

Re: C-faq q13.20

Source: http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2006-11/msg02719.html

- *From:* CBFalconer <cbfalconer@xxxxxxxxxx>
 - *Date:* Sun, 19 Nov 2006 08:55:47 -0500
-

Ben Bacarisse wrote:

.... snip ...

or (if the your source in random bits is rather precious) the decision must be based on a bit or bits in the original sample that are not correlated to the size of the original y :

```
int r = rand();
y = (double)(r + 1) / (RAND_MAX + 1);
y = std_dev * sqrt(-2 * log(y));
return r & 1 ? mean + y : mean - y;
```

I can't see the advantage of the fabs. I think the above allows y to be as small as (randomly) possible whilst remaining > 0 . Of course, this last version raises the question of the suitability of using the bottom bits returned by `rand()` which was well thrashed out about a year ago.

I should add that even then I don't think the result is normally distributed (though it will at least be symmetrical).

Cfaq 13.20 covers it.

13.20: How can I generate random numbers with a normal or Gaussian distribution?

A: Here is one method, recommended by Knuth and due originally to Marsaglia:

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

double gaussrand()
{
```

```
static double V1, V2, S;
static int phase = 0;
double X;

if(phase == 0) {
do {
double U1 = (double)rand() / RAND_MAX;
double U2 = (double)rand() / RAND_MAX;

V1 = 2 * U1 - 1;
V2 = 2 * U2 - 1;
S = V1 * V1 + V2 * V2;
} while(S >= 1 || S == 0);

X = V1 * sqrt(-2 * log(S) / S);
} else
X = V2 * sqrt(-2 * log(S) / S);

phase = 1 - phase;

return X;
}
```

See the extended versions of this list (see question 20.40) for other ideas.

References: Knuth Sec. 3.4.1 p. 117; Marsaglia and Bray, "A Convenient Method for Generating Normal Variables"; Press et al., *Numerical Recipes in C* Sec. 7.2 pp. 288–290.

—

Chuck F (cbfalconer at mainline dot net)
Available for consulting/temporary embedded and systems.
<<http://cbfalconer.home.att.net>>

.