NAME

random, srandom, initstate, setstate − better random number generator; routines for changing generators

SYNOPSIS

#include <stdlib.h>

long random(void)
void srandom(unsigned *seed***)**
char *initstate(unsigned *seed***, char** ***state*****, int** *n***)**
char *setstate(char ***state*****)**

DESCRIPTION

Random uses a non-linear additive feedback random number generator employing a default table of size 31 long integers to return successive pseudo-random numbers in the range from 0 to $2^{31}-1$. The period of this random number generator is very large, approximately $16\times(2^{31}-1)$.

Random/srandom have (almost) the same calling sequence and initialization properties as **rand/srand.** The difference is that **rand**(3) produces a much less random sequence — in fact, the low dozen bits generated by rand go through a cyclic pattern. All the bits generated by **random** are usable. For example, ‘‘random()&01’’ will produce a random binary value.

Unlike **srand**, **srandom** does not return the old seed; the reason for this is that the amount of state information used is much more than a single word. (Two other routines are provided to deal with restarting/changing random number generators). Like **rand**(3), however, **random** will by default produce a sequence of numbers that can be duplicated by calling **srandom** with **1** as the seed.

The **initstate** routine allows a state array, passed in as an argument, to be initialized for future use. The size of the state array (in bytes) is used by **initstate** to decide how sophisticated a random number generator it should use -- the more state, the better the random numbers will be. (Current "optimal" values for the amount of state information are 8, 32, 64, 128, and 256 bytes; other amounts will be rounded down to the nearest known amount. Using less than 8 bytes will cause an error). The seed for the initialization (which specifies a starting point for the random number sequence, and provides for restarting at the same point) is also an argument. **Initstate** returns a pointer to the previous state information array.

Once a state has been initialized, the **setstate** routine provides for rapid switching between states. **Setstate** returns a pointer to the previous state array; its argument state array is used for further random number generation until the next call to **initstate** or **setstate**.

Once a state array has been initialized, it may be restarted at a different point either by calling **initstate** (with the desired seed, the state array, and its size) or by calling both **setstate** (with the state array) and **srandom** (with the desired seed). The advantage of calling both **setstate** and **srandom** is that the size of the state array does not have to be remembered after it is initialized.

With 256 bytes of state information, the period of the random number generator is greater than 2^{69}, which should be sufficient for most purposes.

AUTHOR

Earl T. Cohen

DIAGNOSTICS

If **initstate** is called with less than 8 bytes of state information, or if **setstate** detects that the state information has been garbled, error messages are printed on the standard error output.

SEE ALSO

rand(3).

NOTES

initstate and **setstate** are not declared in *<stdlib.h>*, programmers must provide their own declarations.

BUGS

About 2/3 the speed of **rand**(3).