I think we should deprecate the Random::DEFAULT constant, it doesn't make sense anymore and it's longer than using Random class methods (Random.rand) or Kernel instance methods (#rand).

Also, people might expect it to be global.

If users want a Random instance they should just use Random.new, not assume there is a global instance in Random::DEFAULT, which is actually rather misleading now (Random::DEFAULT is no longer an instance of Random).

Also note that JRuby & TruffleRuby use a per-thread instance for Kernel#rand, etc, to avoid contention (otherwise it becomes a huge source of contention when threads run in parallel).

Which means on those implementations using Random::DEFAULT was inefficient (extra synchronization).

So for all these reasons I think it's time to deprecate Random::DEFAULT and then later remove it (in 3.1?).

I don't think there is any use case for Random::DEFAULT, but happy to hear if there is and there is no trivial replacement.

As mentioned here: https://bugs.ruby-lang.org/issues/17322#note-7 there are several users.

For example, I will use Random::DEFAULT for the default random generator for dice rolling method like:

def def roll rnd = Random::DEFAULT
  rnd.rand(6)+1
end

for hobby use (non-serious use case).
I don't against to remove `Random::DEFAULT`, but need to care about current users.

```ruby
def roll rnd = nil
    (rnd ? rnd.rand(6) : rand(6)) + 1
end
```

is one way, but I think some people can like previous definition.

---

It seems pretty rare to need to supply a custom Random instance. But in such a case, it would be easy to create an explicit one in e.g. the constructor:

```ruby
class Dice
    def initialize(random = Random.new)
        @random = random
    end

    def roll
        @random.rand(6) + 1
    end
end
```

If it needs to scale and run well with parallel threads, the built-in Kernel#rand is probably the only good option.

---

OK, accepted.

Matz.

---

Remove references to `Random::DEFAULT` in the documentation

- `Random::DEFAULT` is no longer a Random instance, and referencing it is more confusing than helpful. Related to [Feature #17351](https://github.com/ruby/ruby/issues/17351)