ruby / **ruby**

Comparing changes

Choose two branches to see what’s changed or to start a new pull request. If you need to, you can also compare across forks.

base: v2_4_2 ▾ ← compare: v2_4_3 ▾

8 commits | **16** files changed | **0** commit comments | **1** contributor

Commits on Sep 15, 2017

nagachika	`bump up teeny version to 2.4.3.` …	✓ ac74cb6
nagachika	`merge revision(s) 58254: [Backport #13402]` …	✓ edda792

Commits on Oct 11, 2017

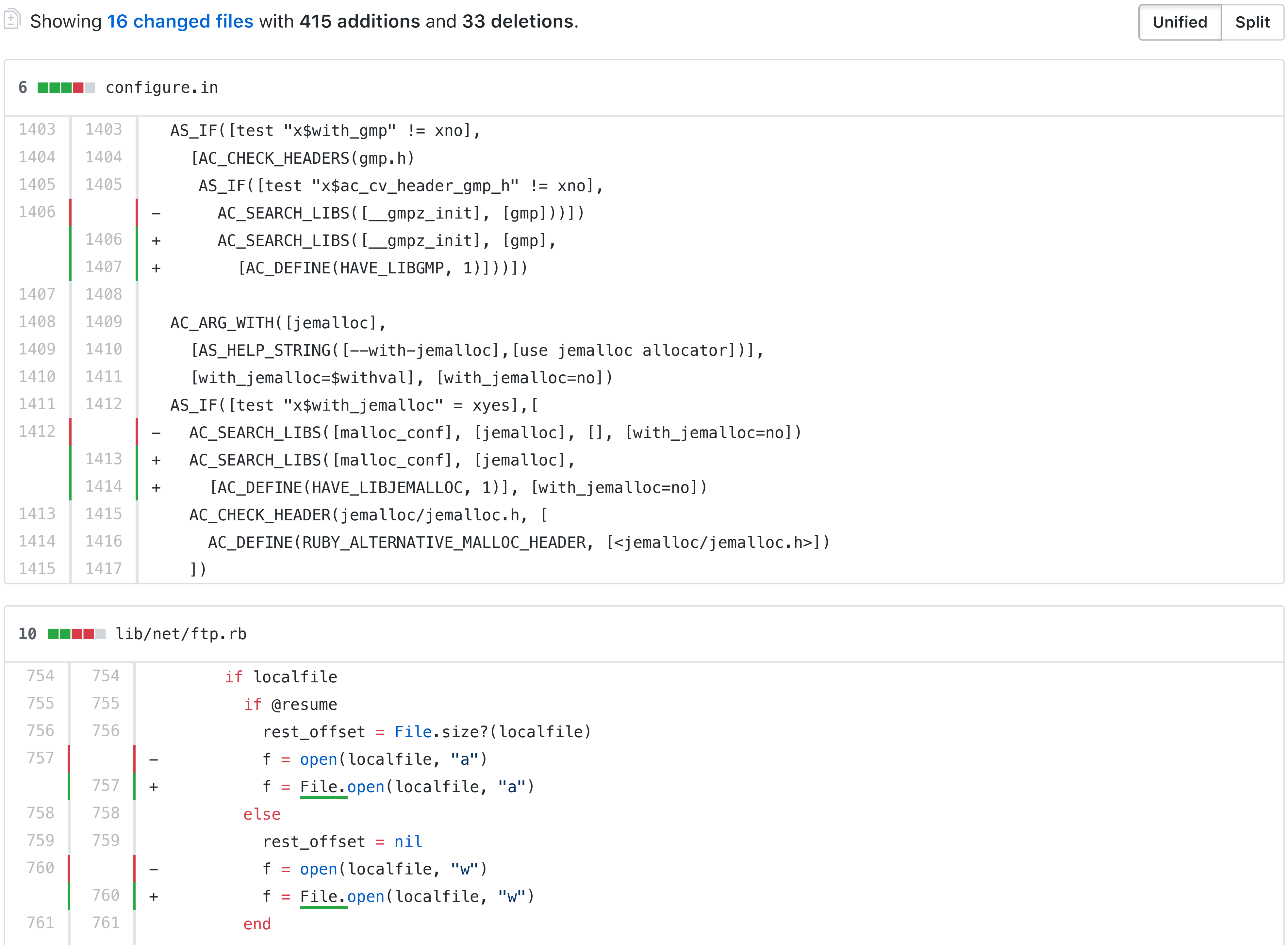
nagachika	`merge revision(s) 60149: [Backport #14003]` …	✗ 1281e56

Commits on Nov 03, 2017

nagachika	`merge revision(s) 58499,58500: [Backport #13181]` …	✓ 86bfcc2

Commits on Dec 14, 2017

nagachika	`merge revision(s) 61197: [Backport #14184]` …	✓ 096db35
nagachika	`merge revision(s) 60123,60172,60189,60208,60210,60211: [Backport #14005]` …	✓ 2e728d5
nagachika	`merge revision(s) 61242: [Backport #14185]` …	✓ 95645f5
nagachika	`add tag v2_4_3` …	✓ a5ec07c



Showing 16 changed files with 415 additions and 33 deletions.

6 🟩🟩🟩🟥⬜ configure.in

```
 AS_IF([test "x$with_gmp" != xno],
   [AC_CHECK_HEADERS(gmp.h)
    AS_IF([test "x$ac_cv_header_gmp_h" != xno],
-      AC_SEARCH_LIBS([__gmpz_init], [gmp]))])
+      AC_SEARCH_LIBS([__gmpz_init], [gmp],
+        [AC_DEFINE(HAVE_LIBGMP, 1)]))])
 
 AC_ARG_WITH([jemalloc],
   [AS_HELP_STRING([--with-jemalloc],[use jemalloc allocator])],
   [with_jemalloc=$withval], [with_jemalloc=no])
 AS_IF([test "x$with_jemalloc" = xyes],[
-  AC_SEARCH_LIBS([malloc_conf], [jemalloc], [], [with_jemalloc=no])
+  AC_SEARCH_LIBS([malloc_conf], [jemalloc],
+    [AC_DEFINE(HAVE_LIBJEMALLOC, 1)], [with_jemalloc=no])
   AC_CHECK_HEADER(jemalloc/jemalloc.h, [
     AC_DEFINE(RUBY_ALTERNATIVE_MALLOC_HEADER, [<jemalloc/jemalloc.h>])
   ])
```

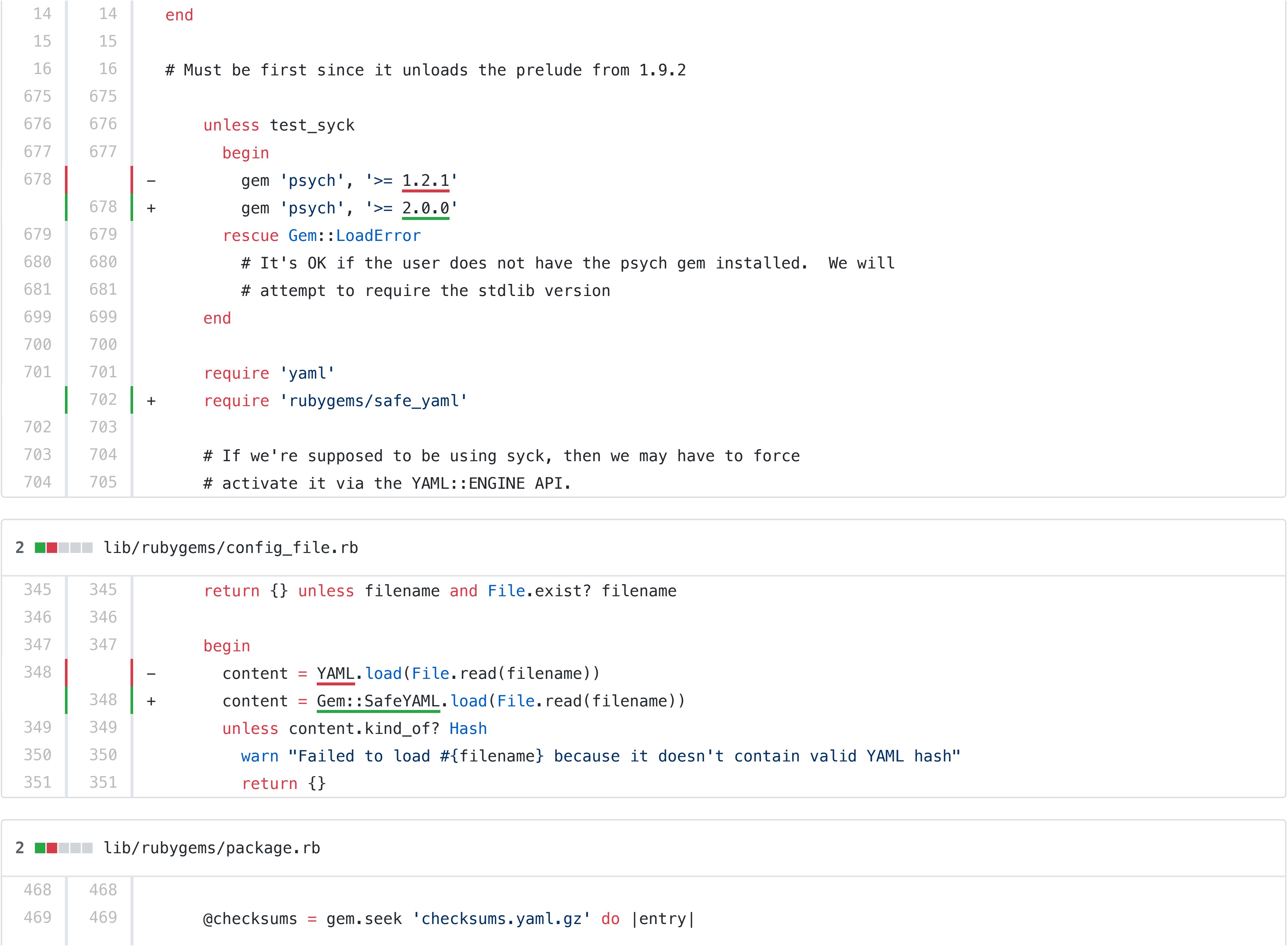
10 🟩🟩🟥🟥⬜ lib/net/ftp.rb

```
       if localfile
         if @resume
           rest_offset = File.size?(localfile)
-          f = open(localfile, "a")
+          f = File.open(localfile, "a")
         else
           rest_offset = nil
-          f = open(localfile, "w")
+          f = File.open(localfile, "w")
         end
```

```diff
        elsif !block_given?
          result = String.new
        f = nil
        result = nil
        if localfile
-         f = open(localfile, "w")
+         f = File.open(localfile, "w")
        elsif !block_given?
          result = String.new
        end
        else
          rest_offset = nil
        end
-       f = open(localfile)
+       f = File.open(localfile)
        begin
          f.binmode
          if rest_offset
      # passing in the transmitted data one line at a time.
      #
      def puttextfile(localfile, remotefile = File.basename(localfile), &block) # :yield: line
-       f = open(localfile)
+       f = File.open(localfile)
        begin
          storlines("STOR #{remotefile}", f, &block)
        ensure
```

5 🟩🟩🟩🟥🟥 lib/rubygems.rb

```diff
  require 'thread'

  module Gem
-   VERSION = "2.6.13"
+   VERSION = "2.6.14"
```



```
  end

  # Must be first since it unloads the prelude from 1.9.2

      unless test_syck
        begin
-         gem 'psych', '>= 1.2.1'
+         gem 'psych', '>= 2.0.0'
        rescue Gem::LoadError
          # It's OK if the user does not have the psych gem installed.  We will
          # attempt to require the stdlib version
      end

      require 'yaml'
+     require 'rubygems/safe_yaml'

      # If we're supposed to be using syck, then we may have to force
      # activate it via the YAML::ENGINE API.
```

2 🟩🟥⬜⬜⬜ lib/rubygems/config_file.rb

```
      return {} unless filename and File.exist? filename

      begin
-       content = YAML.load(File.read(filename))
+       content = Gem::SafeYAML.load(File.read(filename))
        unless content.kind_of? Hash
          warn "Failed to load #{filename} because it doesn't contain valid YAML hash"
          return {}
```

2 🟩🟥⬜⬜⬜ lib/rubygems/package.rb

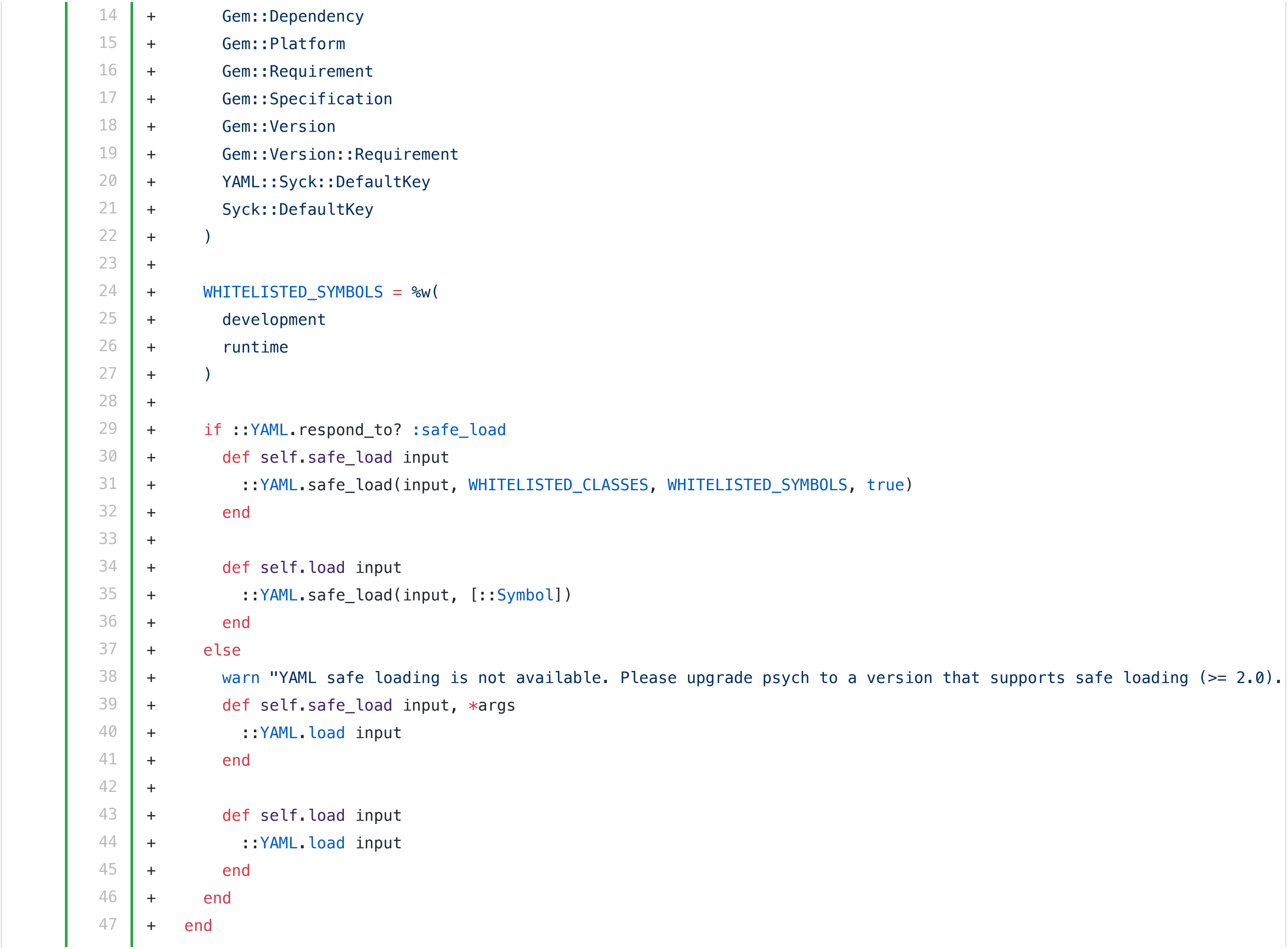
```

      @checksums = gem.seek 'checksums.yaml.gz' do |entry|
          Zlib::GzipReader.wrap entry do |gz_io|
-           YAML.load gz_io.read
+           Gem::SafeYAML.safe_load gz_io.read
          end
        end
      end
```

2 🟩🟥⬜⬜⬜ lib/rubygems/package/old.rb

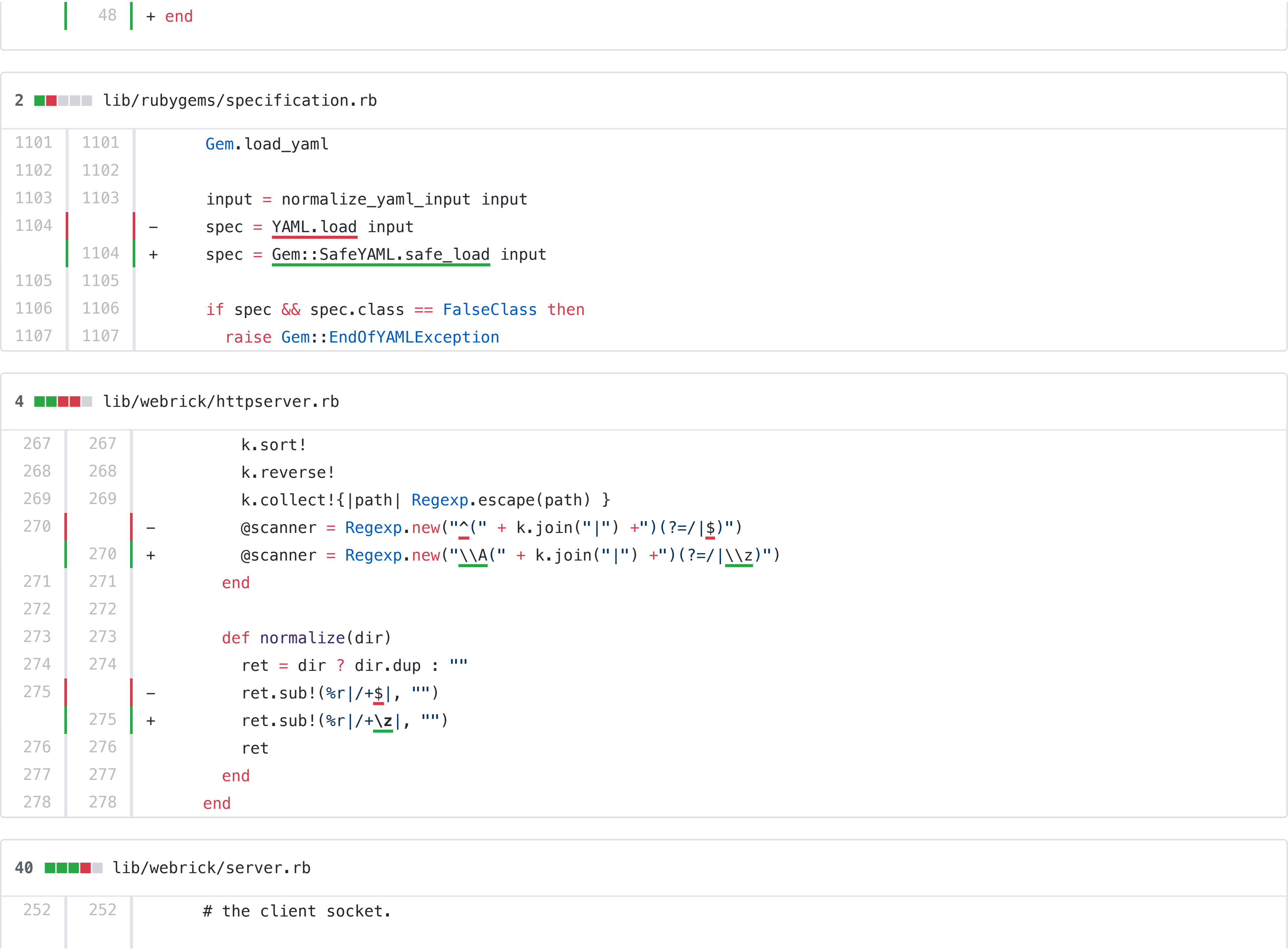
```
          header << line
        end

-       YAML.load header
+       Gem::SafeYAML.safe_load header
      end

      ##
```

48 🟩🟩🟩🟩🟩 lib/rubygems/safe_yaml.rb

```
@@ -0,0 +1,48 @@
+ module Gem
+
+   ###
+   # This module is used for safely loading YAML specs from a gem.  The
+   # `safe_load` method defined on this module is specifically designed for
+   # loading Gem specifications.  For loading other YAML safely, please see
+   # Psych.safe_load
+
+   module SafeYAML
+     WHITELISTED_CLASSES = %w(
+       Symbol
+       Time
+       Date
```



```
+      Gem::Dependency
+      Gem::Platform
+      Gem::Requirement
+      Gem::Specification
+      Gem::Version
+      Gem::Version::Requirement
+      YAML::Syck::DefaultKey
+      Syck::DefaultKey
+    )
+
+    WHITELISTED_SYMBOLS = %w(
+      development
+      runtime
+    )
+
+    if ::YAML.respond_to? :safe_load
+      def self.safe_load input
+        ::YAML.safe_load(input, WHITELISTED_CLASSES, WHITELISTED_SYMBOLS, true)
+      end
+
+      def self.load input
+        ::YAML.safe_load(input, [::Symbol])
+      end
+    else
+      warn "YAML safe loading is not available. Please upgrade psych to a version that supports safe loading (>= 2.0).
+      def self.safe_load input, *args
+        ::YAML.load input
+      end
+
+      def self.load input
+        ::YAML.load input
+      end
+    end
+  end
```



```
+ end
```

2 🟩🟥⬜⬜⬜ lib/rubygems/specification.rb

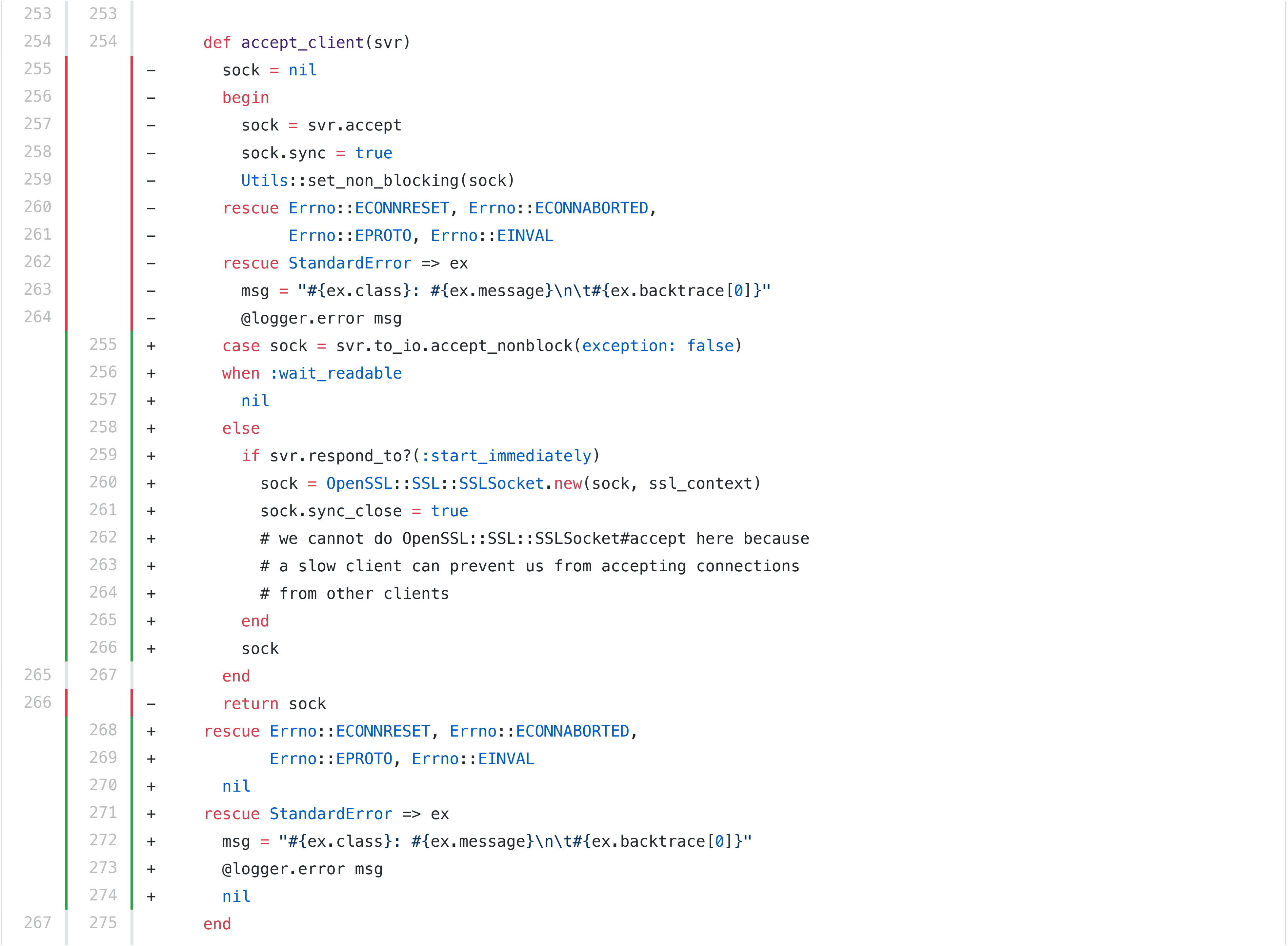
```
       Gem.load_yaml

       input = normalize_yaml_input input
-      spec = YAML.load input
+      spec = Gem::SafeYAML.safe_load input

       if spec && spec.class == FalseClass then
         raise Gem::EndOfYAMLException
```

4 🟩🟩🟥🟥⬜ lib/webrick/httpserver.rb

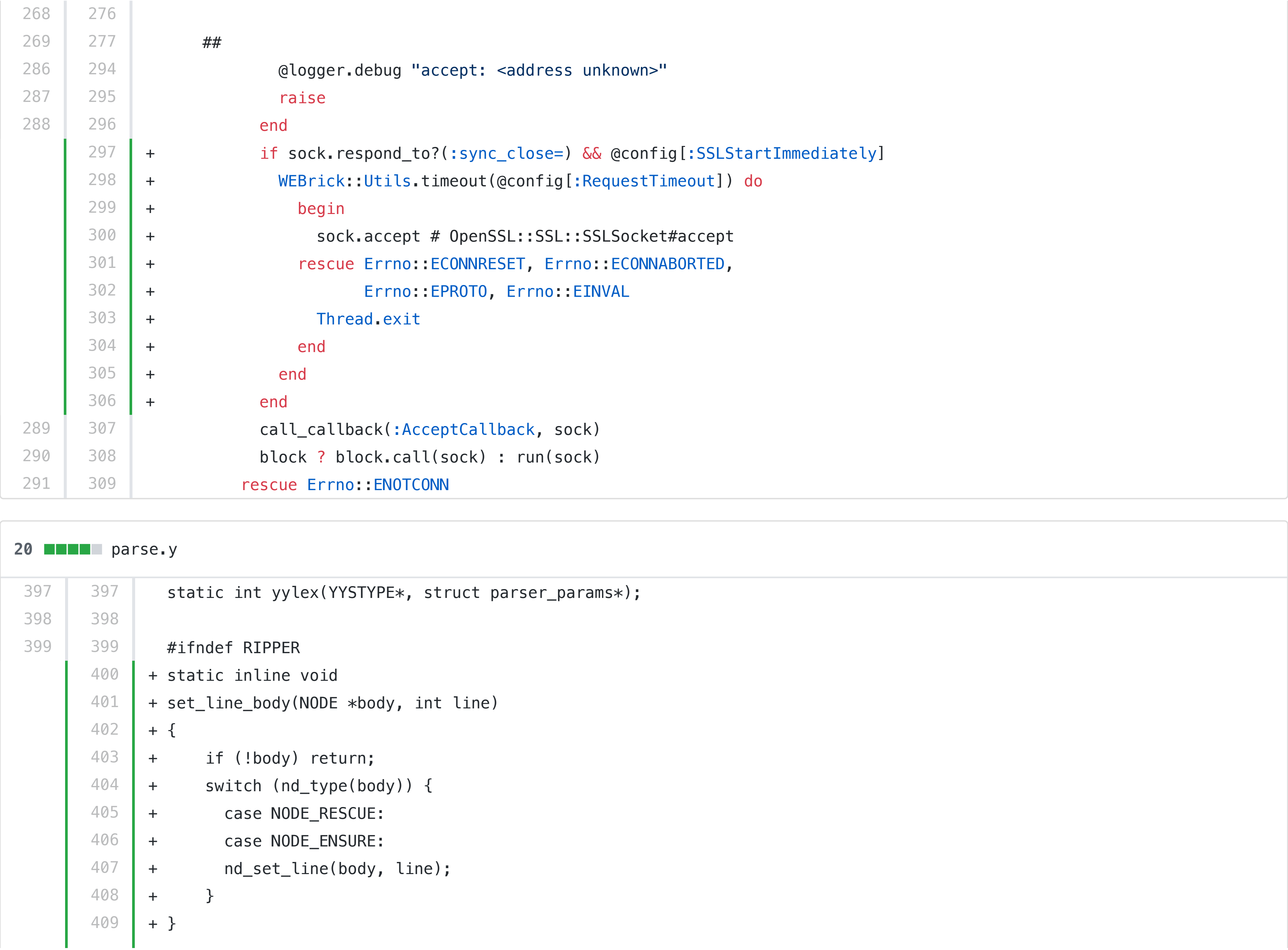
```
           k.sort!
           k.reverse!
           k.collect!{|path| Regexp.escape(path) }
-          @scanner = Regexp.new("^(" + k.join("|") +")(?=/|$)")
+          @scanner = Regexp.new("\\A(" + k.join("|") +")(?=/|\\z)")
         end

         def normalize(dir)
           ret = dir ? dir.dup : ""
-          ret.sub!(%r|/+$|, "")
+          ret.sub!(%r|/+\z|, "")
           ret
         end
       end
```

40 🟩🟩🟩🟥⬜ lib/webrick/server.rb

```
       # the client socket.
```



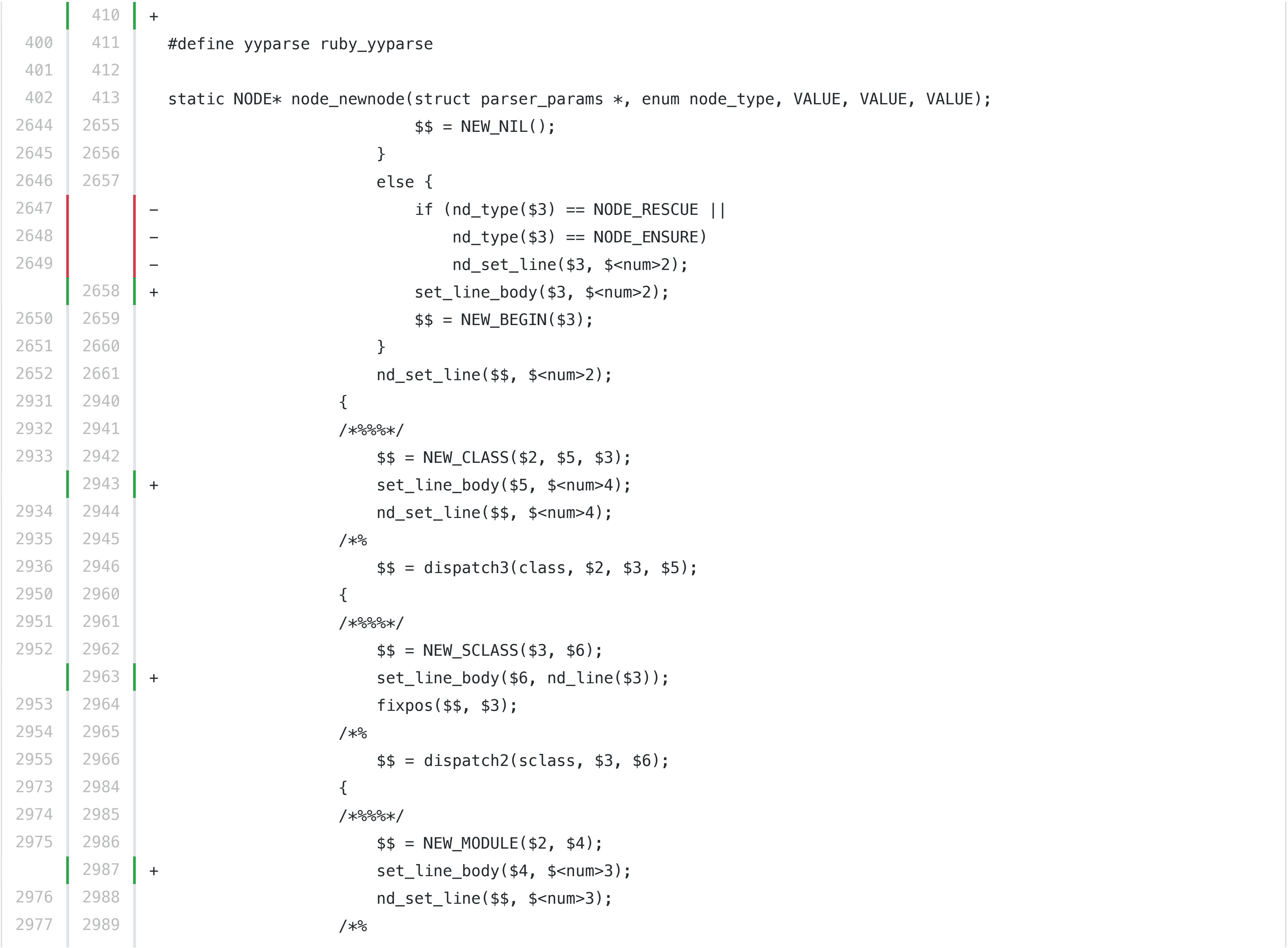
```diff
 
     def accept_client(svr)
-      sock = nil
-      begin
-        sock = svr.accept
-        sock.sync = true
-        Utils::set_non_blocking(sock)
-      rescue Errno::ECONNRESET, Errno::ECONNABORTED,
-             Errno::EPROTO, Errno::EINVAL
-      rescue StandardError => ex
-        msg = "#{ex.class}: #{ex.message}\n\t#{ex.backtrace[0]}"
-        @logger.error msg
+      case sock = svr.to_io.accept_nonblock(exception: false)
+      when :wait_readable
+        nil
+      else
+        if svr.respond_to?(:start_immediately)
+          sock = OpenSSL::SSL::SSLSocket.new(sock, ssl_context)
+          sock.sync_close = true
+          # we cannot do OpenSSL::SSL::SSLSocket#accept here because
+          # a slow client can prevent us from accepting connections
+          # from other clients
+        end
+        sock
       end
-      return sock
+    rescue Errno::ECONNRESET, Errno::ECONNABORTED,
+           Errno::EPROTO, Errno::EINVAL
+      nil
+    rescue StandardError => ex
+      msg = "#{ex.class}: #{ex.message}\n\t#{ex.backtrace[0]}"
+      @logger.error msg
+      nil
     end
```



```
     ##
             @logger.debug "accept: <address unknown>"
             raise
           end
+          if sock.respond_to?(:sync_close=) && @config[:SSLStartImmediately]
+            WEBrick::Utils.timeout(@config[:RequestTimeout]) do
+              begin
+                sock.accept # OpenSSL::SSL::SSLSocket#accept
+              rescue Errno::ECONNRESET, Errno::ECONNABORTED,
+                     Errno::EPROTO, Errno::EINVAL
+                Thread.exit
+              end
+            end
+          end
           call_callback(:AcceptCallback, sock)
           block ? block.call(sock) : run(sock)
         rescue Errno::ENOTCONN
```

20 🟩🟩🟩🟩⬜ parse.y

```
  static int yylex(YYSTYPE*, struct parser_params*);

  #ifndef RIPPER
+ static inline void
+ set_line_body(NODE *body, int line)
+ {
+     if (!body) return;
+     switch (nd_type(body)) {
+       case NODE_RESCUE:
+       case NODE_ENSURE:
+       nd_set_line(body, line);
+     }
+ }
```



```
+
  #define yyparse ruby_yyparse

  static NODE* node_newnode(struct parser_params *, enum node_type, VALUE, VALUE, VALUE);
                            $$ = NEW_NIL();
                        }
                        else {
-                           if (nd_type($3) == NODE_RESCUE ||
-                               nd_type($3) == NODE_ENSURE)
-                               nd_set_line($3, $<num>2);
+                           set_line_body($3, $<num>2);
                            $$ = NEW_BEGIN($3);
                        }
                        nd_set_line($$, $<num>2);
                    {
                    /*%%%*/
                        $$ = NEW_CLASS($2, $5, $3);
+                       set_line_body($5, $<num>4);
                        nd_set_line($$, $<num>4);
                    /*%
                        $$ = dispatch3(class, $2, $3, $5);
                    {
                    /*%%%*/
                        $$ = NEW_SCLASS($3, $6);
+                       set_line_body($6, nd_line($3));
                        fixpos($$, $3);
                    /*%
                        $$ = dispatch2(sclass, $3, $6);
                    {
                    /*%%%*/
                        $$ = NEW_MODULE($2, $4);
+                       set_line_body($4, $<num>3);
                        nd_set_line($$, $<num>3);
                    /*%
```

```diff
                        $$ = dispatch2(module, $2, $4);
                        NODE *body = remove_begin($6);
                        reduce_nodes(&body);
                        $$ = NEW_DEFN($2, $5, body, METHOD_VISI_PRIVATE);
+                       set_line_body(body, $<num>1);
                        nd_set_line($$, $<num>1);
                    /*%
                        $$ = dispatch3(def, $2, $5, $6);
                        NODE *body = remove_begin($8);
                        reduce_nodes(&body);
                        $$ = NEW_DEFS($2, $5, $7, body);
+                       set_line_body(body, $<num>1);
                        nd_set_line($$, $<num>1);
                    /*%
                        $$ = dispatch5(defs, $2, $3, $5, $7, $8);
```

234 🟩🟩🟩🟩🟩 test/net/ftp/test_ftp.rb

```diff
  require "ostruct"
  require "stringio"
  require "tempfile"
+ require "tmpdir"

  class FTPTest < Test::Unit::TestCase
    SERVER_NAME = "localhost"
      end
    end

+   def test_getbinaryfile_command_injection
+     skip "| is not allowed in filename on Windows" if windows?
+     [false, true].each do |resume|
+       commands = []
+       binary_data = (0..0xff).map {|i| i.chr}.join * 4 * 3
+       server = create_ftp_server { |sock|
+        sock.print("220 (test_ftp).\r\n")
+        commands.push(sock.gets)
+        sock.print("331 Please specify the password.\r\n")
+        commands.push(sock.gets)
+        sock.print("230 Login successful.\r\n")
+        commands.push(sock.gets)
+        sock.print("200 Switching to Binary mode.\r\n")
+        line = sock.gets
+        commands.push(line)
+        host, port = process_port_or_eprt(sock, line)
+        commands.push(sock.gets)
+        sock.print("150 Opening BINARY mode data connection for |echo hello (#{binary_data.size} bytes)\r\n")
+        conn = TCPSocket.new(host, port)
+        binary_data.scan(/.{1,1024}/nm) do |s|
+          conn.print(s)
+        end
+        conn.shutdown(Socket::SHUT_WR)
+        conn.read
+        conn.close
+        sock.print("226 Transfer complete.\r\n")
+      }
+      begin
+        chdir_to_tmpdir do
+          begin
+            ftp = Net::FTP.new
+            ftp.resume = resume
+            ftp.read_timeout = 0.2
+            ftp.connect(SERVER_ADDR, server.port)
+            ftp.login
+            assert_match(/\AUSER /, commands.shift)
+            assert_match(/\APASS /, commands.shift)
+            assert_equal("TYPE I\r\n", commands.shift)
+            ftp.getbinaryfile("|echo hello")
+            assert_equal(binary_data, File.binread("./|echo hello"))
+             assert_match(/\A(PORT|EPRT) /, commands.shift)
+             assert_equal("RETR |echo hello\r\n", commands.shift)
+             assert_equal(nil, commands.shift)
+           ensure
+             ftp.close if ftp
+           end
+         end
+       ensure
+         server.close
+       end
+     end
+   end
+
+   def test_gettextfile_command_injection
+     skip "| is not allowed in filename on Windows" if windows?
+     commands = []
+     text_data = <<EOF.gsub(/\n/, "\r\n")
+ foo
+ bar
+ baz
+ EOF
+     server = create_ftp_server { |sock|
+       sock.print("220 (test_ftp).\r\n")
+       commands.push(sock.gets)
+       sock.print("331 Please specify the password.\r\n")
+       commands.push(sock.gets)
+       sock.print("230 Login successful.\r\n")
+       commands.push(sock.gets)
+       sock.print("200 Switching to Binary mode.\r\n")
+       commands.push(sock.gets)
+       sock.print("200 Switching to ASCII mode.\r\n")
+       line = sock.gets
+       commands.push(line)
+       host, port = process_port_or_eprt(sock, line)
+      commands.push(sock.gets)
+      sock.print("150 Opening TEXT mode data connection for |echo hello (#{text_data.size} bytes)\r\n")
+      conn = TCPSocket.new(host, port)
+      text_data.each_line do |l|
+        conn.print(l)
+      end
+      conn.shutdown(Socket::SHUT_WR)
+      conn.read
+      conn.close
+      sock.print("226 Transfer complete.\r\n")
+      commands.push(sock.gets)
+      sock.print("200 Switching to Binary mode.\r\n")
+    }
+    begin
+      chdir_to_tmpdir do
+        begin
+          ftp = Net::FTP.new
+          ftp.connect(SERVER_ADDR, server.port)
+          ftp.login
+          assert_match(/\AUSER /, commands.shift)
+          assert_match(/\APASS /, commands.shift)
+          assert_equal("TYPE I\r\n", commands.shift)
+          ftp.gettextfile("|echo hello")
+          assert_equal(text_data.gsub(/\r\n/, "\n"),
+                       File.binread("./|echo hello"))
+          assert_equal("TYPE A\r\n", commands.shift)
+          assert_match(/\A(PORT|EPRT) /, commands.shift)
+          assert_equal("RETR |echo hello\r\n", commands.shift)
+          assert_equal("TYPE I\r\n", commands.shift)
+          assert_equal(nil, commands.shift)
+        ensure
+          ftp.close if ftp
+        end
+      end
+    ensure
+      server.close
+    end
+  end
+
+  def test_putbinaryfile_command_injection
+    skip "| is not allowed in filename on Windows" if windows?
+    commands = []
+    binary_data = (0..0xff).map {|i| i.chr}.join * 4 * 3
+    received_data = nil
+    server = create_ftp_server { |sock|
+      sock.print("220 (test_ftp).\r\n")
+      commands.push(sock.gets)
+      sock.print("331 Please specify the password.\r\n")
+      commands.push(sock.gets)
+      sock.print("230 Login successful.\r\n")
+      commands.push(sock.gets)
+      sock.print("200 Switching to Binary mode.\r\n")
+      line = sock.gets
+      commands.push(line)
+      host, port = process_port_or_eprt(sock, line)
+      commands.push(sock.gets)
+      sock.print("150 Opening BINARY mode data connection for |echo hello (#{binary_data.size} bytes)\r\n")
+      conn = TCPSocket.new(host, port)
+      received_data = conn.read
+      conn.close
+      sock.print("226 Transfer complete.\r\n")
+    }
+    begin
+      chdir_to_tmpdir do
+        File.binwrite("./|echo hello", binary_data)
+        begin
+          ftp = Net::FTP.new
+          ftp.read_timeout = 0.2
+          ftp.connect(SERVER_ADDR, server.port)
+          ftp.login
+          assert_match(/\AUSER /, commands.shift)
+          assert_match(/\APASS /, commands.shift)
+          assert_equal("TYPE I\r\n", commands.shift)
+          ftp.putbinaryfile("|echo hello")
+          assert_equal(binary_data, received_data)
+          assert_match(/\A(PORT|EPRT) /, commands.shift)
+          assert_equal("STOR |echo hello\r\n", commands.shift)
+          assert_equal(nil, commands.shift)
+        ensure
+          ftp.close if ftp
+        end
+      end
+    ensure
+      server.close
+    end
+  end
+
+  def test_puttextfile_command_injection
+    skip "| is not allowed in filename on Windows" if windows?
+    commands = []
+    received_data = nil
+    server = create_ftp_server { |sock|
+      sock.print("220 (test_ftp).\r\n")
+      commands.push(sock.gets)
+      sock.print("331 Please specify the password.\r\n")
+      commands.push(sock.gets)
+      sock.print("230 Login successful.\r\n")
+      commands.push(sock.gets)
+      sock.print("200 Switching to Binary mode.\r\n")
+      commands.push(sock.gets)
+      sock.print("200 Switching to ASCII mode.\r\n")
+      line = sock.gets
```

```diff
+      commands.push(line)
+      host, port = process_port_or_eprt(sock, line)
+      commands.push(sock.gets)
+      sock.print("150 Opening TEXT mode data connection for |echo hello\r\n")
+      conn = TCPSocket.new(host, port)
+      received_data = conn.read
+      conn.close
+      sock.print("226 Transfer complete.\r\n")
+      commands.push(sock.gets)
+      sock.print("200 Switching to Binary mode.\r\n")
+    }
+    begin
+      chdir_to_tmpdir do
+        File.open("|echo hello", "w") do |f|
+          f.puts("foo")
+          f.puts("bar")
+          f.puts("baz")
+        end
+        begin
+          ftp = Net::FTP.new
+          ftp.connect(SERVER_ADDR, server.port)
+          ftp.login
+          assert_match(/\AUSER /, commands.shift)
+          assert_match(/\APASS /, commands.shift)
+          assert_equal("TYPE I\r\n", commands.shift)
+          ftp.puttextfile("|echo hello")
+          assert_equal(<<EOF.gsub(/\n/, "\r\n"), received_data)
+foo
+bar
+baz
+EOF
+          assert_equal("TYPE A\r\n", commands.shift)
+          assert_match(/\A(PORT|EPRT) /, commands.shift)
+          assert_equal("STOR |echo hello\r\n", commands.shift)
+           assert_equal("TYPE I\r\n", commands.shift)
+           assert_equal(nil, commands.shift)
+         ensure
+           ftp.close if ftp
+         end
+       end
+     ensure
+       server.close
+     end
+   end
+
    private

    def create_ftp_server(sleep_time = nil)
      end
      return data_server
    end
+
+   def chdir_to_tmpdir
+     Dir.mktmpdir do |dir|
+       pwd = Dir.pwd
+       Dir.chdir(dir)
+       begin
+         yield
+       ensure
+         Dir.chdir(pwd)
+       end
+     end
+   end
  end
```

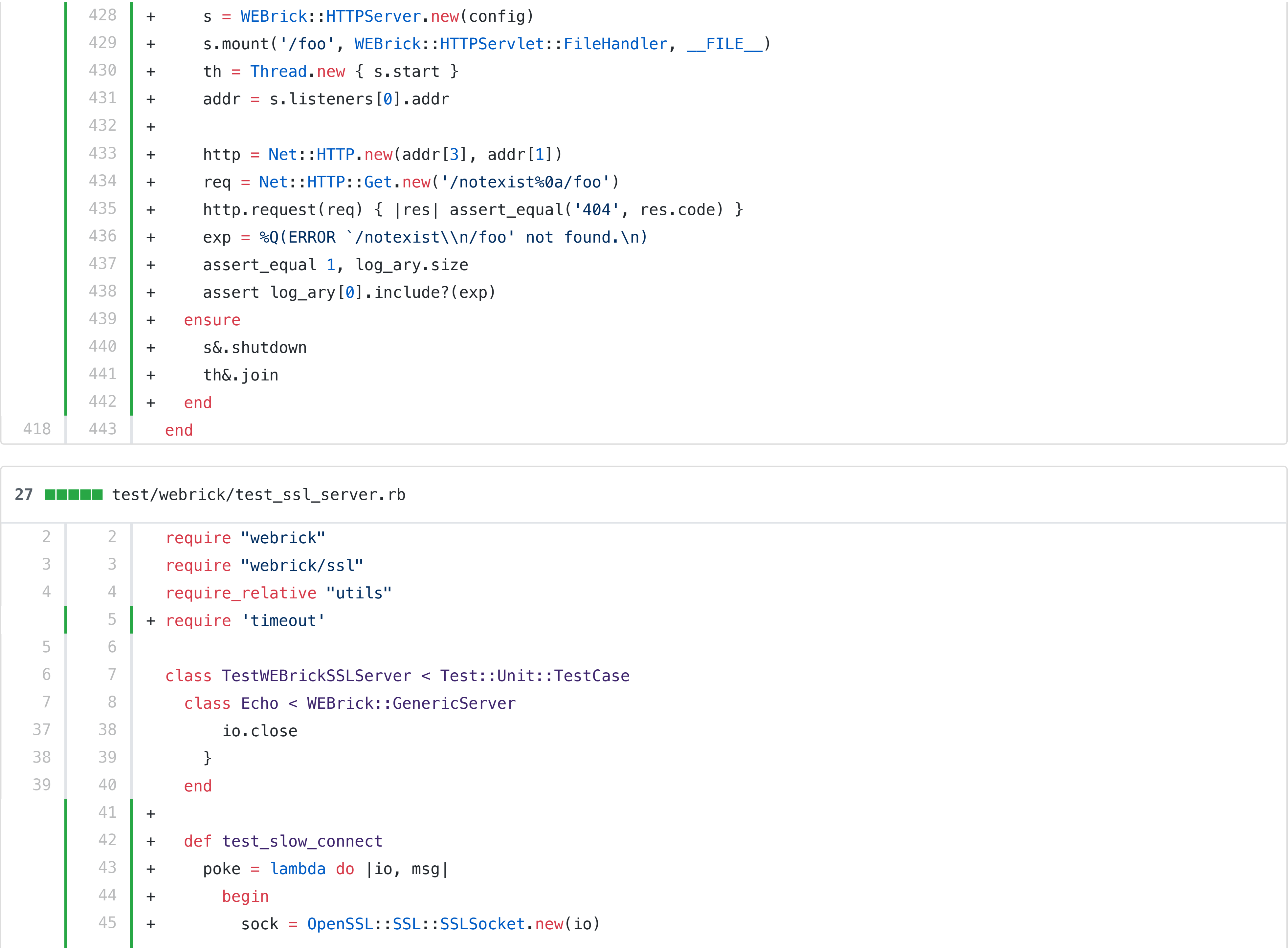
13 🟩🟩🟩🟩🟩 test/ruby/test_parse.rb

```
        end;
```

```
   end

+  def test_method_location_in_rescue
+    bug = '[ruby-core:79388] [Bug #13181]'
+    obj, line = Object.new, __LINE__+1
+    def obj.location
+      #
+      raise
+    rescue
+      caller_locations(1, 1)[0]
+    end
+
+    assert_equal(line, obj.location.lineno, bug)
+  end
+
 =begin
   def test_past_scope_variable
     assert_warning(/past scope/) {catch {|tag| eval("BEGIN{throw tag}; tap {a = 1}; a")}}
```

25 🟩🟩🟩🟩🟩 test/webrick/test_httpserver.rb

```
     }
     assert_equal(0, requested, "Server responded to #{requested} requests after shutdown")
   end
+
+  def test_cntrl_in_path
+    log_ary = []
+    access_log_ary = []
+    config = {
+      :Port => 0,
+      :BindAddress => '127.0.0.1',
+      :Logger => WEBrick::Log.new(log_ary, WEBrick::BasicLog::WARN),
+      :AccessLog => [[access_log_ary, '']],
+    }
```



```
+     s = WEBrick::HTTPServer.new(config)
+     s.mount('/foo', WEBrick::HTTPServlet::FileHandler, __FILE__)
+     th = Thread.new { s.start }
+     addr = s.listeners[0].addr
+
+     http = Net::HTTP.new(addr[3], addr[1])
+     req = Net::HTTP::Get.new('/notexist%0a/foo')
+     http.request(req) { |res| assert_equal('404', res.code) }
+     exp = %Q(ERROR `/notexist\\n/foo' not found.\n)
+     assert_equal 1, log_ary.size
+     assert log_ary[0].include?(exp)
+   ensure
+     s&.shutdown
+     th&.join
+   end
  end
```

27 ■■■■■ test/webrick/test_ssl_server.rb

```
  require "webrick"
  require "webrick/ssl"
  require_relative "utils"
+ require 'timeout'

  class TestWEBrickSSLServer < Test::Unit::TestCase
    class Echo < WEBrick::GenericServer
        io.close
      }
    end
+
+   def test_slow_connect
+     poke = lambda do |io, msg|
+       begin
+         sock = OpenSSL::SSL::SSLSocket.new(io)
+         sock.connect
+         sock.puts(msg)
+         assert_equal "#{msg}\n", sock.gets, msg
+       ensure
+         sock&.close
+         io.close
+       end
+     end
+     config = {
+       :SSLEnable => true,
+       :SSLCertName => "/C=JP/O=www.ruby-lang.org/CN=Ruby",
+     }
+     Timeout.timeout(10) do
+       TestWEBrick.start_server(Echo, config) do |server, addr, port, log|
+         outer = TCPSocket.new(addr, port)
+         inner = TCPSocket.new(addr, port)
+         poke.call(inner, 'fast TLS negotiation')
+         poke.call(outer, 'slow TLS negotiation')
+       end
+     end
+   end
  end
```

8 🟩🟩🟥🟥⬜ version.h

```
@@ -1,9 +1,9 @@
- #define RUBY_VERSION "2.4.2"
- #define RUBY_RELEASE_DATE "2017-09-14"
- #define RUBY_PATCHLEVEL 198
+ #define RUBY_VERSION "2.4.3"
+ #define RUBY_RELEASE_DATE "2017-12-14"
+ #define RUBY_PATCHLEVEL 205

  #define RUBY_RELEASE_YEAR 2017
```

```diff
- #define RUBY_RELEASE_MONTH 9
+ #define RUBY_RELEASE_MONTH 12
  #define RUBY_RELEASE_DAY 14

  #include "ruby/version.h"
```

No commit comments for this range