

**NAME**

`rand`, `rand_r`, `srand` – pseudo-random number generator

**SYNOPSIS**

```
#include <stdlib.h>
```

```
int rand(void);
```

```
int rand_r(unsigned int *seedp);
```

```
void srand(unsigned int seed);
```

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

```
rand_r(): _POSIX_C_SOURCE >= 1 || _XOPEN_SOURCE || _POSIX_SOURCE
```

**DESCRIPTION**

The **rand**() function returns a pseudo-random integer in the range 0 to **RAND\_MAX** inclusive (i.e., the mathematical range [0, **RAND\_MAX**]).

The **srand**() function sets its argument as the seed for a new sequence of pseudo-random integers to be returned by **rand**(). These sequences are repeatable by calling **srand**() with the same seed value.

If no seed value is provided, the **rand**() function is automatically seeded with a value of 1.

The function **rand**() is not reentrant or thread-safe, since it uses hidden state that is modified on each call. This might just be the seed value to be used by the next call, or it might be something more elaborate. In order to get reproducible behavior in a threaded application, this state must be made explicit; this can be done using the reentrant function **rand\_r**().

Like **rand**(), **rand\_r**() returns a pseudo-random integer in the range [0, **RAND\_MAX**]. The *seedp* argument is a pointer to an *unsigned int* that is used to store state between calls. If **rand\_r**() is called with the same initial value for the integer pointed to by *seedp*, and that value is not modified between calls, then the same pseudo-random sequence will result.

The value pointed to by the *seedp* argument of **rand\_r**() provides only a very small amount of state, so this function will be a weak pseudo-random generator. Try **drand48\_r(3)** instead.

**RETURN VALUE**

The **rand**() and **rand\_r**() functions return a value between 0 and **RAND\_MAX** (inclusive). The **srand**() function returns no value.

**CONFORMING TO**

The functions **rand**() and **srand**() conform to SVr4, 4.3BSD, C89, C99, POSIX.1-2001. The function **rand\_r**() is from POSIX.1-2001. POSIX.1-2008 marks **rand\_r**() as obsolete.

**NOTES**

The versions of **rand**() and **srand**() in the Linux C Library use the same random number generator as **random(3)** and **srandom(3)**, so the lower-order bits should be as random as the higher-order bits. However, on older **rand**() implementations, and on current implementations on different systems, the lower-order bits are much less random than the higher-order bits. Do not use this function in applications intended to be portable when good randomness is needed. (Use **random(3)** instead.)

**EXAMPLE**

POSIX.1-2001 gives the following example of an implementation of **rand**() and **srand**(), possibly useful when one needs the same sequence on two different machines.

```
static unsigned long next = 1;
```

```
/* RAND_MAX assumed to be 32767 */
```

```
int myrand(void) {
    next = next * 1103515245 + 12345;
    return((unsigned)(next/65536) % 32768);
}

void mysrand(unsigned int seed) {
    next = seed;
}
```

The following program can be used to display the pseudo-random sequence produced by **rand()** when given a particular seed.

```
#include <stdlib.h>
#include <stdio.h>

int
main(int argc, char *argv[])
{
    int j, r, nloops;
    unsigned int seed;

    if (argc != 3) {
        fprintf(stderr, "Usage: %s <seed> <nloops>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    seed = atoi(argv[1]);
    nloops = atoi(argv[2]);

    srand(seed);
    for (j = 0; j < nloops; j++) {
        r = rand();
        printf("%d\n", r);
    }

    exit(EXIT_SUCCESS);
}
```

## SEE ALSO

**drand48(3)**, **random(3)**

## COLOPHON

This page is part of release 3.74 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 <http://www.kernel.org/doc/man-pages/>.