Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'drand48.3p' command

$ man drand48.3p

DRAND48(3P) POSIX Programmer's Manual DRAND48(3P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

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

SYNOPSIS

```
#include <stdlib.h>
double drand48(void);
double erand48(unsigned short xsubi[3]);
long jrand48(unsigned short xsubi[3]);
void lcong48(unsigned short param[7]);
long lrand48(void);
long mrand48(void);
long nrand48(unsigned short xsubi[3]);
unsigned short *seed48(unsigned short seed16v[3]);
void srand48(long seedval);
```

DESCRIPTION

This family of functions shall generate pseudo-random numbers using a linear congruential algorithm and 48-bit integer arithmetic.

The drand48() and erand48() functions shall return non-negative, dou? ble-precision, floating-point values, uniformly distributed over the interval [0.0,1.0).

The lrand48() and nrand48() functions shall return non-negative, long integers, uniformly distributed over the interval [0,231).

The mrand48() and jrand48() functions shall return signed long integers uniformly distributed over the interval [-231,231).

The srand48(), seed48(), and lcong48() functions are initialization en? try points, one of which should be invoked before either drand48(), lrand48(), or mrand48() is called. (Although it is not recommended practice, constant default initializer values shall be supplied auto? matically if drand48(), lrand48(), or mrand48() is called without a prior call to an initialization entry point.) The erand48(), nrand48(), and jrand48() functions do not require an initialization entry point to be called first.

All the routines work by generating a sequence of 48-bit integer val? ues, X_i , according to the linear congruential formula:

Xn+1 = (aX_n +c)mod m n? 0

The parameter m=2^48; hence 48-bit integer arithmetic is performed. Un? less lcong48() is invoked, the multiplier value a and the addend value c are given by:

a = 5DEECE66D16 = 2736731631558

c = B16 = 138

The value returned by any of the drand48(), erand48(), jrand48(), lrand48(), mrand48(), or nrand48() functions is computed by first gen? erating the next 48-bit X_i in the sequence. Then the appropriate num? ber of bits, according to the type of data item to be returned, are copied from the high-order (leftmost) bits of X_i and transformed into the returned value.

The drand48(), lrand48(), and mrand48() functions store the last 48-bit X_i generated in an internal buffer; that is why the application shall ensure that these are initialized prior to being invoked. The erand48(), nrand48(), and jrand48() functions require the calling pro?

gram to provide storage for the successive X_i values in the array specified as an argument when the functions are invoked. That is why these routines do not have to be initialized; the calling program merely has to place the desired initial value of X_i into the array and pass it as an argument. By using different arguments, erand48(), nrand48(), and jrand48() allow separate modules of a large program to generate several independent streams of pseudo-random numbers; that is, the sequence of numbers in each stream shall not depend upon how many times the routines are called to generate numbers for the other streams.

The initializer function srand48() sets the high-order 32 bits of X_i to the low-order 32 bits contained in its argument. The low-order 16 bits of X_i are set to the arbitrary value 330E_16 .

The initializer function seed48() sets the value of X_i to the 48-bit value specified in the argument array. The low-order 16 bits of X_i are set to the low-order 16 bits of seed16v[0]. The mid-order 16 bits of X_i are set to the low-order 16 bits of seed16v[1]. The high-order 16 bits of X_i are set to the low-order 16 bits of seed16v[2]. In addi? tion, the previous value of X_i is copied into a 48-bit internal buf? fer, used only by seed48(), and a pointer to this buffer is the value returned by seed48(). This returned pointer, which can just be ignored if not needed, is useful if a program is to be restarted from a given point at some future time?use the pointer to get at and store the last X_i value, and then use this value to reinitialize via seed48() when the program is restarted.

The initializer function lcong48() allows the user to specify the ini? tial X_i , the multiplier value a, and the addend value c. Argument ar? ray elements param[0-2] specify X_i , param[3-5] specify the multiplier a, and param[6] specifies the 16-bit addend c. After lcong48() is called, a subsequent call to either srand48() or seed48() shall restore the standard multiplier and addend values, a and c, specified above.

The drand48(), lrand48(), and mrand48() functions need not be thread-safe.

RETURN VALUE

As described in the DESCRIPTION above.

ERRORS

No errors are defined.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

These functions should be avoided whenever non-trivial requirements (including safety) have to be fulfilled.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

initstate(), rand()

The Base Definitions volume of POSIX.1?2017, <stdlib.h>

COPYRIGHT