



Rocky Enterprise Linux 9.2 Manual Pages on command 'srand48.3'

\$ man srand48.3

DRAND48(3) Linux Programmer's Manual DRAND48(3)

NAME

drand48, erand48, lrand48, nrand48, mrand48, jrand48, srand48, seed48,
lcong48 - generate uniformly distributed pseudo-random numbers

SYNOPSIS

```
#include <stdlib.h>

double drand48(void);

double erand48(unsigned short xsubi[3]);

long lrand48(void);

long nrand48(unsigned short xsubi[3]);

long mrand48(void);

long jrand48(unsigned short xsubi[3]);

void srand48(long seedval);

unsigned short *seed48(unsigned short seed16v[3]);

void lcong48(unsigned short param[7]);
```

Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

All functions shown above: `_XOPEN_SOURCE`

`|| /* Glibc since 2.19: */ _DEFAULT_SOURCE`

```
|| /* Glibc versions <= 2.19: */ _SVID_SOURCE
```

DESCRIPTION

These functions generate pseudo-random numbers using the linear congruential algorithm and 48-bit integer arithmetic.

The `drand48()` and `erand48()` functions return nonnegative double-precision floating-point values uniformly distributed over the interval $[0.0, 1.0)$.

The `lrand48()` and `rand48()` functions return nonnegative long integers uniformly distributed over the interval $[0, 2^{31})$.

The `mrnd48()` and `jrand48()` functions return signed long integers uniformly distributed over the interval $[-2^{31}, 2^{31})$.

The `srand48()`, `seed48()`, and `lcong48()` functions are initialization functions, one of which should be called before using `drand48()`, `lrand48()` or `mrnd48()`. The functions `erand48()`, `rand48()`, and `jrand48()` do not require an initialization function to be called first.

All the functions work by generating a sequence of 48-bit integers, X_i , according to the linear congruential formula:

$$X_{n+1} = (aX_n + c) \bmod m, \text{ where } n \geq 0$$

The parameter $m = 2^{48}$, hence 48-bit integer arithmetic is performed.

Unless `lcong48()` is called, a and c are given by:

$$a = 0x5DEECE66D$$

$$c = 0xB$$

The value returned by any of the functions `drand48()`, `erand48()`, `lrand48()`, `rand48()`, `mrnd48()` or `jrand48()` is computed by first generating the next 48-bit X_i in the sequence. Then the appropriate number of bits, according to the type of data item to be returned, is copied from the high-order bits of X_i and transformed into the returned value.

The functions `drand48()`, `lrand48()`, and `mrnd48()` store the last 48-bit X_i generated in an internal buffer. The functions `erand48()`, `rand48()`, and `jrand48()` require the calling program to provide storage for the successive X_i values in the array argument `xsubi`. The functions are initialized by placing the initial value of X_i into the array

before calling the function for the first time.

The initializer function `srand48()` sets the high order 32-bits of `Xi` to the argument `seedval`. The low order 16-bits are set to the arbitrary value `0x330E`.

The initializer function `seed48()` sets the value of `Xi` to the 48-bit value specified in the array argument `seed16v`. The previous value of `Xi` is copied into an internal buffer and a pointer to this buffer is returned by `seed48()`.

The initialization function `lcong48()` allows the user to specify initial values for `Xi`, `a`, and `c`. Array argument elements `param[0-2]` specify `Xi`, `param[3-5]` specify `a`, and `param[6]` specifies `c`. After `lcong48()` has been called, a subsequent call to either `srand48()` or `seed48()` will restore the standard values of `a` and `c`.

ATTRIBUTES

For an explanation of the terms used in this section, see [attributes\(7\)](#).

??

Interface	Attribute	Value	
-----------	-----------	-------	--

??

<code>?drand48()</code> , <code>erand48()</code> ,	? Thread safety	? MT-Unsafe race:drand48	?
--	-----------------	--------------------------	---

<code>?lrand48()</code> , <code>nlrand48()</code> ,	?		?
---	---	--	---

<code>?mrand48()</code> , <code>jrand48()</code> ,	?		?
--	---	--	---

<code>?srand48()</code> , <code>seed48()</code> ,	?		?
---	---	--	---

<code>?lcong48()</code>	?	?	?
-------------------------	---	---	---

??

The above functions record global state information for the random number generator, so they are not thread-safe.

CONFORMING TO

POSIX.1-2001, POSIX.1-2008, SVr4.

SEE ALSO

`rand(3)`, `random(3)`

COLOPHON

This page is part of release 5.10 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/>.

2020-11-01

DRAND48(3)