Ruby master - Bug #3104

Random: seeding issues

04/07/2010 03:08 PM - marcandre (Marc-Andre Lafortune)

Status:	Closed		
Priority:	Normal		
Assignee:	marcandre (Marc-Andre Lafortune)		
Target version:	1.9.2		
ruby -v:	ruby 1.9.2dev (2010-04-03 trunk 27200) [x86_64-darwin10.2.0]	**Backport:**	

Description

=begin
I think there are a couple of very small errors with the seeding of the Random class.

1) Seeding sometimes ignores the high bit of seed data.

```
Random.new((1<<64)-1).rand == Random.new((1 << 65) -1).rand
 # => true, should be false
```

Probably some leftover code?

diff --git a/random.c b/random.c
index 02d081c..447e59f 100644
--- a/random.c
+++ b/random.c
@@ -411,8 +411,6 @@ rand_init(struct MT *mt, VALUE vseed)
init_genrand(mt, buf[0]);
}
else {

- if (buf[len-1] == 1) /* remove leading-zero-guard */
- len--; init_by_array(mt, buf, len); } if (buf != buf0) xfree(buf);

2) Treatment of negative seed values is currently platform dependent for all negative fixnums. From the same negative seed, the random sequence can be different on 32 bit platforms than on 64 bit ones.

```
Random.new(-1).rand == Random.new((1<<63) -1).rand
# => true on 64 bit platform, false on 32 bit
Random.new(-1).rand == Random.new((1<<31) -1).rand
# => false on 64 bit platform, true on 32 bit
```

The simple solution below ignores the sign (which is what's done in case of negative Bignum). This means that the same values will result from seeding with x or -x. Another solution would be to use the lowest bit for the sign (for both fixnums and bignums) so as to avoid this collision, or raise an error in case of negative seeds.

diff --git a/random.c b/random.c
index 02d081c..272f7b5 100644
--- a/random.c
+++ b/random.c
@@ -367,6 +367,7 @@ rand_init(struct MT *mt, VALUE vseed)
{
volatile VALUE seed;
long blen = 0;

- long fixnum_seed; int i, j, len; unsigned int buf0[SIZEOF_LONG / SIZEOF_INT32 * 4], *buf = buf0;

@@ -374,9 +375,11 @@ rand_init(struct MT *mt, VALUE vseed)
switch (TYPE(seed)) {
case T_FIXNUM:
len = 1;

- buf[0] = (unsigned int)(FIX2ULONG(seed) & 0xffffffff);

- fixnum_seed = FIX2LONG(seed);
- if (fixnum_seed < 0) fixnum_seed = -fixnum_seed;
- buf[0] = (unsigned int)(fixnum_seed & 0xffffffff); #if SIZEOF_LONG > SIZEOF_INT32
- if ((buf[1] = (unsigned int)(FIX2ULONG(seed) >> 32)) != 0) ++len;
- if ((buf[1] = (unsigned int)(fixnum_seed >> 32)) != 0) ++len; #endif break; case T_BIGNUM:

3) Finally, I was wondering if there wasn't a typo trashing some of the bits in the initial states when seeding with an array:

diff --git a/random.c b/random.c
index 02d081c..f3db095 100644
--- a/random.c
+++ b/random.c
@@ -151,7 +151,7 @@ init_by_array(struct MT *mt, unsigned int init_key[], int key_length)
if (i>=N) { mt->state[0] = mt->state[N-1]; i=1; }
}

- mt->state[0] = 0x80000000U; /* MSB is 1; assuring non-zero initial array */
- mt->state[0] |= 0x80000000U; /* MSB is 1; assuring non-zero initial array */ }

static void

These changes will surely create test failures which should be trivial to fix.
=end

History

#1 - 04/21/2010 09:25 PM - mame (Yusuke Endoh)

- Status changed from Open to Closed

- Assignee changed from nobu (Nobuyoshi Nakada) to marcandre (Marc-Andre Lafortune)

=begin
Hi,

> 1) Seeding sometimes ignores the high bit of seed data.
>
> Random.new((1< true, should be false

I really agree that this is uncool.
But it should not be changed from three reasons.

A) compatibility
This feature is already released in 1.9.1:

$ ruby-1.9.1-p378 -e 'srand((1 << 64)-1); p rand'
0.0218256954012701
$ ruby-1.9.1-p378 -e 'srand((1 << 65)-1); p rand'
0.0218256954012701
$ ./ruby -ve 'p Random.new((1 << 64)-1).rand'
ruby 1.9.2dev (2010-04-21 trunk 27426) [i686-linux]
0.021825695401270107

B) no guarantee for decorrelation
From what I hear, it is never guaranteed at for two random
sequences (generated from two different seeds) to be
uncorrelated.
The fact is also true for Mersenne Twister itself.

C) seed bias
As akr said, your patch has potential for bias.
This is because MSB of any Bignum always 1.
Incidentally, I prefer removing MSB to current implementation.

> 2) Treatment of negative seed values is currently platform dependent for all negative fixnums. From the same negative seed, the random sequence can be different on 32 bit platforms than on 64 bit ones.

Umm. I bet portability beats compatibility in this case.
This patch has been already imported. Thanks!

--
Yusuke Endoh mame@tsg.ne.jp
=end

#2 - 04/21/2010 10:08 PM - mame (Yusuke Endoh)

=begin
2010/4/21 Tanaka Akira akr@fsij.org:

> 2010/4/21 Yusuke Endoh redmine@ruby-lang.org:
>
> > 1) Seeding sometimes ignores the high bit of seed data.
> >
> > Random.new((1< true, should be false
> >
> > I really agree that this is uncool.
> >
> > Incidentally, I prefer removing MSB to current implementation.
>
> I guess it is worse in the sense of the reporter.
>
> Probably it causes
> Random.new(2*32).rand ==
> Random.new(233).rand ==
> Random.new(234).rand ==
> ...
> Random.new(2*63).rand.

You are right :-)

--
Yusuke Endoh mame@tsg.ne.jp

=end