NAME

arc4random, **arc4random_buf**, **arc4random_uniform** **-** random number generator

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <stdlib.h>

uint32_t
arc4random(*void*);

void
arc4random_buf(*void *buf, size_t nbytes*);

uint32_t
arc4random_uniform(*uint32_t upper_bound*);

DESCRIPTION

This family of functions provides higher quality data than those described in rand(3), random(3), and rand48(3).

Use of these functions is encouraged for almost all random number consumption because the other interfaces are deficient in either quality, portability, standardization, or availability. These functions can be called in almost all coding environments, including pthread(3) and chroot(2).

High quality 32-bit pseudo-random numbers are generated very quickly. On each call, a cryptographic pseudo-random number generator is used to generate a new result. One data pool is used for all consumers in a process, so that consumption under program flow can act as additional stirring. The subsystem is re-seeded from the kernel random(4) subsystem using getentropy(3) on a regular basis, and also upon fork(2).

The **arc4random**() function returns a single 32-bit value. The **arc4random**() function returns pseudo-random numbers in the range of 0 to (2**32)-1, and therefore has twice the range of rand(3) and random(3).

arc4random_buf() fills the region *buf* of length *nbytes* with random data.

arc4random_uniform() will return a single 32-bit value, uniformly distributed but less than *upper_bound*. This is recommended over constructions like "arc4random() % upper_bound" as it avoids "modulo bias" when the upper bound is not a power of two. In the worst case, this function may consume multiple iterations to ensure uniformity; see the source code to understand the problem and solution.

RETURN VALUES

These functions are always successful, and no return value is reserved to indicate an error.

EXAMPLES

The following produces a drop-in replacement for the traditional **rand**() and **random**() functions using **arc4random**():

```
#define foo4random() (arc4random_uniform(RAND_MAX + 1))
```

SEE ALSO

rand(3), rand48(3), random(3)

Daniel J. Bernstein, *ChaCha, a variant of Salsa20*, http://cr.yp.to/papers.html#chacha, 2008-01-28, Document ID: 4027b5256e17b9796842e6d0f68b0b5e.

HISTORY

These functions first appeared in OpenBSD 2.1.

The original version of this random number generator used the RC4 (also known as ARC4) algorithm. In OpenBSD 5.5 it was replaced with the ChaCha20 cipher, and it may be replaced again in the future as cryptographic techniques advance. A good mnemonic is "A Replacement Call for Random".

The **arc4random**() random number generator was first introduced in FreeBSD 2.2.6. The ChaCha20 based implementation was introduced in FreeBSD 12.0, with obsolete stir and addrandom interfaces removed at the same time.