Ruby trunk - Bug #14013

[PATCH] Webrick 60172 fix

10/13/2017 06:30 PM - MSP-Greg (Greg L)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.5.0dev (2017-10-13 trunk 60176) [x64-mingw32]	**Backport:**	2.3: UNKNOWN, 2.4: UNKNOWN

Description

I was looking at the failure from 60172, and just tried changing some code. Low and behold, it passed all tests. But, since I'm not that familiar with 'nonblock' issues, I thought asking someone with more knowledge would be appropriate.

The patch also included a patch to TestNetHTTPS.test_verify, which was bypassed (not skipped) based on an ENV setting. I changed it to skip based on what I think is a sensible criteria.

I'm somewhat concerned about the test times under windows. Since this is a 'ruby only' patch, maybe someone can test under OSX or *nix?

After the patch, I have the following test results:

```
E:\GitHub\ruby\test>ruby -v --disable-gems runner.rb -I./lib -v --show-skip net/http/test_https.rb
ruby 2.5.0dev (2017-10-13 trunk 60176) [x64-mingw32]
Run options: -I./lib -v --show-skip

# Running tests:

[1/9] TestNetHTTPS#test_certificate_verify_failure = 1.08 s
[2/9] TestNetHTTPS#test_get = 1.14 s
[3/9] TestNetHTTPS#test_identity_verify_failure = 0.08 s
[4/9] TestNetHTTPS#test_post = 1.13 s
[5/9] TestNetHTTPS#test_session_reuse = 3.29 s
[6/9] TestNetHTTPS#test_session_reuse_but_expire = 2.25 s
[7/9] TestNetHTTPS#test_timeout_during_SSL_handshake = 0.05 s
[8/9] TestNetHTTPS#test_verify = 0.76 s
[9/9] TestNetHTTPS#test_verify_none = 1.14 s
Finished tests in 10.919400s, 0.8242 tests/s, 4.9453 assertions/s.
9 tests, 54 assertions, 0 failures, 0 errors, 0 skips


E:\GitHub\ruby\test>ruby -v --disable-gems runner.rb -I./lib -v --show-skip webrick/test_ssl_serve
r.rb
ruby 2.5.0dev (2017-10-13 trunk 60176) [x64-mingw32]
Run options: -I./lib -v --show-skip

# Running tests:

[1/3] TestWEBrickSSLServer#test_self_signed_cert_server = 0.14 s
[2/3] TestWEBrickSSLServer#test_self_signed_cert_server_with_string = 0.19 s
[3/3] TestWEBrickSSLServer#test_slow_connect = 0.30 s
Finished tests in 0.639600s, 4.6904 tests/s, 20.3252 assertions/s.
3 tests, 13 assertions, 0 failures, 0 errors, 0 skips
```

Thanks, Greg

Associated revisions

Revision 525ebb86 - 10/16/2017 04:33 AM - normal

webrick: fix up r60172

By making the socket non-blocking in r60172, TLS/SSL negotiation

via the SSL_accept function must handle non-blocking sockets
properly and retry on SSL_ERROR_WANT_READ/SSL_ERROR_WANT_WRITE.
OpenSSL::SSL::SSLSocket#accept cannot do that properly with a
non-blocking socket, so it must use non-blocking logic of
OpenSSL::SSL::SSLSocket#accept_nonblock.

Thanks to MSP-Greg (Greg L) for finding this.

- lib/webrick/server.rb (start_thread): use SSL_accept properly with non-blocking socket. [Bug #14013] [Bug #14005]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@60189 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 60189 - 10/16/2017 04:33 AM - normalperson (Eric Wong)

webrick: fix up r60172

By making the socket non-blocking in r60172, TLS/SSL negotiation
via the SSL_accept function must handle non-blocking sockets
properly and retry on SSL_ERROR_WANT_READ/SSL_ERROR_WANT_WRITE.
OpenSSL::SSL::SSLSocket#accept cannot do that properly with a
non-blocking socket, so it must use non-blocking logic of
OpenSSL::SSL::SSLSocket#accept_nonblock.

Thanks to MSP-Greg (Greg L) for finding this.

- lib/webrick/server.rb (start_thread): use SSL_accept properly with non-blocking socket. [Bug #14013] [Bug #14005]

Revision 60189 - 10/16/2017 04:33 AM - normal

webrick: fix up r60172

By making the socket non-blocking in r60172, TLS/SSL negotiation
via the SSL_accept function must handle non-blocking sockets
properly and retry on SSL_ERROR_WANT_READ/SSL_ERROR_WANT_WRITE.
OpenSSL::SSL::SSLSocket#accept cannot do that properly with a
non-blocking socket, so it must use non-blocking logic of
OpenSSL::SSL::SSLSocket#accept_nonblock.

Thanks to MSP-Greg (Greg L) for finding this.

- lib/webrick/server.rb (start_thread): use SSL_accept properly with non-blocking socket. [Bug #14013] [Bug #14005]

Revision 60189 - 10/16/2017 04:33 AM - normal

webrick: fix up r60172

By making the socket non-blocking in r60172, TLS/SSL negotiation
via the SSL_accept function must handle non-blocking sockets
properly and retry on SSL_ERROR_WANT_READ/SSL_ERROR_WANT_WRITE.
OpenSSL::SSL::SSLSocket#accept cannot do that properly with a
non-blocking socket, so it must use non-blocking logic of
OpenSSL::SSL::SSLSocket#accept_nonblock.

Thanks to MSP-Greg (Greg L) for finding this.

- lib/webrick/server.rb (start_thread): use SSL_accept properly with non-blocking socket. [Bug #14013] [Bug #14005]

Revision 3b1db7d3 - 10/18/2017 09:45 PM - normal

webrick: fix up r60172 and revert r60189

Thanks to MSP-Greg (Greg L) for helping with this.

- lib/webrick/server.rb (start_thread): ignore ECONNRESET, ECONNABORTED, EPROTO, and EINVAL on TLS negotiation errors the same way they were ignored before r60172 in the accept_client method of the main acceptor thread. [Bug #14013] [Bug #14005]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@60208 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 60208 - 10/18/2017 09:45 PM - normalperson (Eric Wong)

webrick: fix up r60172 and revert r60189

Thanks to MSP-Greg (Greg L) for helping with this.

- lib/webrick/server.rb (start_thread): ignore ECONNRESET, ECONNABORTED, EPROTO, and EINVAL on TLS negotiation errors the same way they were ignored before r60172 in the accept_client method of the main acceptor thread. [Bug #14013] [Bug #14005]

Revision 60208 - 10/18/2017 09:45 PM - normal

webrick: fix up r60172 and revert r60189

Thanks to MSP-Greg (Greg L) for helping with this.

- lib/webrick/server.rb (start_thread): ignore ECONNRESET, ECONNABORTED, EPROTO, and EINVAL on TLS negotiation errors the same way they were ignored before r60172 in the accept_client method of the main acceptor thread. [Bug #14013] [Bug #14005]

Revision 60208 - 10/18/2017 09:45 PM - normal

webrick: fix up r60172 and revert r60189

Thanks to MSP-Greg (Greg L) for helping with this.

- lib/webrick/server.rb (start_thread): ignore ECONNRESET, ECONNABORTED, EPROTO, and EINVAL on TLS negotiation errors the same way they were ignored before r60172 in the accept_client method of the main acceptor thread. [Bug #14013] [Bug #14005]

Revision 2e728d51 - 12/14/2017 01:31 PM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 60123,60172,60189,60208,60210,60211: [Backport #14005]

```
webrick: avoid unnecessary IO#sync= call

Sockets and pipes are always created with FMODE_SYNC flag
already set (otherwise many things would be broken).

* lib/webrick/server.rb (accept_client): remove unnecessary
  IO#sync= call

webrick: do not hang acceptor on slow TLS connections

OpenSSL::SSL::SSLSocket#accept may block indefinitely on clients
which negotiate the TCP connection, but fail (or are slow) to
negotiate the subsequent TLS handshake.  This prevents the
multi-threaded WEBrick server from accepting other connections.

Since the TLS handshake (via OpenSSL::SSL::SSLSocket#accept)
consists of normal read/write traffic over TCP, handle it in the
per-client thread, instead.

Furthermore, using non-blocking accept() is useful for non-TLS
sockets anyways because spurious wakeups are possible from
select(2).

* lib/webrick/server.rb (accept_client): use TCPServer#accept_nonblock
  and remove OpenSSL::SSL::SSLSocket#accept call
* lib/webrick/server.rb (start_thread): call OpenSSL::SSL::SSLSocket#accept
* test/webrick/test_ssl_server.rb (test_slow_connect): new test
  [ruby-core:83221] [Bug #14005]

webrick: fix up r60172

By making the socket non-blocking in r60172, TLS/SSL negotiation
via the SSL_accept function must handle non-blocking sockets
properly and retry on SSL_ERROR_WANT_READ/SSL_ERROR_WANT_WRITE.
OpenSSL::SSL::SSLSocket#accept cannot do that properly with a
non-blocking socket, so it must use non-blocking logic of
OpenSSL::SSL::SSLSocket#accept_nonblock.

Thanks to MSP-Greg (Greg L) for finding this.

* lib/webrick/server.rb (start_thread): use SSL_accept properly
  with non-blocking socket.
  [Bug #14013] [Bug #14005]

webrick: fix up r60172 and revert r60189

Thanks to MSP-Greg (Greg L) for helping with this.

* lib/webrick/server.rb (start_thread): ignore ECONNRESET, ECONNABORTED,
  EPROTO, and EINVAL on TLS negotiation errors the same way they
  were ignored before r60172 in the accept_client method of the
  main acceptor thread.
  [Bug #14013] [Bug #14005]

webrick: fix up r60172 and r60208

Thanks to MSP-Greg (Greg L) for helping with this.

* lib/webrick/server.rb (start_thread): fix non-local return
  introduced in r60208

webrick: fix up r60172 and r60210

Thanks to MSP-Greg (Greg L) for helping with this.

* lib/webrick/server.rb (start_thread): properly fix non-local return
  introduced in r60208 and r60210
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_4@61239 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 61239 - 12/14/2017 01:31 PM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 60123,60172,60189,60208,60210,60211: [Backport #14005]

```
webrick: avoid unnecessary IO#sync= call

Sockets and pipes are always created with FMODE_SYNC flag
already set (otherwise many things would be broken).

* lib/webrick/server.rb (accept_client): remove unnecessary
  IO#sync= call

webrick: do not hang acceptor on slow TLS connections

OpenSSL::SSL::SSLSocket#accept may block indefinitely on clients
which negotiate the TCP connection, but fail (or are slow) to
negotiate the subsequent TLS handshake.  This prevents the
multi-threaded WEBrick server from accepting other connections.

Since the TLS handshake (via OpenSSL::SSL::SSLSocket#accept)
consists of normal read/write traffic over TCP, handle it in the
per-client thread, instead.

Furthermore, using non-blocking accept() is useful for non-TLS
sockets anyways because spurious wakeups are possible from
select(2).

* lib/webrick/server.rb (accept_client): use TCPServer#accept_nonblock
  and remove OpenSSL::SSL::SSLSocket#accept call
* lib/webrick/server.rb (start_thread): call OpenSSL::SSL::SSLSocket#accept
* test/webrick/test_ssl_server.rb (test_slow_connect): new test
  [ruby-core:83221] [Bug #14005]

webrick: fix up r60172

By making the socket non-blocking in r60172, TLS/SSL negotiation
via the SSL_accept function must handle non-blocking sockets
properly and retry on SSL_ERROR_WANT_READ/SSL_ERROR_WANT_WRITE.
OpenSSL::SSL::SSLSocket#accept cannot do that properly with a
non-blocking socket, so it must use non-blocking logic of
OpenSSL::SSL::SSLSocket#accept_nonblock.

Thanks to MSP-Greg (Greg L) for finding this.

* lib/webrick/server.rb (start_thread): use SSL_accept properly
  with non-blocking socket.
  [Bug #14013] [Bug #14005]

webrick: fix up r60172 and revert r60189

Thanks to MSP-Greg (Greg L) for helping with this.

* lib/webrick/server.rb (start_thread): ignore ECONNRESET, ECONNABORTED,
  EPROTO, and EINVAL on TLS negotiation errors the same way they
  were ignored before r60172 in the accept_client method of the
  main acceptor thread.
  [Bug #14013] [Bug #14005]

webrick: fix up r60172 and r60208

Thanks to MSP-Greg (Greg L) for helping with this.

* lib/webrick/server.rb (start_thread): fix non-local return
  introduced in r60208

webrick: fix up r60172 and r60210

Thanks to MSP-Greg (Greg L) for helping with this.

* lib/webrick/server.rb (start_thread): properly fix non-local return
  introduced in r60208 and r60210
```

Revision 1beda297 - 12/14/2017 01:33 PM - usa (Usaku NAKAMURA)

merge revision(s) 60172,60189,60208,60210,60211: [Backport #14005]

```
webrick: do not hang acceptor on slow TLS connections

OpenSSL::SSL::SSLSocket#accept may block indefinitely on clients
which negotiate the TCP connection, but fail (or are slow) to
negotiate the subsequent TLS handshake.  This prevents the
multi-threaded WEBrick server from accepting other connections.

Since the TLS handshake (via OpenSSL::SSL::SSLSocket#accept)
consists of normal read/write traffic over TCP, handle it in the
per-client thread, instead.

Furthermore, using non-blocking accept() is useful for non-TLS
sockets anyways because spurious wakeups are possible from
select(2).

* lib/webrick/server.rb (accept_client): use TCPServer#accept_nonblock
  and remove OpenSSL::SSL::SSLSocket#accept call
* lib/webrick/server.rb (start_thread): call OpenSSL::SSL::SSLSocket#accept
* test/webrick/test_ssl_server.rb (test_slow_connect): new test
  [ruby-core:83221] [Bug #14005]

webrick: fix up r60172

By making the socket non-blocking in r60172, TLS/SSL negotiation
via the SSL_accept function must handle non-blocking sockets
properly and retry on SSL_ERROR_WANT_READ/SSL_ERROR_WANT_WRITE.
OpenSSL::SSL::SSLSocket#accept cannot do that properly with a
non-blocking socket, so it must use non-blocking logic of
OpenSSL::SSL::SSLSocket#accept_nonblock.

Thanks to MSP-Greg (Greg L) for finding this.

* lib/webrick/server.rb (start_thread): use SSL_accept properly
  with non-blocking socket.
  [Bug #14013] [Bug #14005]

webrick: fix up r60172 and revert r60189

Thanks to MSP-Greg (Greg L) for helping with this.

* lib/webrick/server.rb (start_thread): ignore ECONNRESET, ECONNABORTED,
  EPROTO, and EINVAL on TLS negotiation errors the same way they
  were ignored before r60172 in the accept_client method of the
  main acceptor thread.
  [Bug #14013] [Bug #14005]

webrick: fix up r60172 and r60208

Thanks to MSP-Greg (Greg L) for helping with this.

* lib/webrick/server.rb (start_thread): fix non-local return
  introduced in r60208

webrick: fix up r60172 and r60210
```

```
Thanks to MSP-Greg (Greg L) for helping with this.

* lib/webrick/server.rb (start_thread): properly fix non-local return
  introduced in r60208 and r60210
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_3@61240 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 61240 - 12/14/2017 01:33 PM - usa (Usaku NAKAMURA)

merge revision(s) 60172,60189,60208,60210,60211: [Backport #14005]

```
webrick: do not hang acceptor on slow TLS connections

OpenSSL::SSL::SSLSocket#accept may block indefinitely on clients
which negotiate the TCP connection, but fail (or are slow) to
negotiate the subsequent TLS handshake.  This prevents the
multi-threaded WEBrick server from accepting other connections.

Since the TLS handshake (via OpenSSL::SSL::SSLSocket#accept)
consists of normal read/write traffic over TCP, handle it in the
per-client thread, instead.

Furthermore, using non-blocking accept() is useful for non-TLS
sockets anyways because spurious wakeups are possible from
select(2).

* lib/webrick/server.rb (accept_client): use TCPServer#accept_nonblock
  and remove OpenSSL::SSL::SSLSocket#accept call
* lib/webrick/server.rb (start_thread): call OpenSSL::SSL::SSLSocket#accept
* test/webrick/test_ssl_server.rb (test_slow_connect): new test
  [ruby-core:83221] [Bug #14005]

webrick: fix up r60172

By making the socket non-blocking in r60172, TLS/SSL negotiation
via the SSL_accept function must handle non-blocking sockets
properly and retry on SSL_ERROR_WANT_READ/SSL_ERROR_WANT_WRITE.
OpenSSL::SSL::SSLSocket#accept cannot do that properly with a
non-blocking socket, so it must use non-blocking logic of
OpenSSL::SSL::SSLSocket#accept_nonblock.

Thanks to MSP-Greg (Greg L) for finding this.

* lib/webrick/server.rb (start_thread): use SSL_accept properly
  with non-blocking socket.
  [Bug #14013] [Bug #14005]

webrick: fix up r60172 and revert r60189

Thanks to MSP-Greg (Greg L) for helping with this.

* lib/webrick/server.rb (start_thread): ignore ECONNRESET, ECONNABORTED,
  EPROTO, and EINVAL on TLS negotiation errors the same way they
  were ignored before r60172 in the accept_client method of the
  main acceptor thread.
  [Bug #14013] [Bug #14005]

webrick: fix up r60172 and r60208

Thanks to MSP-Greg (Greg L) for helping with this.

* lib/webrick/server.rb (start_thread): fix non-local return
  introduced in r60208

webrick: fix up r60172 and r60210

Thanks to MSP-Greg (Greg L) for helping with this.

* lib/webrick/server.rb (start_thread): properly fix non-local return
  introduced in r60208 and r60210
```

History

#1 - 10/13/2017 06:31 PM - MSP-Greg (Greg L)

- ruby -v set to ruby 2.5.0dev (2017-10-13 trunk 60176) [x64-mingw32]

#2 - 10/14/2017 03:34 AM - MSP-Greg (Greg L)

- File webrick_60172_fix.patch added

Original patch file did not allow for cert file values of nil. Corrected in attached version. Patch was used on most recent Appveyor build, all tests passed.

#3 - 10/16/2017 04:34 AM - Anonymous

- Status changed from Open to Closed

Applied in changeset trunk|r60189.

---

webrick: fix up r60172

By making the socket non-blocking in r60172, TLS/SSL negotiation
via the SSL_accept function must handle non-blocking sockets
properly and retry on SSL_ERROR_WANT_READ/SSL_ERROR_WANT_WRITE.
OpenSSL::SSL::SSLSocket#accept cannot do that properly with a
non-blocking socket, so it must use non-blocking logic of
OpenSSL::SSL::SSLSocket#accept_nonblock.

Thanks to MSP-Greg (Greg L) for finding this.

- lib/webrick/server.rb (start_thread): use SSL_accept properly with non-blocking socket. [Bug #14013] [Bug #14005]

#4 - 10/16/2017 04:41 AM - normalperson (Eric Wong)

Greg.mpls@gmail.com wrote:

> Issue #14013 has been updated by MSP-Greg (Greg L).
>
> File webrick_60172_fix.patch added

Thanks, I based r60189 on your patch to webrick/server.rb

However, I'm unsure about the test/net/http/test_https.rb
test_verify change since it requires Internet access and the
test suite should be able to run offline. I am also not
maintainer of net/http, only webrick.   Thanks again.

#5 - 10/16/2017 03:23 PM - MSP-Greg (Greg L)

Eric,

Thank you for looking at this, along with r60189. It seems that the Appveyor mswin builds are still failing on TestNetHTTPS#test_certificate_verify_failure.

Since many contributors are not using (or familiar with) Windows, I created a GitHub repo MSP-Greg/trunk_mingw_testing. It allows one to easily add patch files, then run tests against the latest mingw trunk build from ruby-loco. Since it's using the trunk build, it only works with *.rb changes.

I haven't created a README yet, but one could fork it and easily test *.rb code changes.

Using it, I created a patch using r60189, and I also had the failure in TestNetHTTPS#test_certificate_verify_failure. See here, log below:

```
ruby -v --disable-gems runner.rb -v -I./lib --show-skip net/http/test_https.rb
ruby 2.5.0dev (2017-10-16 trunk 60190) [x64-mingw32]
Run options: -v -I./lib --show-skip

# Running tests:
[1/8] TestNetHTTPS#test_certificate_verify_failure = 1.06 s
[2/8] TestNetHTTPS#test_get = 1.12 s
[3/8] TestNetHTTPS#test_identity_verify_failure = 0.07 s
[4/8] TestNetHTTPS#test_post = 1.12 s
[5/8] TestNetHTTPS#test_session_reuse = 3.30 s
[6/8] TestNetHTTPS#test_session_reuse_but_expire = 2.23 s
[7/8] TestNetHTTPS#test_timeout_during_SSL_handshake = 0.04 s
[8/8] TestNetHTTPS#test_verify_none = 1.14 s

  1) Failure:
TestNetHTTPS#test_certificate_verify_failure [C:/ruby/test/net/http/utils.rb:48]:
<[]> expected but was
```

```
<["[2017-10-16 13:59:08] ERROR Errno::ECONNRESET: An existing connection was forcibly closed by the remote hos
t. - SSL_accept\n" +
 "\tC:/ruby_trunk/lib/ruby/2.5.0/webrick/server.rb:301:in `accept_nonblock'\n" +
 "\tC:/ruby_trunk/lib/ruby/2.5.0/webrick/server.rb:301:in `block (2 levels) in start_thread'\n" +
 "\tC:/ruby_trunk/lib/ruby/2.5.0/webrick/utils.rb:263:in `timeout'\n" +
 "\tC:/ruby_trunk/lib/ruby/2.5.0/webrick/server.rb:297:in `block in start_thread'\n"]>.

Finished tests in 10.146761s, 0.7884 tests/s, 4.8291 assertions/s.
8 tests, 49 assertions, 1 failures, 0 errors, 0 skips
```

You're much more familiar with server code and sockets than I am, so if you could have another look at it, I'd appreciate it. I'd be happy to test anything...

#6 - 10/16/2017 08:21 PM - normalperson (Eric Wong)

Odd, so using IO#wait_*able methods doesn't work for you, but
IO.select does? Can you try the following patch?

It's basically your patch with "writable" spelled correctly:

```
 diff --git a/lib/webrick/server.rb b/lib/webrick/server.rb
 index 2d678273e5..25d507b67d 100644
 --- a/lib/webrick/server.rb
 +++ b/lib/webrick/server.rb
 @@ -299,8 +299,10 @@ def start_thread(sock, &block)
  # we must call OpenSSL::SSL::SSLSocket#accept_nonblock until
  # it stop returning wait_* symbols:
 case ret = sock.accept_nonblock(exception: false)
 -              when :wait_readable, :wait_writable
 -                sock.to_io.__send__(ret)
 +              when :wait_readable
 +                IO.select([sock])
 +              when :wait_writable
 +                IO.select(nil, [sock])
 else
 break
 end while true
```

I guess your original didn't loop on "while true", either, but
it really should (to properly finish the negotiation). There's
also no need to call .to_io with IO.select, as it's already done
transparently.

(The reason I favor IO#wait_readable and IO#wait_writable is
it is optimized under Linux)

I won't be using proprietary services like GitHub, so I'll let
you handle the testing. Can you repeat this failure on a local
machine or using something like ssh?

#7 - 10/17/2017 03:49 PM - MSP-Greg (Greg L)

- File webrick.patch added

Eric, thanks.

I was working some read_nonblock code at the same time...

> There's also no need to call .to_io with IO.select, as it's already done transparently.

Aware of that, it was used in several ruby doc examples...

Anyway, since you mentioned 'optimized under Linux', I worked with that; it seems that on Windows, after the select or sock.to_io.__send__(ret) calls complete, it cannot have sock.accept_nonblock(exception: false) called again. Hence, I added the additional break, but only on windows. Passes locally and on Appveyor CI.

Patch attached and below:

```
diff --git a/lib/webrick/server.rb b/lib/webrick/server.rb
index 2d678273e5..57ffe5a48b 100644
--- a/lib/webrick/server.rb
+++ b/lib/webrick/server.rb
@@ -295,12 +295,14 @@ def start_thread(sock, &block)
           end
           if sock.respond_to?(:sync_close=) && @config[:SSLStartImmediately]
             WEBrick::Utils.timeout(@config[:RequestTimeout]) do
-
-               # we must call OpenSSL::SSL::SSLSocket#accept_nonblock until
-               # it stop returning wait_* symbols:
+
+               # we must call OpenSSL::SSL::SSLSocket#accept_nonblock until it
+               # stops returning wait_* symbols
+               # accept_nonblock can only be called once on Windows
               case ret = sock.accept_nonblock(exception: false)
               when :wait_readable, :wait_writable
                 sock.to_io.__send__(ret)
+                break if /mingw|mswin/ =~ RUBY_PLATFORM
               else
                 break
               end while true
```

#8 - 10/17/2017 04:32 PM - normalperson (Eric Wong)

Greg.mpls@gmail.com wrote:

> diff --git a/lib/webrick/server.rb b/lib/webrick/server.rb
> index 2d678273e5..57ffe5a48b 100644
> --- a/lib/webrick/server.rb
> +++ b/lib/webrick/server.rb
> @@ -295,12 +295,14 @@ def start_thread(sock, &block)
> end
> if sock.respond_to?(:sync_close=) && @config[:SSLStartImmediately]
>
> ## WEBrick::Utils.timeout(@config[:RequestTimeout]) do
>
> - # we must call OpenSSL::SSL::SSLSocket#accept_nonblock until
> - # it stop returning wait_* symbols:
>
> - # we must call OpenSSL::SSL::SSLSocket#accept_nonblock until it
> - # stops returning wait_* symbols
> - # accept_nonblock can only be called once on Windows

That is bizarre and this is not the right place for
platform-specific code. There's bound to be other places where
SSLSocket#accept_nonblock is used like this...

Also, does test/openssl/test_pair.rb work for you? Does it
still work when the IO.select calls in test_connect_accept_nonblock_no_exception
are replaced with corresponding IO#wait_*able calls?

(I guess it's desirable to minimize dependencies in those tests,
which is why IO.select is used instead of IO#wait_*able.

Thanks.

>```
>           case ret = sock.accept_nonblock(exception: false)
>           when :wait_readable, :wait_writable
>             sock.to_io.__send__(ret)
>```
>
> - break if /mingw|mswin/ =~ RUBY_PLATFORM   else   break   end while true

#9 - 10/18/2017 03:09 AM - MSP-Greg (Greg L)

normalperson (Eric Wong) wrote:

> Also, does test/openssl/test_pair.rb work for you?

Yes.

> Does it still work when the IO.select calls in test_connect_accept_nonblock_no_exception are replaced with corresponding IO#wait_*able calls?

Yes. Changes (needed #to_io), starting at line 401:

```
    th = Thread.new do
      rets = []
      begin
        rv = s1.connect_nonblock(exception: false)
        rets << rv
        case rv
        when :wait_writable
          s1.to_io.wait_writable         # IO.select(nil, [s1], nil, 5)
        when :wait_readable
          s1.to_io.wait_readable         # IO.select([s1], nil, nil, 5)
        end
      end until rv == s1
      rets
    end
end
```

#10 - 10/18/2017 03:51 AM - MSP-Greg (Greg L)

Eric,

I'm about to 'go offline', but the following fails when replacing the case statement group

```
ret = sock.accept_nonblock(exception: false)
if ret == :wait_readable || ret == :wait_writable
  sock.to_io.__send__(ret)
  sock.accept_nonblock(exception: false)
end
```

If I comment out the 2nd sock.accept_nonblock(exception: false) statement, it passes.

Also, similar to the code in test_pair.rb, the following works:

```
begin
  ret = sock.accept_nonblock(exception: false)
  if ret == :wait_readable || ret == :wait_writable
    sock.to_io.__send__(ret)
    break if /mingw|mswin/ =~ RUBY_PLATFORM  # or ret = sock instead of break
  end
end until ret == sock
```

Being a windows type, I have no idea what's required for Linux/OSX...

Thanks again for your work (this and all the rest)

#11 - 10/18/2017 07:29 PM - MSP-Greg (Greg L)

- File webrick_ssl.patch added

I posted GitHub PR #1718, which passed both Travis & Appveyor. It also passes on my local MinGW trunk build

Rather than an OS check, it checks to see if the #wait_* methods return nil.
Below is patch, attached also.

```
diff --git a/lib/webrick/server.rb b/lib/webrick/server.rb
index 2d678273e5..2ece008a01 100644
--- a/lib/webrick/server.rb
+++ b/lib/webrick/server.rb
@@ -297,13 +297,13 @@ def start_thread(sock, &block)
             WEBrick::Utils.timeout(@config[:RequestTimeout]) do

               # we must call OpenSSL::SSL::SSLSocket#accept_nonblock until
-              # it stop returning wait_* symbols:
-              case ret = sock.accept_nonblock(exception: false)
-              when :wait_readable, :wait_writable
-                sock.to_io.__send__(ret)
-              else
-                break
-              end while true
+              # it stop returning wait_* symbols or wait_* methods return !nil:
+              begin
+                ret = sock.accept_nonblock(exception: false)
+                if ret == :wait_readable || ret == :wait_writable
+                  break unless sock.to_io.__send__(ret).nil?
+                end
+              end until ret == sock
```

```
          end
        end
        call_callback(:AcceptCallback, sock)
```

Thanks for all your help & suggestions.
Greg

#12 - 10/18/2017 08:51 PM - normalperson (Eric Wong)

Greg.mpls@gmail.com wrote:

> Eric wrote:
>
> > Also, does test/openssl/test_pair.rb work for you?

Oops, I mean how accept_nonblock worked in that test. In other
words, can you try this to dump all the outputs?

diff --git a/test/openssl/test_pair.rb b/test/openssl/test_pair.rb
index 55b62321b8..0168164810 100644
--- a/test/openssl/test_pair.rb
+++ b/test/openssl/test_pair.rb
@@ -413,10 +413,13 @@ def test_connect_accept_nonblock_no_exception
rets
end

- rets = [] until th.join(0.01) accepted = s2.accept_nonblock(exception: false) assert_include([s2, :wait_readable, :wait_writable], accepted)
- rets << accepted end
- warn rets.inspect

rets = th.value
assert_instance_of Array, rets

#13 - 10/18/2017 09:02 PM - normalperson (Eric Wong)

Greg.mpls@gmail.com wrote:

> Issue #14013 has been updated by MSP-Greg (Greg L).
>
> File webrick_ssl.patch added
>
> I posted GitHub PR #1718, which passed both Travis & Appveyor. It also passes on my local MinGW trunk build
>
> Rather than an OS check, it checks to see if the #wait_* methods return nil.
> Below is patch, attached also.

Thanks. I'm somewhat inclined to accept it because it solves
your problem; but the troubling thing is I don't understand why
it is necessary....

> ```
>          WEBrick::Utils.timeout(@config[:RequestTimeout]) do
> ```
>
> - # it stop returning wait_* symbols or wait_* methods return !nil:
> - begin
> - ret = sock.accept_nonblock(exception: false)
> - if ret == :wait_readable || ret == :wait_writable
> - break unless sock.to_io.**send**(ret).nil?

IO#wait_readable and IO#wait_writable only return nil if a
timeout is specified, so I'm not sure how it could ever return nil.

In other words, we might as well not be looping at all. And
reading the implementation of ossl_start_ssl (in
ext/openss/ossl_ssl.c) more carefully, it seems my initial use
of SSLSocket#accept was entirely sufficient to handle blocking
sockets.

I'm actually suspecting there's a bug in the openssl extension
around error handling for Windows, since there is a
win32-specific errno path for ssl_get_error in ext/openssl/ossl_ssl.c

- end
- end until ret == sock

SSLSocket#accept_nonblock clearly documents ret == sock is the
expected and eventual outcome of a successful negotiation, so
that is fine.

```
          end
        end
        call_callback(:AcceptCallback, sock)
```

```
Thanks for all your help & suggestions.
Greg
```

#14 - 10/18/2017 09:51 PM - normalperson (Eric Wong)

Eric Wong normalperson@yhbt.net wrote:

> Greg.mpls@gmail.com wrote:
>
> > Issue #14013 has been updated by MSP-Greg (Greg L).
> >
> > File webrick_ssl.patch added
> >
> > I posted GitHub PR #1718, which passed both Travis & Appveyor. It also passes on my local MinGW trunk build
> >
> > Rather than an OS check, it checks to see if the #wait_* methods return nil.
> > Below is patch, attached also.
>
> Thanks. I'm somewhat inclined to accept it because it solves
> your problem; but the troubling thing is I don't understand why
> it is necessary....

Nope; because this fails under Linux. Actually, I now understand the problem:
ECONNRESET (and other errors) are not being discarded as they were
in accept_client.

[1/8] TestNetHTTPS#test_certificate_verify_failure = 0.03 s
1) Failure:
TestNetHTTPS#test_certificate_verify_failure [/path/to/ruby/test/net/http/test_https.rb:156]:
expected but was
.

Which caused me to finally notice this comment in that test:

unless /mswin|mingw/ =~ RUBY_PLATFORM
# on Windows, Errno::ECONNRESET will be raised, and it'll be eaten by
WEBrick

So I made r60208; which should really fix the problem on all
platforms and not pollute logs when bad clients connect.

#15 - 10/18/2017 10:59 PM - MSP-Greg (Greg L)

Eric,

Bad news. Appveyor just failed with:

```
  1) Failure:
TestNetHTTPS#test_certificate_verify_failure [C:/projects/ruby/test/net/http/utils.rb:48]:
<[]> expected but was
<["[2017-10-18 22:01:37] ERROR LocalJumpError: unexpected return\n" +
 "\tC:/projects/ruby/lib/webrick/server.rb:302:in `rescue in block (2 levels) in start_thread'\n" +
 "\tC:/projects/ruby/lib/webrick/server.rb:298:in `block (2 levels) in start_thread'\n" +
 "\tC:/projects/ruby/lib/webrick/utils.rb:263:in `timeout'\n" +
 "\tC:/projects/ruby/lib/webrick/server.rb:297:in `block in start_thread'\n"]>.
Finished tests in 495.765993s, 33.2657 tests/s, 4417.9997 assertions/s.
16492 tests, 2190294 assertions, 1 failures, 0 errors, 308 skips
ruby -v: ruby 2.5.0dev (2017-10-19) [x64-mswin64_120]
```

Re the patch you included above, can you please bracket it with triple backticks (```), with the first being ```diff ? May need a blank line before the first set. I view this on the web; the markdown parser makes a mess of diff listings without the backticks.

Re the #nil?, the docs/comments do state returns 'nil on timeout'. Maybe we should remove the WEBrick::Utils.timeout block and put the timeout value in the #wait_* methods?

I'm not sure what to do. The patch I listed did pass Travis, but it fails for you on Linux... Thanks, Greg

#16 - 10/18/2017 11:31 PM - normalperson (Eric Wong)

```
Greg.mpls@gmail.com wrote:
> Bad news.  Appveyor just failed with:

Oops, try r60210; I moved the return outside of the timeout
block.

> Re the patch you included above, can you please bracket it
> with triple backticks (\`\`\`), with the first being
> \`\`\`diff ?  May need a blank line before the first set.  I
> view this on the web; the markdown parser makes a mess of diff
> listings without the backticks.

I'll try to remember in the future.  I remember there was a time
when redmine didn't try to assume emails were markdown and
always treaded them as preformatted (I either use email or
w3m, so no variable-width fonts).

> Re the `#nil?`, the docs/comments do state returns 'nil on
> timeout'.  Maybe we should remove the `WEBrick::Utils.timeout`
> block and put the timeout value in the `#wait_*` methods?

I've considered it, but it requires accounting for total time
spent in case we need to call accept_nonblock multiple times;
so it's extra math + logic which I don't want...


I actually want to improve 'timeout' in stdlib and implement it
in the core VM so it affects all methods like IO.select and
IO#wait*able, and even IO#read transparently.
```

#17 - 10/18/2017 11:41 PM - normalperson (Eric Wong)

```
Eric Wong <normalperson@yhbt.net> wrote:
> Oops, try r60210; I moved the return outside of the timeout
> block.

Erm, make that r60211 :x
```

#18 - 10/19/2017 12:05 AM - MSP-Greg (Greg L)

Eric,

Thanks. I tested 60211. My local build passed, Travis passed, and Appveyor passed.

Now maybe I can finally write the code to run a Webrick SSL server locally. I normally use Puma or Thin...

> I've considered it, but it requires accounting for total time spent in case we need to call accept_nonblock multiple times; so it's extra math + logic which I don't want...

I'm a math type, but I realized that also...

If you ever have any code you need tested on Windows, feel free to contact me.

Thanks again, Greg

#19 - 10/19/2017 01:02 AM - normalperson (Eric Wong)

Greg.mpls@gmail.com wrote:

> Thanks. I tested 60211. My local build passed, Travis passed, and Appveyor passed.

Thanks for your help with testing this!

> Now maybe I can finally write the code to run a Webrick SSL server locally. I normally use Puma or Thin...
>
>> I've considered it, but it requires accounting for total time spent in case we need to call accept_nonblock multiple times; so it's extra math + logic which I don't want...
>
> I'm a math type, but I realized that also...
>
> If you ever have any code you need tested on Windows, feel free to contact me.

Sure, at least off the top of my head, there's at least
auto-Fiber aka Thriber:

`https://bugs.ruby-lang.org/issues/13618`

and more GVL release points for file.c and dir.c. Most of the
current work pretty naively done but future optimizations may be
more aggressive and I am likely to introduce bugs in dir.c
around UTF-8 path name normalization:

`https://bugs.ruby-lang.org/issues/13996`

Files

webrick_60172_fix.patch	1.8 KB	10/13/2017	MSP-Greg (Greg L)
webrick_60172_fix.patch	1.81 KB	10/14/2017	MSP-Greg (Greg L)
webrick.patch	990 Bytes	10/17/2017	MSP-Greg (Greg L)
webrick_ssl.patch	1.04 KB	10/18/2017	MSP-Greg (Greg L)