NAME

rand, **srand**, **rand_r** - bad random number generator

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <stdlib.h>

void
srand(*unsigned seed*);

int
rand(*void*);

int
rand_r(*unsigned *ctx*);

DESCRIPTION

The functions described in this manual page are not cryptographically secure. Applications which require unpredictable random numbers should use arc4random(3) instead.

The **rand**() function computes a sequence of pseudo-random integers in the range of 0 to RAND_MAX, inclusive.

The **srand**() function seeds the algorithm with the *seed* parameter. Repeatable sequences of **rand**() output may be obtained by calling **srand**() with the same *seed*. **rand**() is implicitly initialized as if **srand**(*1*) had been invoked explicitly.

In FreeBSD 13, **rand**() is implemented using the same 128-byte state LFSR generator algorithm as random(3). However, the legacy **rand_r**() function is not (and can not be, because of its limited **ctx* size). **rand_r**() implements the historical, poor-quality Park-Miller 32-bit LCG and should not be used in new designs.

IMPLEMENTATION NOTES

Since FreeBSD 13, **rand**() is implemented with the same generator as random(3), so the low-order bits should no longer be significantly worse than the high-order bits.

SEE ALSO

arc4random(3), random(3), random(4)

STANDARDS

The **rand**() and **srand**() functions conform to ISO/IEC 9899:1990 ("ISO C90").

The **rand_r**() function is not part of ISO/IEC 9899:1990 ("ISO C90") and is marked obsolescent in IEEE Std 1003.1-2008 ("POSIX.1"). It may be removed in a future revision of POSIX.

CAVEATS

Prior to FreeBSD 13, **rand**() used the historical Park-Miller generator with 32 bits of state and produced poor quality output, especially in the lower bits. **rand**() in earlier versions of FreeBSD, as well as other standards-conforming implementations, may continue to produce poor quality output.

These functions should not be used in portable applications that want a high quality or high performance pseudorandom number generator. One possible replacement, random(3), is portable to Linux -- but it is not especially fast, nor standardized.

If broader portability or better performance is desired, any of the widely available and permissively licensed SFC64/32, JSF64/32, PCG64/32, or SplitMix64 algorithm implementations may be embedded in your application. These algorithms have the benefit of requiring less space than random(3) and being quite fast (in header inline implementations).