Ruby master - Bug #4284

Timeout.timeout may cause application exit unintetionally, again

01/17/2011 02:44 PM - kosaki (Motohiro KOSAKI)

Status:	Closed		
Priority:	Normal		
Assignee:	matz (Yukihiro Matsumoto)		
Target version:	1.9.3		
ruby -v:	ruby 1.9.3dev (2010-12-22 trunk 30291) [x86_64-linux]	**Backport:**	

Description

=begin
This issue was discovered during [Bug#4266] discussion.
Current timeout is racy.

Now, timeout module has following code.

```
def timeout()
begin
x = Thread.current
y = Thread.start {
begin
sleep sec
rescue => e
x.raise e
else
x.raise exception, "execution expired" if x.alive?
end
}
return yield(sec)
rescue exception => e
raise Error, e.message, e.backtrace
ensure
if y and y.alive?
y.kill
y.join # make sure y is dead.
end
end
end
```

Unfortunatelly,

```
y = Thread.start {}
```

is not an atomic operation. Then, A following race can occur.

CPU0(thread x) CPU1(thread y) remark

---

enter begin block
[thread construct] but no assign y yet
sleep sec
wakeup from sleep
x.raise
if y return false. (see above)

Therefore, CPU0 don't call y.join and leak y's thread resource. C# have solved
this two-step-construction vs asynchrounous exception race by RAII.

But unfortunately, Ruby don't have such language feature. So, We can't write

async-exception-safe code. One of solution is to move timeout module from ruby code
into c code as JRuby does. But I don't think timeout is only asynchrounos exception user.
we also have Interrupt class (for Ctrl-C) and I think we need to allow to write async
exception safe code by ruby.

Or, Am I missing something?
=end

History

#1 - 01/17/2011 04:05 PM - kosaki (Motohiro KOSAKI)

- Status changed from Open to Closed

=begin
ruby-dev is no good place for this discussion. So, I'll close this ticket and reopen at ruby-core.

I'm sorry.
=end