Ruby - Feature #3620

Add Queue, SIzedQueue and ConditionVariable implementations in C in addition to ruby ones

07/28/2010 02:51 AM - panaggio (Ricardo Panaggio)

Status:	Closed
Priority:	Normal
Assignee:	ko1 (Koichi Sasada)
Target version:	2.6

Description

=begin
Queue, SizedQueue and ConditionVariable are important synchronization primitives and are nowadays implemented in Ruby.

Attached patch (initiated by myself and heavily enriched by Nobu) contains these sync primitives implemented in C, which makes them faster (see [1] for the benchmark's code):

```
Rehearsal ----------------------------------------------
Q#push          1.590000   0.010000   1.600000 (  1.605502)
T#push          0.600000   0.010000   0.610000 (  0.630444)
Q#pop           4.390000   0.000000   4.390000 (  4.389781)
T#pop           0.580000   0.000000   0.580000 (  0.578918)
Q#empty?        0.480000   0.000000   0.480000 (  0.484305)
T#empty?        0.360000   0.000000   0.360000 (  0.358559)
Q#clear         1.210000   0.000000   1.210000 (  1.214494)
T#clear         0.600000   0.000000   0.600000 (  0.588611)
Q#size          0.370000   0.000000   0.370000 (  0.365587)
T#size          0.350000   0.000000   0.350000 (  0.356985)
Q#num_waiting   0.380000   0.000000   0.380000 (  0.379199)
T#num_waiting   0.370000   0.000000   0.370000 (  0.368075)
------------------------------------- total: 11.300000sec
```

It has already been discussed on ruby-core (see ruby-core:31100).

This patch is one of the deliverables of my RubySoC project (slot #17): "Improving Ruby's Synchronization Primitives and Core Libraries" [2,3]

[1] http://github.com/panaggio/rubysoc-2010/blob/master/benchmarks/queue.rb
[2] http://pastebin.com/viSnfqe6
[3] http://rubysoc.org/projects
=end

Associated revisions

Revision e334bb2ce5d8876b020ab681f21595e2e1c9d601 - 09/06/2013 03:15 PM - Glass_saga (Masaki Matsushita)

- common.mk: use RUNRUBY instead of MINIRUBY because MINIRUBY can't require extension libraries. The patch is from nobu (Nobuyoshi Nakada).
- ext/thread/extconf.rb: for build ext/thread/thread.c.
- include/ruby/intern.h: ditto.
- thread.c: ditto.
- lib/thread.rb: removed and replaced by ext/thread/thread.c.
- ext/thread/thread.c: Queue, SizedQueue and ConditionVariable implementations in C. This patch is based on patches from panaggio (Ricardo Panaggio) and funny_falcon (Yura Sokolov) and ko1 (Koichi Sasada). [ruby-core:31513] [Feature #3620]
- test/thread/test_queue.rb (test_queue_thread_raise): add a test for ensuring that killed thread should be removed from waiting threads.

It is based on a code by ko1 (Koichi Sasada). [ruby-core:45950]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@42862 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision e334bb2c - 09/06/2013 03:15 PM - Glass_saga (Masaki Matsushita)

- common.mk: use RUNRUBY instead of MINIRUBY because MINIRUBY can't require extension libraries. The patch is from nobu (Nobuyoshi Nakada).
- ext/thread/extconf.rb: for build ext/thread/thread.c.
- include/ruby/intern.h: ditto.
- thread.c: ditto.
- lib/thread.rb: removed and replaced by ext/thread/thread.c.
- ext/thread/thread.c: Queue, SizedQueue and ConditionVariable implementations in C. This patch is based on patches from panaggio (Ricardo Panaggio) and funny_falcon (Yura Sokolov) and ko1 (Koichi Sasada). [ruby-core:31513] [Feature #3620]
- test/thread/test_queue.rb (test_queue_thread_raise): add a test for ensuring that killed thread should be removed from waiting threads. It is based on a code by ko1 (Koichi Sasada). [ruby-core:45950]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@42862 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 06/26/2011 01:30 PM - akr (Akira Tanaka)

- Project changed from 8 to Ruby

- Category changed from ext to ext

#2 - 12/17/2011 10:00 AM - normalperson (Eric Wong)

ping? The subject of Queue performance came up again in:
http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-talk/391324

#3 - 12/17/2011 10:23 AM - normalperson (Eric Wong)

Issue #3620 has been updated by Eric Wong.

ping? The subject of Queue performance came up again in:
http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-talk/391324

Feature #3620: Add Queue, SIzedQueue and ConditionVariable implementations in C in addition to ruby ones
http://redmine.ruby-lang.org/issues/3620

Author: Ricardo Panaggio
Status: Open
Priority: Normal
Assignee:
Category: ext
Target version:

=begin
Queue, SizedQueue and ConditionVariable are important synchronization primitives and are nowadays implemented in Ruby.

Attached patch (initiated by myself and heavily enriched by Nobu) contains these sync primitives implemented in C, which makes them faster (see [1] for the benchmark's code):

```
Rehearsal ----------------------------------------------
Q#push         1.590000   0.010000   1.600000 (  1.605502)
T#push         0.600000   0.010000   0.610000 (  0.630444)
Q#pop          4.390000   0.000000   4.390000 (  4.389781)
T#pop          0.580000   0.000000   0.580000 (  0.578918)
```

```
Q#empty?       0.480000   0.000000   0.480000 (  0.484305)
T#empty?       0.360000   0.000000   0.360000 (  0.358559)
Q#clear        1.210000   0.000000   1.210000 (  1.214494)
T#clear        0.600000   0.000000   0.600000 (  0.588611)
Q#size         0.370000   0.000000   0.370000 (  0.365587)
T#size         0.350000   0.000000   0.350000 (  0.356985)
Q#num_waiting  0.380000   0.000000   0.380000 (  0.379199)
T#num_waiting  0.370000   0.000000   0.370000 (  0.368075)
---------------------------------------- total: 11.300000sec
```

It has already been discussed on ruby-core (see ruby-core:31100).

This patch is one of the deliverables of my RubySoC project (slot #17): "Improving Ruby's Synchronization Primitives and Core Libraries" [2,3]

[1] http://github.com/panaggio/rubysoc-2010/blob/master/benchmarks/queue.rb
[2] http://pastebin.com/viSnfqe6
[3] http://rubysoc.org/projects
=end

--
http://redmine.ruby-lang.org

#4 - 03/18/2012 05:52 PM - nahi (Hiroshi Nakamura)

- Description updated

- Assignee set to ko1 (Koichi Sasada)

#5 - 03/18/2012 06:46 PM - shyouhei (Shyouhei Urabe)

- Status changed from Open to Assigned

#6 - 03/19/2012 04:46 AM - funny_falcon (Yura Sokolov)

Related issue (but about plain ruby realization)
https://bugs.ruby-lang.org/issues/6174

#7 - 06/26/2012 04:35 AM - ko1 (Koichi Sasada)

Sorry for absent from this request.

Eric, can this patch work fine on current trunk?
... which patch should I use?

#8 - 06/26/2012 09:53 AM - normalperson (Eric Wong)

"ko1 (Koichi Sasada)" redmine@ruby-lang.org wrote:

> Eric, can this patch work fine on current trunk?
> ... which patch should I use?

Yes, I just tested final-queue.patch with current trunk (r36217)
and it works.

The .gitignore hunk in the patch can be ignored, everything
else creates a new file and doesn't conflict.

This is Richard's patch, btw. I'm not sure if Richard still
pays attention to Ruby (haven't seen any activity from him in
a while).

push/pop are still significantly faster on current trunk with the
queue.rb benchmark:

```
                  user     system      total        real
```

```
Q#push         0.640000   0.000000   0.640000 (  0.642801)
T#push         0.200000   0.000000   0.200000 (  0.197539)
Q#pop          1.390000   0.000000   1.390000 (  1.393139)
T#pop          0.230000   0.000000   0.230000 (  0.228802)
```

(https://raw.github.com/panaggio/rubysoc-2010/master/benchmarks/queue.rb)

#9 - 06/26/2012 04:18 PM - funny_falcon (Yura Sokolov)

- File final_queue_without_mutex.diff added

It seems that there is no need for mutex in a native queue implementation (considering we have GVL),
and so that queue_synchronized wrapper:
rb_mutex_sleep could be replaces with rb_thread_sleep_forever without semantic change.

Without mutex, native queue becomes 2 times faster.
Modified patch is attached.

#10 - 06/29/2012 05:53 PM - ko1 (Koichi Sasada)

Hi,

(2012/06/26 16:18), funny_falcon (Yura Sokolov) wrote:

> It seems that there is no need for mutex in a native queue implementation (considering we have GVL),
> and so that queue_synchronized wrapper:
> rb_mutex_sleep could be replaces with rb_thread_sleep_forever without semantic change.
>
> Without mutex, native queue becomes 2 times faster.
> Modified patch is attached.

I found a bug.

begin sample code

```
q = Thread::Queue.new
#q = Queue.new
th1 = Thread.new{
begin
p [:th1, q.pop]
rescue RuntimeError => e
sleep
p e
end
}
th2 = Thread.new{
sleep 0.1
p [:th2, q.pop]
}
p [th1, th2]
sleep 0.5
th1.raise "async interrupt!"
sleep 0.5
q << :s
```

th1.join

```
p [th1, th2]
th2.join          # BLOCK forever!
```

end sample code

When th1 escapes from blocking by "pop" by exception, then waiting list
of Queue should be maintained (remove th1 from waiting list).

Other comment:

- "extthread" is good name?
- variable name

```
static void
wakeup_all_threads(VALUE list)
{
VALUE thread, list0 = list;
long i;

 list = rb_ary_subseq(list, 0, LONG_MAX);
 rb_ary_clear(list0);
 for (i = 0; i < RARRAY_LEN(list); ++i) {
```

```
thread = RARRAY_PTR(list)[i];
rb_thread_wakeup_alive(thread);
 }
 RB_GC_GUARD(list);
```

}

I prefer:

static void
wakeup_all_threads(VALUE list0)
{
VALUE thread, list = rb_ary_subseq(list0, 0, LONG_MAX);
long i;

```
 rb_ary_clear(list0);
 for (i = 0; i < RARRAY_LEN(list); ++i) {
thread = RARRAY_PTR(list)[i];
rb_thread_wakeup_alive(thread);
 }
 RB_GC_GUARD(list);
```

}

But I prefer more:

static void
wakeup_all_threads(VALUE list)
{
VALUE thread;
long i;

```
 for (i = 0; i < RARRAY_LEN(list); ++i) {
thread = RARRAY_PTR(list)[i];
rb_thread_wakeup_alive(thread);
 }
 rb_ary_clear(list);
```

}

Any reason to dup array before iteration?

- T_DATA -> T_STRUCT or T_OBJECT

In this case, you don't need to use T_DATA. You can only use T_STRUCT
or T_OBJECT (with hidden attr). Maybe it will be simple.

Thanks,
Koichi

> Feature #3620: Add Queue, SIzedQueue and ConditionVariable implementations in C in addition to ruby ones
> https://bugs.ruby-lang.org/issues/3620#change-27471

--
// SASADA Koichi at atdot dot net

#11 - 09/22/2012 09:00 AM - ko1 (Koichi Sasada)

ping.

#12 - 10/30/2012 08:39 AM - ko1 (Koichi Sasada)

- Target version set to 2.6

ping.
I think it can be introduce into Ruby 2.0 if there is a nice implementation.

#13 - 05/17/2013 10:59 PM - Glass_saga (Masaki Matsushita)

- File patch.diff added

I fixed some bugs:

(1) blocking forever bug pointed out by ko1 in [ruby-core:45950]
(2) SEGV in do_sleep()
(3) SizedQueue's bug which is similar to (1)

Now, the C implementation passes test-all.
Following diff is from final_queue_without_mutex.diff to my patch.

diff --git a/ext/thread/thread.c b/ext/thread/thread.c
index 9aff5e5..33365c1 100644
--- a/ext/thread/thread.c
+++ b/ext/thread/thread.c
@@ -150,7 +150,7 @@ static VALUE
do_sleep(VALUE args)
{
struct sleep_call *p = (struct sleep_call *)args;

- return rb_funcall(p->argv[0], rb_intern("sleep"), p->argc-1, p->argv+1);

- return rb_funcall(p->argv[0], rb_intern("sleep"), p->argc-1, p->argv[1]); /* (2) */
  }

static VALUE
@@ -339,15 +339,31 @@ rb_queue_push(VALUE self, VALUE obj)
return self;
}

+struct waiting_delete {

- VALUE waiting;
- VALUE th;
  +};

+static VALUE
+queue_delete_from_waiting(struct waiting_delete *p)
+{

- rb_ary_delete(p->waiting, p->th);
- return Qnil;
  +}

static VALUE
queue_do_pop(Queue *queue, VALUE should_block)
{

- while (!RARRAY_LEN(queue->que)) {

- struct waiting_delete args;

- while (RARRAY_LEN(queue->que) == 0) {
  if (!(int)should_block) {
  rb_raise(rb_eThreadError, "queue empty");
  }

- [redacted]

- [redacted]

- [redacted]

- [redacted]

- [redacted]

- [redacted]

  }

  return rb_ary_shift(queue->que);
  @@ -590,9 +606,14 @@ rb_szqueue_max_set(VALUE self, VALUE vmax)
  static VALUE
  szqueue_do_push(SizedQueue *szqueue, VALUE obj)

{

- struct waiting_delete args;
- VALUE thread;
- while (queue_length(&szqueue->queue_) >= szqueue->max) {
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]

}
return queue_do_push(&szqueue->queue_, obj);
}

#14 - 05/17/2013 11:19 PM - Glass_saga (Masaki Matsushita)

Here is benchmark results.
I used the code same as [ruby-core:45871].
(https://raw.github.com/panaggio/rubysoc-2010/master/benchmarks/queue.rb)

trunk (r40799):
user system total real
Q#push 3.290000 0.000000 3.290000 (3.292121)
Q#pop 1.900000 0.000000 1.900000 (1.898899)
Q#empty? 0.120000 0.000000 0.120000 (0.122000)
Q#clear 1.700000 0.010000 1.710000 (1.706695)
Q#size 0.130000 0.000000 0.130000 (0.128248)
Q#num_waiting 0.120000 0.000000 0.120000 (0.117081)

proposed:
user system total real
T#push 0.130000 0.000000 0.130000 (0.138489)
T#pop 0.120000 0.000000 0.120000 (0.119966)
T#empty? 0.100000 0.000000 0.100000 (0.104266)
T#clear 0.130000 0.000000 0.130000 (0.135279)
T#size 0.110000 0.000000 0.110000 (0.104839)
T#num_waiting 0.100000 0.000000 0.100000 (0.097017)

#15 - 05/18/2013 04:49 AM - nobu (Nobuyoshi Nakada)

=begin
You can use (({rb_funcall2()})) instead here.

- return rb_funcall(p->argv[0], rb_intern("sleep"), p->argc-1, p->argv+1);
- return rb_funcall(p->argv[0], rb_intern("sleep"), p->argc-1, p->argv[1]); /* (2) */
  =end

#16 - 05/22/2013 09:07 PM - Glass_saga (Masaki Matsushita)

- File patch2.diff added

2013/5/18 nobu (Nobuyoshi Nakada) nobu@ruby-lang.org

> You can use rb_funcall2() instead here.

I made a patch which use rb_funcall2().

diff from patch.diff to patch2.diff:

diff --git a/ext/thread/thread.c b/ext/thread/thread.c

index 33365c1..f2d17e9 100644
--- a/ext/thread/thread.c
+++ b/ext/thread/thread.c
@@ -142,15 +142,17 @@ rb_condvar_initialize(VALUE self)
}

struct sleep_call {

- int argc;
- VALUE *argv;

- VALUE mutex;
- VALUE timeout;
  };

+static ID id_sleep;
+
static VALUE
do_sleep(VALUE args)
{
struct sleep_call *p = (struct sleep_call *)args;

- return rb_funcall(p->argv[0], rb_intern("sleep"), p->argc-1, p->argv[1]);

- return rb_funcall2(p->mutex, id_sleep, 1, &p->timeout);
  }

static VALUE
@@ -173,12 +175,16 @@ static VALUE
rb_condvar_wait(int argc, VALUE *argv, VALUE self)
{
VALUE waiters = get_condvar_ptr(self)->waiters;

- VALUE mutex, timeout;
  struct sleep_call args;

- args.argc = argc;
- args.argv = argv;

- rb_scan_args(argc, argv, "11", &mutex, &timeout);

- args.mutex   = mutex;
- args.timeout = timeout;
  rb_ary_push(waiters, rb_thread_current());
  rb_ensure(do_sleep, (VALUE)&args, delete_current_thread, waiters);
- return self;
  }

@@ -607,7 +613,6 @@ static VALUE
szqueue_do_push(SizedQueue *szqueue, VALUE obj)
{
struct waiting_delete args;

- VALUE thread;

  while (queue_length(&szqueue->queue_) >= szqueue->max) {
  args.waiting = szqueue->queue_wait;
  @@ -699,6 +704,8 @@ Init_thread(void)
  VALUE rb_cQueue = DEFINE_CLASS_UNDER_THREAD(Queue, rb_cObject);
  VALUE rb_cSizedQueue = DEFINE_CLASS_UNDER_THREAD(SizedQueue, rb_cQueue);

- id_sleep = rb_intern("sleep");
- rb_define_alloc_func(rb_cConditionVariable, condvar_alloc);
  rb_define_method(rb_cConditionVariable, "initialize", rb_condvar_initialize, 0);
  rb_define_method(rb_cConditionVariable, "wait", rb_condvar_wait, -1);

#17 - 08/29/2013 06:54 PM - ko1 (Koichi Sasada)

- File thread.c added

I rewrite to use T_STRUCT instead of T_DATA (attached).

Matsushita-san:
Could you check it?

Issue:
With ext/thraed/thread.c instead of lib/thread.rb, we can't install ruby itself.

(1) rbinstall.rb is invoked with miniruby
(2) rbinstall.rb requires rubygems.rb
(3) rubygems.rb requires lib/rubygems/core_ext/kernel_require.rb
(4) lib/rubygems/core_ext/kernel_require.rb requires lib/monitor.rb
(5) lib/monitor.rb requires thread.rb
Problems are
(a) miniruby can't require extension libraries
(b) rbinstall.rb is not invoked with -I option

Easy way to solve these problem is use rbinstall.rb by ./ruby instead of ./miniruby with -I option.

Another solution is to embed thread.rb features (CV and Queue) as embeded classes.

#18 - 08/30/2013 02:23 AM - normalperson (Eric Wong)

"ko1 (Koichi Sasada)" redmine@ruby-lang.org wrote:

> Another solution is to embed thread.rb features (CV and Queue) as
> embeded classes.

I prefer this with no extra .so. Too many .so files hurts load time.
Just leave an empty thread.rb for compatibility.

#19 - 08/30/2013 02:53 PM - ko1 (Koichi Sasada)

(2013/08/30 12:14), SASADA Koichi wrote:

> A patch for this approach:
> http://www.atdot.net/sp/raw/m5xbsm

Note that last patch does not care about deadlock detection.

--
// SASADA Koichi at atdot dot net

#20 - 08/30/2013 03:29 PM - nobu (Nobuyoshi Nakada)

(13/08/29 18:54), ko1 (Koichi Sasada) wrote:

> Problems are
> (a) miniruby can't require extension libraries
> (b) rbinstall.rb is not invoked with -I option
>
> Easy way to solve these problem is use rbinstall.rb by ./ruby instead of ./miniruby with -I option.

Use $(RUNRUBY) in $(INSTRUBY) instead of $(MINIRUBY).

#21 - 08/30/2013 08:53 PM - nobu (Nobuyoshi Nakada)

(13/08/30 15:26), Nobuyoshi Nakada wrote:

> (13/08/29 18:54), ko1 (Koichi Sasada) wrote:
>
> > Problems are
> > (a) miniruby can't require extension libraries
> > (b) rbinstall.rb is not invoked with -I option
> >
> > Easy way to solve these problem is use rbinstall.rb by ./ruby instead of ./miniruby with -I option.
>
> Use $(RUNRUBY) in $(INSTRUBY) instead of $(MINIRUBY).

Since tool/rbinstall.rb replaces $:, it doesn't work as-is.

diff --git a/common.mk b/common.mk
index f295fb5..64ff5cc 100644
--- a/common.mk
+++ b/common.mk

```
@@ -123,7 +123,7 @@ SCRIPT_ARGS   = --dest-dir="$(DESTDIR)"
--make-flags="$(MAKEFLAGS)"
EXTMK_ARGS    = $(SCRIPT_ARGS) --extension $(EXTS) --extstatic $(EXTSTATIC)
--make-flags="V=$(V) MINIRUBY='$(MINIRUBY)'" --
-INSTRUBY      = $(SUDO) $(MINIRUBY) $(srcdir)/tool/rbinstall.rb
+INSTRUBY      = $(SUDO) $(RUNRUBY) -r./$(arch)-fake $(srcdir)/tool/rbinstall.rb
INSTRUBY_ARGS = $(SCRIPT_ARGS)
--data-mode=$(INSTALL_DATA_MODE)
--prog-mode=$(INSTALL_PROG_MODE)
@@ -449,7 +449,7 @@ post-no-install-doc::

CLEAR_INSTALLED_LIST = clear-installed-list

-install-prereq: $(CLEAR_INSTALLED_LIST) PHONY
+install-prereq: $(CLEAR_INSTALLED_LIST) yes-fake PHONY

clear-installed-list: PHONY
@> $(INSTALLED_LIST) set MAKE="$(MAKE)"
```

--
Nobu Nakada

#22 - 09/05/2013 08:52 PM - Glass_saga (Masaki Matsushita)

- File patch3.diff added

Sorry for my late response.
I fixed a bug in ext/thread.c ([ruby-core:56861]).
It was not compatible with Objects extended by Mutex_m and test/test_mutex_m.rb failed.
Moreover, I use rb_thread_sleep_deadly() to make it get along with deadlock detection.

Attached patch includes:

- updated ext/thread.c
- added test from [ruby-core:45950] to test/thread/test_queue.rb
- common.mk rewrited by Nakada-san

#23 - 09/05/2013 09:59 PM - ko1 (Koichi Sasada)

(2013/09/05 20:52), Glass_saga (Masaki Matsushita) wrote:

> I fixed a bug in ext/thread.c ([ruby-core:56861]).
> It was not compatible with Objects extended by Mutex_m and test/test_mutex_m.rb failed.
> Moreover, I use rb_thread_sleep_deadly() to make it get along with deadlock detection.
>
> Attached patch includes:
>
> - updated ext/thread.c
> - added test from [ruby-core:45950] to test/thread/test_queue.rb
> - common.mk rewrited by Nakada-san

Go ahead.

I'll make another ticket to embed Queue in srcdir/thread.c.

--
// SASADA Koichi at atdot dot net

#24 - 09/07/2013 12:15 AM - Anonymous

- Status changed from Assigned to Closed

- % Done changed from 0 to 100

This issue was solved with changeset r42862.
Ricardo, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

- common.mk: use RUNRUBY instead of MINIRUBY because MINIRUBY can't
  require extension libraries. The patch is from nobu
  (Nobuyoshi Nakada).

- ext/thread/extconf.rb: for build ext/thread/thread.c.
- include/ruby/intern.h: ditto.
- thread.c: ditto.
- lib/thread.rb: removed and replaced by ext/thread/thread.c.
- ext/thread/thread.c: Queue, SizedQueue and ConditionVariable implementations in C. This patch is based on patches from panaggio (Ricardo Panaggio) and funny_falcon (Yura Sokolov) and ko1 (Koichi Sasada). [ruby-core:31513] [Feature #3620]
- test/thread/test_queue.rb (test_queue_thread_raise): add a test for ensuring that killed thread should be removed from waiting threads. It is based on a code by ko1 (Koichi Sasada). [ruby-core:45950]

#25 - 09/13/2013 03:23 AM - kosaki (Motohiro KOSAKI)

(9/5/13 8:55 AM), SASADA Koichi wrote:

> (2013/09/05 20:52), Glass_saga (Masaki Matsushita) wrote:
>
> > I fixed a bug in ext/thread.c ([ruby-core:56861]).
> > It was not compatible with Objects extended by Mutex_m and test/test_mutex_m.rb failed.
> > Moreover, I use rb_thread_sleep_deadly() to make it get along with deadlock detection.
> >
> > Attached patch includes:
> >
> > - updated ext/thread.c
> > - added test from [ruby-core:45950] to test/thread/test_queue.rb
> > - common.mk rewrited by Nakada-san
>
> Go ahead.
>
> I'll make another ticket to embed Queue in srcdir/thread.c.

We already have.

http://bugs.ruby-lang.org/issues/7923

#26 - 09/18/2013 06:23 PM - ko1 (Koichi Sasada)

(2013/09/13 3:03), KOSAKI Motohiro wrote:

> We already have.
>
> http://bugs.ruby-lang.org/issues/7923

I feel "making trap safe Queue (Feature #7923)" and embedding Queue class is not same.

--
// SASADA Koichi at atdot dot net

Files

final-queue.patch	18 KB	07/28/2010	panaggio (Ricardo Panaggio)
final_queue_without_mutex.diff	16.7 KB	06/26/2012	funny_falcon (Yura Sokolov)
patch.diff	24.3 KB	05/17/2013	Glass_saga (Masaki Matsushita)
patch2.diff	24.4 KB	05/22/2013	Glass_saga (Masaki Matsushita)
thread.c	12.4 KB	08/29/2013	ko1 (Koichi Sasada)
patch3.diff	23 KB	09/05/2013	Glass_saga (Masaki Matsushita)