02-08

`random(3)`