

Ruby 2.2.4 Language Reference

API Reference

This is the API documentation for 'Ruby 2.2.4 Language Reference API Reference'.

Classes/Modules

Generated by [RDoc](#) 3.12.2.























Generated with the [Darkfish Rdoc Generator](#) 3.

Ruby 2.2.4 Language Reference

API Reference

This is the API documentation for 'Ruby 2.2.4 Language Reference API Reference'.

Files

-  [ChangeLog-YARV](#)
-  [contributing.rdoc](#)
-  [contributors.rdoc](#)
-  [dtrace_probes.rdoc](#)
-  [globals.rdoc](#)
-  [keywords.rdoc](#)
-  [maintainers.rdoc](#)
-  [marshal.rdoc](#)
-  [regexp.rdoc](#)
-  [security.rdoc](#)
-  [standard_library.rdoc](#)
-  [syntax.rdoc](#)
-  [assignment.rdoc](#)
-  [calling_methods.rdoc](#)
-  [control_expressions.rdoc](#)
-  [exceptions.rdoc](#)
-  [literals.rdoc](#)
-  [methods.rdoc](#)
-  [miscellaneous.rdoc](#)
-  [modules_and_classes.rdoc](#)
-  [precedence.rdoc](#)
-  [refinements.rdoc](#)

Generated by [RDoc 3.12.2](#).

Generated with the [Darkfish Rdoc Generator 3](#).

\$Id: ChangeLog 590 2006-12-31 09:02:34Z ko1 \$ # #
YARV ChangeLog # from Mon, 03 May 2004 01:24:19
+0900 #

Sun Dec 31 18:01:50 2006 Koichi Sasada
<ko1@atdot.net>

```
* bin/* : ruby/trunk/bin 11437
```

Sun Dec 31 17:42:05 2006 Koichi Sasada
<ko1@atdot.net>

```
* eval.c : remove old Kernel#funcall definition
```

2006-12-30(Sat) 07:59:26 +0900 Koichi Sasada
<ko1@atdot.net>

```
* catch up ruby/trunk 11437  
  
* eval_intern.h : reorder tag initialization  
  
* eval.c : fix to support __send!, funcall and prohib  
send  
  
* eval_error.h, eval_jump.h, eval_safe.h : fix protot  
  
* eval_method.h, vm.c : check re-definition at rb_add  
  
* yarvcore.h : fix typo  
  
* compile.c : fix white spaces  
  
* lib/delegate.rb : fix to support __send, ...  
  
* lib/getoptlong.rb : fix to work on YARV  
  
* lib/rss/parser.rb : use __send! instead of __send_  
  
* sample/test.rb : comment out codes which use |&b| t
```

```
* ext/ripper/extconf.rb : turn off
* test/ripper/test_files.rb, test_parser_events.rb,
test_scanner_events.rb : fix to check it has ripper m
* vm_dump.c : remove showing file path length limitat
* yarvtest/test_eval.rb : use __send! instead of __se
```

2006-12-19(Tue) 11:46:08 +0900 Koichi Sasada
<ko1@atdot.net>

```
* doc/* : added
* ext/openssl : added
* ext/ripper : added
* test/openssl : added
* test/ripper : added
* misc : added
* rb/ -> tool/ : renamed
* common.mk : fixed for above change
* ruby_doc/* : move to topdir
* sample/* : added
* test2.rb : removed
```

2006-12-15(Fri) 09:42:46 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : remove obsolete codes
* insns.def : fix a comment of getconstant
```

2006-12-13(Wed) 16:26:06 +0900 Koichi Sasada

<ko1@atdot.net>

```
* blockinlining.c, compile.c, compile.h, debug.c, deb  
insnhelper.h, insns.def, iseq.c, thread.c, thread_pth  
thread_pthread.h, thread_win32.ci, thread_win32.h, vm  
vm_dump.c, vm_evalbody.ci, vm_opts.h.base, yarv.h,  
yarv_version.h, yarvcore.c, yarvcore.h :  
add a header includes copyright
```

2006-12-12(Tue) 13:13:32 +0900 Koichi Sasada

<ko1@atdot.net>

```
* rb/insns2vm.rb : add PREFETCH() statement  
  
* vm.h : ditto  
  
* yarvcore.h : fix LIKELY(x) and  
remove main_thread_val field from yarv_vm_t  
  
* yarvcore.c : ditto  
  
* thread.c : support fork  
  
* eval_thread.c : ditto  
  
* process.c : ditto  
  
* signal.c : ditto  
  
* test/ruby/test_signal.rb :  
  
* thread_pthread.ci : rename timer thread functions  
  
* thread_win32.ci : ditto
```

2006-11-10(Fri) 21:29:13 +0900 Koichi Sasada

<ko1@atdot.net>

```
* compile.c : fix to compile arguments  
  
* insns.def : fix to duplicate first array value on c  
instruction
```

```
* yarvtest/test_bin.rb : add a test for above change
* sample/test.rb : fix to catch up Ruby HEAD (fix to
module duplicate)
```

2006-11-10(Fri) 12:49:11 +0900 Koichi Sasada
<ko1@atdot.net>

```
* vm_macro.def : fix to inherit visibility on
NODE_SUPER method invocation
```

2006-11-10(Fri) 09:13:46 +0900 Koichi Sasada
<ko1@atdot.net>

```
* class.c : revert module duplicate inclusion
* parse.y : catch up current Ruby HEAD
* node.h : ditto
* compile.c : ditto
* gc.c : ditto
* iseq.c : ditto
* eval_thread.c : define Continuation (null class)
* vm_dump.c : fix to output backtrae to stderr
* yarvtest/test_block.rb : remove unsupported test
* yarvtest/test_class.rb : add a test about super
* yarvtest/test_syntax.rb : add a test about case/when
```

2006-11-09(Thu) 10:22:59 +0900 Koichi Sasada
<ko1@atdot.net>


```
* call_cfunc.h -> call_cfunc.ci : renamed
* vm_evalbody.h, vm_evalbody.ci : ditto
* thread_pthread.h, thread_pthread.ci : separate decl
implementation
* thread_win32.h, thread_win32.ci : ditto
* thread.c : use *.ci instead of *.c as implementatio
* vm.c : ditto
* common.mk : fix rules for above changes
```

2006-11-08(Wed) 17:23:23 +0900 Koichi Sasada
<ko1@atdot.net>

```
* vm_dump.c : show C level backtrace (pointer only) w
backtrace() function (glibc feature)
* configure.in : ditto
* yarvcore.c : add NSDR method (show C level backtrac
* error.c : fix indent
```

2006-11-07(Tue) 13:17:10 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval.c (rb_set_errinfo) : added
* ruby.h : ditto
* version.h : fix version number
* lib/webrick/utils.rb : fix to remove Thread.critical
* ext/dbm, dl, gdbm, iconv, io, pty, sdbm : added
* test/dbm, gdbm, io, logger, net, readline, sdbm, so
webrick, win32ole, wsdl, xsd : added
```

2006-11-06(Mon) 22:32:18 +0900 Koichi Sasada
<ko1@atdot.net>

```
* array.c : import Ruby HEAD
* ext/socket/extconf.rb : ditto
* ext/socket/socket.c : ditto
* gc.c : ditto
* lib/date.rb : ditto
* lib/net/imap.rb : ditto
* lib/rss/0.9.rb : ditto
* lib/set.rb : ditto
* lib/soap/mapping/rubytypeFactory.rb : ditto
* lib/soap/mimemessage.rb : ditto
* lib/soap/property.rb : ditto
* lib/webrick/httprequest.rb : ditto
* lib/webrick/httputils.rb : ditto
* lib/xmlrpc/create.rb : ditto
* lib/xsd/codegen/gensupport.rb : ditto
* object.c : ditto
* ruby.h : ditto
* string.c : ditto
* version.h : ditto
* rb/ir.rb : fix to use "diffs" directory
* vm_dump.c : add "const"
```

2006-11-06(Mon) 16:36:47 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval_proc.c : remove "static" from external global
* eval_thread.c : ditto
* array.c : fix indent
* insns.def : add a suitable cast
* vm_macro.def : allow scalar value on splat argument
* yarvtest/test_block.rb : fix to synchronize Ruby HE
* rb/insns2vm.rb : remove String#each for 1.9
* template/vm.inc.tpl : ditto (remove String#each_wi
```

2006-11-06(Mon) 13:22:34 +0900 Koichi Sasada
<ko1@atdot.net>

```
* iseq.c : fixed GC debugging outputs
* rb/parse.rb : fixed output format
```

2006-11-04(Sat) 09:46:50 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : fix to duplicate "#{ 'foo' }" string
* yarvtest/test_bin.rb : add a test for above
* ext/readline/readline.c : import Ruby HEAD
* keywords : ditto
* lex.c : ditto
* parse.y : ditto
```

```
* lib/mkrf.rb : ditto
* test/ruby/test_hash.rb : fix to current specificati
* test/ruby/test_string.rb : ditto
```

2006-11-03(Fri) 20:58:36 +0900 Koichi Sasada
<ko1@atdot.net>

```
* ext/nkf/nkf-utf8/utf8tbl.h : missed to add
* configure.in : import ruby HEAD
* test/ruby/test_array.rb : ditto
* test/ruby/test_assignment.rb : ditto
* test/ruby/test_clone.rb : ditto
* test/socket/test_socket.rb : ditto
* test/socket/test_unix.rb : ditto
* test/strscan/test_stringscanner.rb : ditto
* test/testunit/collector/test_dir.rb : ditto
```

2006-11-03(Fri) 20:22:24 +0900 Koichi Sasada
<ko1@atdot.net>

```
* array.c : import current ruby HEAD and apply API ch
This version has some known bugs
* bignum.c : ditto
* blockinlining.c : ditto
* class.c : ditto
* compile.c : ditto
* dir.c : ditto
```

```
* dln.c : ditto
* enum.c : ditto
* enumerator.c : ditto
* error.c : ditto
* eval.c : ditto
* eval_error.h : ditto
* eval_jump.h : ditto
* eval_load.c : ditto
* eval_proc.c : ditto
* ext/*
* file.c : ditto
* gc.c : ditto
* hash.c : ditto
* insns.def : ditto
* instruby.rb : ditto
* intern.h : ditto
* io.c : ditto
* iseq.c : ditto
* lib/*
* marshal.c : ditto
* math.c : ditto
* missing/vsnprintf.c : ditto
* mkconfig.rb : ditto
```

```
* node.h : ditto
* numeric.c : ditto
* object.c : ditto
* oniguruma.h : ditto
* pack.c : ditto
* parse.y : ditto
* prec.c : ditto
* process.c : ditto
* random.c : ditto
* range.c : ditto
* rb/ir.rb : ditto
* re.c : ditto
* regcomp.c : ditto
* regerror.c : ditto
* regexec.c : ditto
* regint.h : ditto
* regparse.c : ditto
* regparse.h : ditto
* ruby.c : ditto
* ruby.h : ditto
* rubytest.rb : ditto
* runruby.rb : ditto
* sample/test.rb : ditto
* signal.c : ditto
```

```
* sprintf.c : ditto
* st.c : ditto
* st.h : ditto
* string.c : ditto
* struct.c : ditto
* test/*
* thread.c : ditto
* time.c : ditto
* util.c : ditto
* variable.c : ditto
* version.h : ditto
* vm.c : ditto
* vm_dump.c : ditto
* vm_macro.def : ditto
* win32/*
```

2006-10-31(Tue) 22:47:50 +0900 Koichi Sasada
<ko1@atdot.net>

```
* parse.y : fix NEWHEAP bugs (import HEAD)
* ruby.c, intern.h, yarcvcore.c (rb_load_file) : chang
return parsed node pointer
* rb/ir.rb : add check mode
```

2006-09-01(Fri) 22:05:28 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : fix a bug of peephole optimization and  
regexp optimization
```

2006-08-21(Mon) 05:27:48 +0900 Koichi Sasada
<ko1@atdot.net>

```
* lib/mathn.rb : remove "remove_method :gcd2"  
* opt_insn_unif.def : unset opt setting  
* opt_operand.def : ditto
```

2006-08-18(Fri) 17:55:31 +0900 Koichi Sasada
<ko1@atdot.net>

```
* common.mk : add dependency of yarvcore.h to thread.  
* gc.c : change comment line  
* thread.c : remove some line break  
* yarvcore.c : reoder initialize sequence to mark mai
```

2006-08-18(Fri) 16:51:34 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.h : add a support for cache values per thr  
* yarvcore.c : ditto  
* gc.c : ditto  
* thread.c : move a expression after acquiring lock  
* compile.c : add a cast to remove warning
```

2006-08-18(Fri) 02:07:45 +0900 Koichi Sasada
<ko1@atdot.net>


```
* compile.c : fix to return rhs value on ATTRASGIN
* insns.def (setn) : add insn setn
* yarvtest/test_bin.rb : add tests for above
```

2006-08-17(Thu) 22:46:08 +0900 Koichi Sasada
<ko1@atdot.net>

```
* vm.c : clear callee_id ([yarv-dev:1073])
```

2006-08-17(Thu) 22:14:15 +0900 Koichi Sasada
<ko1@atdot.net>

```
* thread_pthread.h : fix error message
```

2006-08-17(Thu) 12:23:52 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval.c : change initialize routine order ([yarv-dev:
* yarcvcore.c (Init_yarv) : init th->machine_stack_sta
* thread_pthread.h : add malloc value check ([yarv-de
* insns.def (opt_eq) : fix typo ([yarv-dev:1072])
* yarcvtest/test_opts.rb : add a test for above
* yarcvtest/test_class.rb : add a test for last commit
```

2006-08-17(Thu) 11:02:16 +0900 Koichi Sasada
<ko1@atdot.net>

```
* class.c (clone_method) : check undef-ed method ([ya
```

2006-08-15(Tue) 15:07:43 +0900 Koichi Sasada

<ko1@atdot.net>

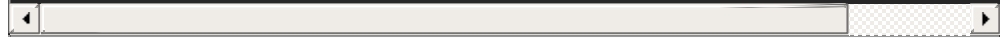
```
* insns.def : fix opt_plus routine ([yarv-dev-en:149]  
* yarvtest/test_opts.rb : add tests for above
```

2006-08-06(Sun) 06:24:51 +0900 Koichi Sasada
<ko1@atdot.net>

```
* common.mk : fix build rule (build only ruby binary  
* yarvcore.[ch] : fix and add yarv_iseq_new_with_* AP  
* blockinlining.c : ditto  
* compile.c : ditto  
* compile.h : ditto  
* iseq.c : ditto  
* eval_method.h : check redefinition for specialized  
* insnhelper.h : ditto  
* insns.def : ditto  
* vm.c : ditto  
* vm.h : ditto  
* numeric.c : add Fixnum#succ  
* thread.c : remove duplicated method Thread#current  
* yarvcore.c : remove duplicated method Proc#clone  
* yarvtest/test_opts.rb : added
```

2006-07-20(Thu) 04:10:13 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : fix [yarv-dev:1041] problem (raise Type
* eval.c : rb_funcall2 send as NOEX_PRIVATE and check
```

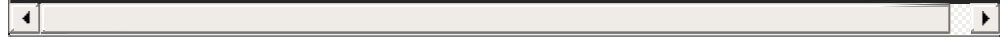


2006-07-20(Thu) 03:38:46 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.c : fix [yarv-dev:1040] bug
```

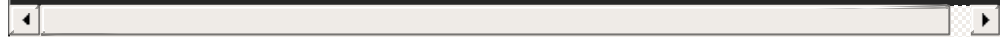
2006-07-18(Tue) 18:45:52 +0900 Koichi Sasada
<ko1@atdot.net>

```
* some files : set property "svn:eol-style" as native
```



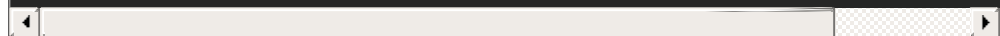
2006-07-18(Tue) 18:35:55 +0900 Koichi Sasada
<ko1@atdot.net>

```
* gc.h : fix a static function name
* vm.c : remove Japanese comments
* yarvcore.c : add a comment
* some files : set property "svn:eol-style" as native
```



2006-07-18(Tue) 16:48:01 +0900 Koichi Sasada
<ko1@atdot.net>

```
* vm.c : remove unused code
* compile.c : add checking value
* iseq.c : ditto
* yarvcore.c : fix yarv_th_eval prototype declaration
* yarvtest/yarvtest.rb : use compile instead of parse
```



2006-07-12(Wed) 15:18:58 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarv_version.h : 0.4.1
* Changes : ditto
```

2006-07-12(Wed) 13:38:03 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval.c : fix indent
* gc.h : fix syntax bug
* thread_pthread.h : vanish warning message
* iseq.c : ditto
* compile.c : ditto
* thread.c : ditto
* vm.c : ditto
* yarvcore.c : prohibit tail call optimization to mark
iseq object
* yarvcore.h : add some allocator function declaratio
* yarvtest/test_eval.rb : remove output
```

2006-07-12(Wed) 05:01:23 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.c : undef alloc funcs
* eval_proc.c : ditto (use factory faction)
* thread.c : ditto
* vm.c : ditto
```

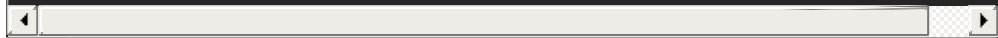
```
* iseq.c : fix compile option creation
* rb/allload.rb : use compile_file method
* rb/compile.rb : ditto
* rb/parse.rb : ditto
* template/insnstbl.html : hide mail addr
```

2006-07-11(Tue) 21:34:29 +0900 Minero Aoki
<aamine@loveruby.net>

```
* test/ruby/test_dir.rb: new test test_JVN_13947696.
```

2006-07-11(Tue) 21:26:41 +0900 Minero Aoki
<aamine@loveruby.net>

```
* test/ruby/test_alias.rb: new test test_JVN_83768862
```



2006-07-11(Tue) 11:33:49 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : fix compile error on C90
```

2006-07-11(Tue) 10:40:23 +0900 Koichi Sasada
<ko1@atdot.net>

```
* disasm.c : removed
* iseq.c : added
* common.mk : ditto
* blockinlining.c : Get*Val => Get*Ptr
* eval.c : ditto
* yarvcore.c : ditto
```

```
* eval_proc.c : ditto
* vm_dump.c : ditto
* vm_macro.def : ditto
* signal.c : ditto
* vm.c : ditto
* thread.c : ditto
* compile.c : rename local variable insnobj => iobj
* compile.c : support yarv_compile_option_t
* gc.h : added
* insns.def : use OPT_CHECKED_RUN instead of IGNORE_O
* rb/compile.rb : use compile option
* template/optinsn.inc.tpl : fix function name
* vm_opts.h.base : change macros
* rb/insns2vm.rb : ditto
* yarv.h : fix yarvcore_eval_parsed parameter type
* yarvcore.c : fix some interfaces (functions)
* yarvcore.h : add a type yarv_compile_option_t
```

2006-07-06(Thu) 13:45:20 +0900 Koichi Sasada
<ko1@atdot.net>

```
* lib/yasm.rb : pass builder object if block arity ==
```

2006-07-05(Wed) 11:23:50 +0900 Koichi Sasada
<ko1@atdot.net>

```
* lib/yasm.rb : fix method name
* vm.c (th_set_top_stack) : check toplevel or not
```

2006-07-04(Tue) 20:05:38 +0900 Koichi Sasada
<ko1@atdot.net>

```
* rb/compile.rb : added
* yarvtest/yarvtest.rb : disable load/store test
```

2006-07-04(Tue) 18:17:15 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : fix some bugs about load iseq data
* disasm.c : ditto (store)
* eval.c (rb_f_local_variables) : fix bugs
* insns.def : fix otp_ltlt condition bug
* vm.c : ditto
* yarvcore.c : rename some functions
* yarvtest/yarvtest.rb : add iseq load/store tests
(to enable this, remove comment)
```

2006-07-03(Mon) 01:54:23 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval_thread.c : add parameter "th" to thread_set_ra
* yarvcore.h : ditto
* eval_intern.h : ditto
* eval.c : ditto
* eval_error.h : declare with ANSI style
```

```
* disasm.c : rename iseq_iseq2simplifiedata() to iseq_da  
* lib/yasm.rb : rename Instruction#to_simplifiedata to  
Instruction#to_a  
* yarvcore.c : ditto  
* vm.c : fix bug (Proc.new{|*args| p args}.call(1) #=  
* yarvtest/test_proc.rb : add a tests for above
```

2006-06-21(Wed) 09:19:06 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : remove yarv_iseq_t#catch_table_ary and  
add yarv_iseq_t#compile_data#catch_table_ary  
* compile.h : ditto  
* yarvcore.c : ditto  
* yarvcore.h : ditto  
* eval_thread.c : remove unused code  
* thread.c : add rb_gc_mark_threads() (from eval_thre
```

2006-05-31(Wed) 21:26:38 +0900 Koichi Sasada
<ko1@atdot.net>

```
* parse.y : prohibit tail call optimization to mark v  
object
```

2006-05-25(Thu) 15:37:11 +0900 Koichi Sasada
<ko1@atdot.net>

```
* blockinlining.c : support NEW_ATTRASGN node  
* class.c : skip undefined method to collect ([yarv-d
```



```
* yarvtest/test_class.rb : add a test for above
* compile.c : fix opt_regexpmatch1 condition
* lib/monitor.rb : fix [yarv-dev:1009]
* rb/insns2vm.rb : fix typo
* thread.c : prohibit unlock by not mutex owner threa
* vm_opts.h.base : change default option
```

2006-05-18(Thu) 16:00:50 +0900 Koichi Sasada
<ko1@atdot.net>

```
* intern.h : fix prototype declarations for last re.c
```

2006-05-18(Thu) 12:12:03 +0900 Koichi Sasada
<ko1@atdot.net>

```
* rb/runruby.rb : added
* thread.c (rb_thread_alone) : check if vm->living_th
is available
```

2006-05-18(Thu) 12:05:35 +0900 Koichi Sasada
<ko1@atdot.net>

```
* signal.c : not mask SIGSEGV
* thread.c : fix debug output on Win32
* thread.c, thread_pthread.h : add some debug prints
* yarvcore.c : mark machine registers on thread_mark
```

2006-05-17(Wed) 18:09:20 +900 Yukihiro Matsumoto
<matz@ruby-lang.org>

```
* dir.c (sys_warning): should not call a vararg funct  
rb_sys_warning() indirectly. [ruby-core:07886]
```

2006-05-17(Wed) 16:41:41 +900 Yukihiro Matsumoto
<matz@ruby-lang.org>

```
* re.c (rb_reg_initialize): should not allow modifyin  
regexps. frozen check moved from rb_reg_initialize  
  
* re.c (rb_reg_initialize): should not modify untaint  
safe levels higher than 3.  
  
* re.c (rb_memcmp): type change from char* to const v  
  
* dir.c (dir_close): should not close untainted dir s  
  
* dir.c (GetDIR): add tainted/frozen check for each d
```

2006-05-07(Sun) 21:06:28 +0900 Koichi Sasada
<ko1@atdot.net>

```
* thread.c : remove Mutex#unlock_and_stop and add Mut  
  
* lib/monitor.rb : ditto  
  
* lib/thread.rb : ditto  
  
* thread_pthread.h : fix stack size  
  
* thread_win32.h : fix sleep  
  
* yarvcore.h : disable to use get/setcontext  
  
* lib/webrick/server.rb : add experimental implementa  
using thraeds pool
```

2006-05-05(Fri) 13:59:00 +0900 Koichi Sasada
<ko1@atdot.net>

```
* test/ruby/test_signal.rb : disable a test
* thread.c : do trylock before lock on mutex_lock
* thread_win32.h : use CriticalSection instead of Mut
```

2006-05-05(Fri) 03:03:22 +0900 Koichi Sasada
<ko1@atdot.net>

```
* common.mk : vtune rule make run test.rb
* disasm.c : fix syntax errors (on VC)
* yarvcore.c : ditto
* lib/thread.rb : Mutex#synchronize is defined here
* lib/*.rb : ditto
* signal.c : separate pthread or not
* thread.c : support lightweight wakeup
* thread_pthread.h : ditto
* thread_win32.h : ditto
* yarvcore.h : ditto
* yarvtest/test_thread.rb : restore last change
```

2006-05-04(Thu) 18:11:43 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval_thread.c : remove rb_thread_interrupt
* intern.h : ditto
* signal.c : change signal transfer route
* thread.c : ditto
```

```
* thread_pthread.h : ditto
* thread_win32.h : ditto
* yarv.h : support GET_VM()
* yarvcore.h : change yarv_thread_t/yarv_vm_t structu
* yarvtest/test_thread.rb : decrease threads to test
```

2006-05-04(Thu) 00:26:18 +0900 Koichi Sasada
<ko1@atdot.net>

```
* thread_pthread.h : experimental support of thread c
```

2006-04-25(Tue) 22:30:14 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.h : remove struct yarv_cmethod_info, add
data structure for profiling and extend yarv_control_
* vm.c : make pop_frame() and apply above change
* eval.c : ditto
* vm_dump.c : ditto
* vm_macro.def : ditto
* insns.def (leave): use pop_frame() instead of
POP_CONTROL_STACK_FRAME() macro
* insnhelper.h : remove some macros
* yarvcore.c : change th_set_top_stack() prototype
```

2006-04-18(Tue) 18:37:08 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c, disasm.c : support export/import excepti
```

```
information
```

```
* yarvcore.h : change "struct catch_table_entry" memb  
order
```

2006-04-13(Thu) 17:11:30 +0900 Koichi Sasada
<ko1@atdot.net>

```
* bignum.c : import ruby 1.9 HEAD (Ruby 1.9.0 2006-04  
* dir.c : ditto  
* enumerator.c : ditto  
* ext/.document : ditto  
* ext/extmk.rb : ditto  
* ext/nkf/lib/kconv.rb : ditto  
* ext/nkf/nkf-utf8/nkf.c : ditto  
* ext/nkf/nkf-utf8/utf8tbl.c : ditto  
* ext/nkf/nkf.c : ditto  
* ext/nkf/test.rb : ditto  
* ext/socket/.cvsignore : ditto  
* ext/win32ole/sample/excel2.rb : ditto  
* ext/win32ole/tests/testOLEMETHOD.rb : ditto  
* ext/win32ole/tests/testOLEPARAM.rb : ditto  
* ext/win32ole/tests/testOLETYPE.rb : ditto  
* ext/win32ole/tests/testOLETYPELIB.rb : ditto  
* ext/win32ole/tests/testOLEVARIABLE.rb : ditto  
* ext/win32ole/tests/testOLEVARIANT.rb : ditto
```

```
* ext/win32ole/tests/testWIN32OLE.rb : ditto
* ext/win32ole/tests/testall.rb : ditto
* ext/win32ole/win32ole.c : ditto
* gc.c : ditto
* instruby.rb : ditto
* io.c : ditto
* lib/delegate.rb : ditto
* lib/fileutils.rb : ditto
* lib/find.rb : ditto
* lib/irb/ruby-lex.rb : ditto
* lib/mkrf.rb : ditto
* lib/net/http.rb : ditto
* lib/open-uri.rb : ditto
* lib/pathname.rb : ditto
* lib/rational.rb : ditto
* lib/rdoc/parsers/parse_rb.rb : ditto
* lib/rdoc/ri/ri_paths.rb : ditto
* lib/resolv.rb : ditto
* lib/test/unit/collector/objectspace.rb : ditto
* lib/webrick/httpservlet/cgihandler.rb : ditto
* math.c : ditto
* mkconfig.rb : ditto
* object.c : ditto
* oniguruma.h : ditto
```

```
* pack.c : ditto
* parse.y : ditto
* re.c : ditto
* re.h : ditto
* regcomp.c : ditto
* regerror.c : ditto
* regparse.c : ditto
* ruby.h : ditto
* rubytest.rb : ditto
* runruby.rb : ditto
* string.c : ditto
* test/digest/test_digest.rb : ditto
* test/pathname/test_pathname.rb : ditto
* test/ruby/envutil.rb : ditto
* test/ruby/test_float.rb : ditto
* test/ruby/test_pack.rb : ditto
* time.c : ditto
* util.c : ditto
* version.h : ditto
* win32/mkexports.rb : ditto
* win32/resource.rb : ditto
* win32/win32.c : ditto
```

2006-04-11(Tue) 11:26:53 +0900 Koichi Sasada
<ko1@atdot.net>

```
* rb/yasm.rb : move to lib/yasm.rb
```

2006-04-09(Sun) 03:04:04 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : change to accept method iseq object whe  
simple data  
  
* yarvcore.c : add a debug output  
  
* rb/yasm.rb : change some interfaces
```

2006-04-07(Fri) 20:25:03 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : fix miss about range of catch "next"  
  
* eval.c : add braces
```

2006-04-07(Fri) 11:09:43 +0900 Koichi Sasada
<ko1@atdot.net>

```
* common.mk : fix some make rules  
  
* insns.def : rename some instructions name  
  
* rb/insns2vm.rb : change some operand type name  
  
* vm_evalbody.h : ditto  
  
* template/insns.inc.tmpl : add YARV_MAX_INSTRUCTION_  
  
* compile.c, disasm.c, yarvcore.c : support load/stor  
data structure such as array, literals, and so on  
  
* rb/yasm.rb : supported
```



```
* vm.c : change interface of eval_define_method
* yarvcore.h : remove unused externals
```

2006-03-08(Wed) 10:31:29 +0900 Minero Aoki
<aamine@loveruby.net>

```
* lib/delegate.rb (DelegateClass): do not delegate #s
#funcall.
```

2006-02-27(Mon) 22:39:17 +0900 Minero Aoki
<aamine@loveruby.net>

```
* lib/thread.rb: last commit causes busy loop, revert
* lib/thread.rb: non_block=true wrongly caused Thread
```

2006-02-27(Mon) 21:33:49 +0900 Koichi Sasada
<ko1@atdot.net>

```
* common.mk : fix to display command line
* compile.c : fix comparison between a pointer and 0
* debug.c : fix to output stder
* disasm.c : add debug function
* vm_dump.c : ditto
* eval_proc.c : fix to skip class definition
* ruby.h : fix T_VALUE to T_VALUES
* gc.c : ditto
* node.h : fix prototypes
* vm.c : add VM_DEBUG macro
```

```
* vm.c : fix compile error on VC++
* vm.c : fix to inherit last lfp[0] on th_set_finish_
* vm.c : fix to add one svar location for any frame
* vm_macro.def : ditto
* yarvcore.h : add YARV_CLASS_SPECIAL_P() and YARV_BL
* rdoc/ : removed
* insns.def : fix to propagete throw state
```

2006-02-27(Mon) 13:54:47 +0900 Minero Aoki
<aamine@loveruby.net>

```
* ext/syslog: imported from Ruby CVS trunk HEAD.
* ext/racc: ditto.
```

2006-02-27(Mon) 12:47:10 +0900 Minero Aoki
<aamine@loveruby.net>

```
* parse.y: follow coding style change.
```

2006-02-27(Mon) 11:53:07 +0900 Minero Aoki
<aamine@loveruby.net>

```
* lib/README: imported from Ruby CVS trunk HEAD.
* lib/gserver.rb: ditto.
* lib/readbytes.rb: ditto.
* lib/parsearg.rb: ditto.
* lib/racc: ditto.
* lib/rinda: ditto.
```

2006-02-27(Mon) 11:27:19 +0900 Minero Aoki
<aamine@loveruby.net>

```
* lib/thread.rb (Queue#pop): faster code. [yarv-dev:9  
* lib/thread.rb (Queue#pop): avoid to push same threa  
  @waiting.
```

2006-02-23(Thu) 23:32:53 +0900 Minero Aoki
<aamine@loveruby.net>

```
* lib/open3.rb: imported from Ruby CVS trunk HEAD (re
```

2006-02-23(Thu) 15:10:09 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval.c : support rb_frame_self()  
* eval_intern.h (th_get_ruby_level_cfp) : return 0 if  
* eval_load.c : comment out scope set  
* yarvcore.c : fix to initialize/free process of iseq  
* vm.c (th_invoke_proc) : fix to set special cref alw  
* yarvtest/test_proc.rb : add a test for above
```

2006-02-22(Wed) 23:33:47 +0900 Koichi Sasada
<ko1@atdot.net>

```
* common.mk : add rule "runruby"  
* eval_thread.c : remove obsolete comment  
* eval.c : remove unused functions  
* signal.c : ditto
```

```
* gc.c : add rb_register_mark_object() and use it
* eval_load.c : ditto
* eval_proc.c : ditto
* ext/etc/etc.c : ditto
* ext/win32ole/win32ole.c : ditto
* ruby.h : ditto
* yarvcore.h : ditto
* thread.c : add rb_thread_run_parallel()
* yarvcore.c : change bootstrap
```

2006-02-22(Wed) 19:27:33 +0900 Koichi Sasada
<ko1@atdot.net>

```
* ext/win32ole/.cvsignore : removed
* ext/win32ole/.document : ditto
```

2006-02-22(Wed) 18:17:06 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.c : set Binding as YARVCore::VM::Binding
```

2006-02-22(Wed) 12:54:45 +0900 Koichi Sasada
<ko1@atdot.net>

```
* ChangeLog : remove needless line
```

2006-02-22(Wed) 12:49:02 +0900 Koichi Sasada
<ko1@atdot.net>

```
* rubysig.h : remove CHECK_INTS
* eval.c : ditto
```

```
* eval_load.c : ditto
* ext/readline/readline.c : ditto
* thread.c : ditto
* win32/win32.c : ditto
* yarv_version.h : 0.4.0
* Changes : ditto
```

2006-02-22(Wed) 11:36:04 +0900 Koichi Sasada
<ko1@atdot.net>

```
* test.rb : removed
```

2006-02-22(Wed) 11:12:17 +0900 Koichi Sasada
<ko1@atdot.net>

```
* README : renewed
* version.c : fixed version message
* yarvext/ : removed
```

2006-02-22(Wed) 10:33:04 +0900 Koichi Sasada
<ko1@atdot.net>

```
* lib/.document : imported from Ruby 1.9 HEAD
* .document : ditto
* ext/.document : ditto
* lib/ftools.rb : ditto
* lib/rdoc/ : ditto
* eval_thread.c : remove unused functions
```

```
* process.c : ditto
* rb/insns2vm.rb : compare modified date of vm_opts.h
vm_opts.h.base
* ruby.h : rename RValue to RValues
* gc.c : ditto
* vm.c : ditto
```

2006-02-22(Wed) 06:32:10 +0900 Koichi Sasada
<ko1@atdot.net>

```
* configure.in : remove last commit
```

2006-02-22(Wed) 06:18:53 +0900 Koichi Sasada
<ko1@atdot.net>

```
* configure.in : add default program prefix "-yarv"
```

2006-02-22(Wed) 06:11:36 +0900 Koichi Sasada
<ko1@atdot.net>

```
* common.mk : change default rule (same as HEAD)
* configure : removed
* eval.c : remove last commit
* vm.c : fix stack traverse
* yarvcore.c : initialize top of control frame
* version.c : 2.0
* version.h : ditto
```

2006-02-22(Wed) 04:50:42 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval.c : change to rewind C level control frame
* vm.c : change to initialize cfp#proc and fix compar
cfp and limit_cfp
* yarvcore.c : remove last commit
```

2006-02-22(Wed) 03:25:56 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.c : initialize each stack of thread
```

2006-02-22(Wed) 00:02:08 +0900 Koichi Sasada
<ko1@atdot.net>

```
* thread.c : fix synchornize return value ([yarv-dev:
and some synchornization error
* thread_pthread.h : add debug helper function
```

2006-02-21(Tue) 20:54:28 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval.c : fix place of rb_thread_terminate_all()
* eval_thread.c : remove unused functions
* yarv.h : remove GET_VM()
* eval_jump.h : ditto
* insns.def : ditto
* vm_dump.c :
* intern.h : change rb_thread_signal_raise/exit inter
* signal.c : ditto
* thread.c : ditto
```

```
* test/ruby/test_beginendblock.rb : use block with IO
* thread_pthread.h : fix interrupt process
* thread_win32.h : ditto
* yarvcore.c : fix thread free process
* yarvcore.h : remove yarv_vm_t#thread_critical, etc
```

2006-02-21(Tue) 12:42:44 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval_thread.c : remove unused function rb_thread_sc
* thread.c : rename yarv_thread_schedule to rb_thread
* thread.c, eval.c : fix to terminate all thread and
eval.c#ruby_cleanup()
* thread_win32.h : remove native_thread_cleanup()
* thread_pthread.h : ditto
* yarvcore.c : ditto
* yarvtest/test_thread.rb : separete assersions to te
```

2006-02-21(Tue) 02:13:33 +900 Yukihiro Matsumoto
<matz@ruby-lang.org>

```
* parse.y (f_arglist): should set command_start = Qtr
command body. [ruby-talk:180648]
```

2006-02-20(Mon) 20:41:07 +0900 Koichi Sasada
<ko1@atdot.net>

```
* thread.c : fix to synchronize signal_thread_list ac
and fix typo
```


2006-02-20(Mon) 17:54:58 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval_proc.c : remove unused Binding functions and
set is_lambda of Proc used define_method

* yarvcore.c : support Proc#dup/clone, Binding#dup/cl

* sample/test.rb : remove unsupport features (Proc as
```

2006-02-20(Mon) 16:28:59 +0900 Koichi Sasada
<ko1@atdot.net>

```
* common.mk : add a dependency to vm.c on eval_intern

* eval_intern.h : fix to initialize tag->tag

* yarvtest/test_jump.rb : add tests for above

* eval_jump.h : use local variable
```

2006-02-20(Mon) 15:13:24 +0900 Koichi Sasada
<ko1@atdot.net>

```
* benchmark/bm_vm3_thread_create_join.rb : added

* test/yaml/test_yaml.rb : imported from Ruby CVS tru
```

2006-02-20(Mon) 14:49:46 +0900 Minero Aoki
<aamine@loveruby.net>

```
* lib/yaml.rb: imported from Ruby CVS trunk HEAD.

* lib/yaml: ditto.

* ext/syck: ditto.
```

2006-02-20(Mon) 13:58:03 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : support block parameter which is NODE_A
* yarvtest/test_block.rb : add tests for above
* compile.c : fix NODE_DASGN_CURR level check
* compile.c : fix "||=" (at first, check "defined? v
* compile.c : fix NODE_MATCH3 (permute receiver and a
* yarvtest/test_bin.rb : add tests for above
* eval.c : add rb_each()
* test/ruby/test_signal.rb : increment a timeout valu
* thread.c, yarvcore.h : fix "join" flow
* thread_pthread.h : ditto
* thread_win32.h : ditto
* yarvtest/test_thread.rb : add a test for above
* vm.h, vm.c, vm_dump.c, insns.def : add FRAME_MAGIC
support return from lambda (especially retrun from me
by "define_method")
* yarvtest/test_method.rb : add a test for above
* yarvcore.c : remove unused functions
```

2006-02-20(Mon) 11:22:31 +0900 Minero Aoki
<aamine@loveruby.net>

```
* test/ruby/test_eval.rb: now Object#funcall is defin
```

2006-02-20(Mon) 11:04:32 +0900 Minero Aoki
<aamine@loveruby.net>

```
* lib/irb/lc/ja/CVS: removed.
```

2006-02-20(Mon) 10:55:59 +0900 Minero Aoki
<aamine@loveruby.net>

```
* lib/mutex_m.rb: imported from Ruby CVS trunk HEAD.  
* lib/observer.rb: ditto.  
* lib/wsdl: ditto.  
* lib/monitor.rb: ditto (removing Thread.critical=).  
* lib/xsd: ditto.  
* lib/soap: ditto.  
* lib/drb.rb: ditto.  
* lib/drb: ditto.
```

2006-02-20(Mon) 10:49:31 +0900 Minero Aoki
<aamine@loveruby.net>

```
* yarvcore.c (Init_yarvcore): fix typo (duo -> dup).
```

2006-02-19(Sun) 01:27:08 +0900 Koichi Sasada
<ko1@atdot.net>

```
* vm.c : "return" from lambda{} break block  
* eval.c : Unsupport Proc as Binding  
* test/ruby/test_eval.rb : apply above changes  
* yarvcore.c : remove unused function yarv_yield_valu
```

2006-02-18(Sat) 03:19:36 +0900 Koichi Sasada
<ko1@atdot.net>

```
* thread.c, insns.def : fix passing value when thread
* yarvtest/test_thread.rb : add tests for above
```

2006-02-19(Sun) 01:19:42 +0900 Minero Aoki
<aamine@loveruby.net>

```
* lib/thread.rb (SizedQueue): didn't work. This patch
contributed by yukimizake. [yarv-dev:916]
```

2006-02-18(Sat) 03:19:36 +0900 Koichi Sasada
<ko1@atdot.net>

```
* thread.c, insns.def : fix passing value when thread
* yarvtest/test_thread.rb : add tests for above
```

2006-02-18(Sat) 02:40:18 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def, vm.c, vm_macro.def : change BMETHOD algo
([yarv-dev:914])
* yarvtest/test_class.rb : add a test for above
```

2006-02-17(Fri) 23:59:51 +0900 Koichi Sasada
<ko1@atdot.net>

```
* vm.c, yarv.h : change th_invoke_proc() interface
* eval_proc.c : ditto
* signal.c : ditto
* thread.c : ditto
* yarvcore.c : ditto
```

```
* vm_macro.def : ditto and fix NODE_BMETHOD call
* vm.c : change name ("th_set_env()" to "push_frame()
change interface
* insns.def : ditto
* eval.c : remove proc_jump_error()
* benchmark/bm_app_answer.rb : added
* vm_opts.h.base : add optimize option
```

2006-02-17(Fri) 13:37:57 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval.c, ruby.h : add rb_errinfo()
* eval_error.h (error_pos) : fix process order
* bin/erb : imported from ruby 1.9
* bin/irb : ditto
* bin/rdoc : ditto
* bin/ri : ditto
* bin/testrb : ditto
* ext/curses/.cvsignore : ditto
* ext/curses/curses.c : ditto
* ext/curses/depend : ditto
* ext/curses/extconf.rb : ditto
* ext/curses/hello.rb : ditto
* ext/curses/mouse.rb : ditto
* ext/curses/rain.rb : ditto
```

```
* ext/curses/view.rb : ditto
* ext/curses/view2.rb : ditto
* ext/fcntl/.cvsignore : ditto
* ext/fcntl/depend : ditto
* ext/fcntl/extconf.rb : ditto
* ext/fcntl/fcntl.c : ditto
* ext/readline/README : ditto
* ext/readline/README.ja : ditto
* ext/readline/depend : ditto
* ext/readline/extconf.rb : ditto
* ext/readline/readline.c : ditto
* ext/win32ole/.document : ditto
* ext/zlib/doc/zlib.rd : ditto
* ext/zlib/extconf.rb : ditto
* ext/zlib/zlib.c : ditto
* lib/cgi/.document : ditto
* lib/cgi/session.rb : ditto
* lib/cgi/session/pstore.rb : ditto
* lib/shell/builtin-command.rb : ditto
* lib/shell/command-processor.rb : ditto
* lib/shell/error.rb : ditto
* lib/shell/filter.rb : ditto
* lib/shell/process-controller.rb : ditto
```

```
* lib/shell/system-command.rb : ditto
* lib/shell/version.rb : ditto
* lib/xmlrpc/.document : ditto
* lib/xmlrpc/README.rdoc : ditto
* lib/xmlrpc/README.txt : ditto
* lib/xmlrpc/base64.rb : ditto
* lib/xmlrpc/client.rb : ditto
* lib/xmlrpc/config.rb : ditto
* lib/xmlrpc/create.rb : ditto
* lib/xmlrpc/datetime.rb : ditto
* lib/xmlrpc/httpserver.rb : ditto
* lib/xmlrpc/marshal.rb : ditto
* lib/xmlrpc/parser.rb : ditto
* lib/xmlrpc/server.rb : ditto
* lib/xmlrpc/utils.rb : ditto
* rdoc/README : ditto
* rdoc/code_objects.rb : ditto
* rdoc/diagram.rb : ditto
* rdoc/dot/dot.rb : ditto
* rdoc/generators/chm_generator.rb : ditto
* rdoc/generators/html_generator.rb : ditto
* rdoc/generators/ri_generator.rb : ditto
* rdoc/generators/template/chm/chm.rb : ditto
* rdoc/generators/template/html/hefss.rb : ditto
```

```
* rdoc/generators/template/html/html.rb : ditto
* rdoc/generators/template/html/kilmer.rb : ditto
* rdoc/generators/template/html/old_html.rb : ditto
* rdoc/generators/template/html/one_page_html.rb : di
* rdoc/generators/template/xml/rdf.rb : ditto
* rdoc/generators/template/xml/xml.rb : ditto
* rdoc/generators/xml_generator.rb : ditto
* rdoc/markup/sample/rdoc2latex.rb : ditto
* rdoc/markup/sample/sample.rb : ditto
* rdoc/markup/simple_markup.rb : ditto
* rdoc/markup/simple_markup/fragments.rb : ditto
* rdoc/markup/simple_markup/inline.rb : ditto
* rdoc/markup/simple_markup/lines.rb : ditto
* rdoc/markup/simple_markup/preprocess.rb : ditto
* rdoc/markup/simple_markup/to_flow.rb : ditto
* rdoc/markup/simple_markup/to_html.rb : ditto
* rdoc/markup/simple_markup/to_latex.rb : ditto
* rdoc/markup/test/AllTests.rb : ditto
* rdoc/markup/test/TestInline.rb : ditto
* rdoc/markup/test/TestParse.rb : ditto
* rdoc/options.rb : ditto
* rdoc/parsers/parse_c.rb : ditto
* rdoc/parsers/parse_f95.rb : ditto
```



```
* rdoc/parsers/parse_rb.rb : ditto
* rdoc/parsers/parse_simple.rb : ditto
* rdoc/parsers/parserfactory.rb : ditto
* rdoc/rdoc.rb : ditto
* rdoc/ri/ri_cache.rb : ditto
* rdoc/ri/ri_descriptions.rb : ditto
* rdoc/ri/ri_display.rb : ditto
* rdoc/ri/ri_driver.rb : ditto
* rdoc/ri/ri_formatter.rb : ditto
* rdoc/ri/ri_options.rb : ditto
* rdoc/ri/ri_paths.rb : ditto
* rdoc/ri/ri_reader.rb : ditto
* rdoc/ri/ri_util.rb : ditto
* rdoc/ri/ri_writer.rb : ditto
* rdoc/template.rb : ditto
* rdoc/tokenstream.rb : ditto
* rdoc/usage.rb : ditto
* test/xmlrpc/data/bug_bool.expected : ditto
* test/xmlrpc/data/bug_bool.xml : ditto
* test/xmlrpc/data/bug_cdata.expected : ditto
* test/xmlrpc/data/bug_cdata.xml : ditto
* test/xmlrpc/data/bug_covert.expected : ditto
* test/xmlrpc/data/bug_covert.xml : ditto
* test/xmlrpc/data/datetime_iso8601.xml : ditto
```

```
* test/xmlrpc/data/fault.xml : ditto
* test/xmlrpc/data/value.expected : ditto
* test/xmlrpc/data/value.xml : ditto
* test/xmlrpc/data/xml1.expected : ditto
* test/xmlrpc/data/xml1.xml : ditto
* test/xmlrpc/test_datetime.rb : ditto
* test/xmlrpc/test_features.rb : ditto
* test/xmlrpc/test_marshal.rb : ditto
* test/xmlrpc/test_parser.rb : ditto
* test/xmlrpc/test_webrick_server.rb : ditto
* test/xmlrpc/webrick_testing.rb : ditto
* test/zlib/test_zlib.rb : ditto
```

2006-02-17(Fri) 09:41:35 +900 Yukihiro Matsumoto
<matz@ruby-lang.org>

```
* thread.c (sleep_timeval): sleep should always sleep
specified amount of time. [ruby-talk:180067]
```

2006-02-17(Fri) 02:20:32 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval_safe.h, ruby.h : remove ruby_safe_level and add
rb_safe_level() and rb_set_safe_level_force()

* eval.c : use above functions

* eval_jump.h : ditto

* eval_load.c : ditto
```

```
* eval_method.h : ditto
* eval_proc.c : ditto
* eval_thread.c : ditto
* gc.c : ditto
* signal.c : ditto
* variable.c : ditto
* ext/win32ole/win32ole.c : ditto
* vm.c (th_invoke_proc) : save and restore safe level
* yarvtest/test_proc.rb : add tests for above
* thread.c : remove unused functions
```

2006-02-17(Fri) 01:08:23 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c, insns.def : remove a setspecial second u
* eval_load.c : remove unused variable th
* eval_proc.c, yarvcore.c : remove some functions fro
and move to yarvcore.c
* insns.def : fix to delete warnings
* sample/test.rb : comment out Proc#clone tests
* version.c : add constant RUBY_VM_DATE
* vm.c : fix some functions
```

2006-02-16(Thu) 22:58:27 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def, vm.c : use th_yield_setup_args at yield
```

2006-02-16(Thu) 19:51:52 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : fix analysis of block parameter
* disasm.c : remove rb_bug() (temporarily)
* insns.def, vm.c : fix passing block parameter
* sample/test.rb : add "Proc = YARVCore::VM::Proc"
* test/ruby/test_readpartial.rb : disable on mswin32
* test/socket/test_tcp.rb : ditto
* thread.c : fix syntax error (for non GCC)
```

2006-02-15(Wed) 22:34:04 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval_method.h : move rb_clear_cache_by_id position
* thread.c : fix Thread#kill
* test/ruby/test_readpartial.rb : enable tests except
* test/ruby/test_signal.rb : ditto and enable timeout
```

2006-02-15(Wed) 22:13:29 +0900 Minero Aoki
<aamine@loveruby.net>

```
* lib/runit: forgot to commit.
```

2006-02-15(Wed) 22:12:25 +0900 Minero Aoki
<aamine@loveruby.net>

```
* lib/weakref.rb: do not use Thread.critical=.  
* lib/singleton.rb: ditto.  
* lib/timeout.rb: ditto.  
* lib/thread.rb: ditto.  
* test/inlinetest.rb: forgot to commit.
```

2006-02-15(Wed) 21:34:17 +0900 Minero Aoki
<aamine@loveruby.net>

```
* test/test_pp.rb: imported from Ruby CVS trunk HEAD.  
* test/test_shellwords.rb: ditto.  
* test/test_set.rb: ditto.  
* test/test_time.rb: ditto.  
* test/test_ipaddr.rb: ditto.  
* test/test_prettyprint.rb: ditto.  
* test/test_tsort.rb: ditto.  
* test/strscan: ditto.  
* test/testunit: ditto.
```

2006-02-15(Wed) 20:03:21 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval_method.h : duplicate NODE_METHOD at make an al  
* yarvtest/test_method.rb : add a test for above
```

2006-02-15(Wed) 19:48:59 +0900 Minero Aoki
<aamine@loveruby.net>

```
* test/rss: imported from Ruby CVS trunk HEAD.
```

2006-02-15(Wed) 19:47:51 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def, compile.c, vm.c : remove methoddef, sing  
instructions and make new insn definemethod  
  
* yarvcore.c : set toplevel visibility to private
```

2006-02-15(Wed) 17:39:16 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval_intern.h :  
  
* eval_jump.h, vm.c : localjump_error() and jump_tag_  
move to th_localjump_error and th_jump_tag_but_local_  
  
* eval.c : ditto  
  
* eval_load.c : ditto  
  
* insns.def : ditto  
  
* vm.c : ditto  
  
* vm.c (th_make_jump_tag_but_local_jump) : added  
  
* opt_insn_unif.def : fix indnet (revert change)  
  
* opt_operand.def : ditto  
  
* rb/insns2vm.rb : fix error message  
  
* thread.c : raise exception at join if illegal local
```

2006-02-15(Wed) 14:21:45 +900 Yukihiro Matsumoto
<matz@ruby-lang.org>

```
* ChangeLog: add local variables line to support Emac
```

```
* eval.c (rb_obj_instance_exec): add new method from
* eval.c (rb_mod_module_exec): ditto.
* eval.c (yield_under_i): should not pass self as an
  the block for instance_eval. [ruby-core:07364]
* eval.c (rb_obj_instance_eval): should be no singlet
  true, false, and nil. [ruby-dev:28186]
```

2006-02-14(Tue) 19:30:20 +0900 Koichi Sasada
<ko1@atdot.net>

```
* array.c : fix indent
* eval.c : fix block_given
* gc.c : add STACK_START and use it as a substitute f
  rb_gc_stack_start
* vm.c : fix to raise error if th_yield doesn't have
* yarvcore.c : fix to skip iseq mark array at Objects
```

2006-02-14(Tue) 18:15:03 +0900 Koichi Sasada
<ko1@atdot.net>

```
* configure.in : enable pthread by default
* ascii.c : import ruby 1.9 HEAD
* bignum.c : ditto
* compar.c : ditto
* configure : ditto
* defines.h : ditto
* dln.c : ditto
```

```
* dlh.h : ditto
* enum.c : ditto
* enumerator.c : ditto
* euc_jp.c : ditto
* ext/win32ole/tests/testWIN32OLE.rb : ditto
* ext/win32ole/win32ole.c : ditto
* file.c : ditto
* hash.c : ditto
* io.c : ditto
* lex.c : ditto
* lib/irb/init.rb : ditto
* lib/rexml/document.rb : ditto
* main.c : ditto
* marshal.c : ditto
* math.c : ditto
* missing.h : ditto
* object.c : ditto
* oniguruma.h : ditto
* pack.c : ditto
* process.c : ditto
* random.c : ditto
* range.c : ditto
* rb/ir.rb : ditto
* re.c : ditto
```



```
* regcomp.c : ditto
* regenc.c : ditto
* regenc.h : ditto
* regerror.c : ditto
* regexec.c : ditto
* regint.h : ditto
* regparse.c : ditto
* regparse.h : ditto
* ruby.c : ditto
* ruby.h : ditto
* rubyio.h : ditto
* sjis.c : ditto
* sprintf.c : ditto
* st.c : ditto
* st.h : ditto
* struct.c : ditto
* test/ruby/envutil.rb : ditto
* test/ruby/test_struct.rb : ditto
* time.c : ditto
* utf8.c : ditto
* util.c : ditto
* util.h : ditto
* version.h : ditto
```

```
* win32/Makefile.sub : ditto
* win32/win32.c : ditto
```

2006-02-14(Tue) 16:40:01 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.c, eval_proc.c : fix rb_proc_arity
* eval.c : declare funcall same as send (temporarily)
* lib/thread.rb : added
* test/pathname/test_pathname.rb : imported from ruby
* test/scanf/data.txt : ditto
* test/scanf/test_scanf.rb : ditto
* test/scanf/test_scanfblocks.rb : ditto
* test/scanf/test_scanfio.rb : ditto
* test/socket/test_socket.rb : ditto
* test/socket/test_tcp.rb : ditto
* test/socket/test_udp.rb : ditto
* test/socket/test_unix.rb : ditto
* test/stringio/test_stringio.rb : ditto
* test/uri/test_common.rb : ditto
* test/uri/test_ftp.rb : ditto
* test/uri/test_generic.rb : ditto
* test/uri/test_http.rb : ditto
* test/uri/test_ldap.rb : ditto
* test/uri/test_mailto.rb : ditto
```

2006-02-14(Tue) 15:59:28 +0900 Koichi Sasada
<ko1@atdot.net>

```
* thread.c : Change Thread.critical warning message
* lib/webrick.rb : imported from ruby 1.9
* lib/webrick/accesslog.rb : ditto
* lib/webrick/cgi.rb : ditto
* lib/webrick/compat.rb : ditto
* lib/webrick/config.rb : ditto
* lib/webrick/cookie.rb : ditto
* lib/webrick/htmlutils.rb : ditto
* lib/webrick/httpauth.rb : ditto
* lib/webrick/httpauth/authenticator.rb : ditto
* lib/webrick/httpauth/basicauth.rb : ditto
* lib/webrick/httpauth/digestauth.rb : ditto
* lib/webrick/httpauth/htdigest.rb : ditto
* lib/webrick/httpauth/htgroup.rb : ditto
* lib/webrick/httpauth/htpasswd.rb : ditto
* lib/webrick/httpauth/userdb.rb : ditto
* lib/webrick/httpproxy.rb : ditto
* lib/webrick/httprequest.rb : ditto
* lib/webrick/httpresponse.rb : ditto
* lib/webrick/https.rb : ditto
* lib/webrick/httpserver.rb : ditto
* lib/webrick/httpservlet.rb : ditto
```

```
* lib/webrick/httpservlet/abstract.rb : ditto
* lib/webrick/httpservlet/cgi_runner.rb : ditto
* lib/webrick/httpservlet/cgihandler.rb : ditto
* lib/webrick/httpservlet/erbhandler.rb : ditto
* lib/webrick/httpservlet/filehandler.rb : ditto
* lib/webrick/httpservlet/prochandler.rb : ditto
* lib/webrick/httpstatus.rb : ditto
* lib/webrick/httputils.rb : ditto
* lib/webrick/httpversion.rb : ditto
* lib/webrick/log.rb : ditto
* lib/webrick/server.rb : ditto
* lib/webrick/ssl.rb : ditto
* lib/webrick/utils.rb : ditto
* lib/webrick/version.rb : ditto
```

2006-02-14(Tue) 14:55:51 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c, insns.def : support "defined?($1)", ...
* yarvtest/test_syntax.rb : add a test for above
* rb/makedocs.rb : fix template directory path
* vm.c : fix to handle break from proc
```

2006-02-14(Tue) 12:42:59 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval.c : fix rb_iterate hook
```

```
* yarvtest/test_block.rb : add a tests for above
* vm.c : remove unused comment
```

2006-02-14(Tue) 12:01:06 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval.c : fix to check passed block at block_given_p
* eval_proc.c : fix to pass block at Method#call
* runruby.rb : fix to apply ruby
* test/runner.rb : GC.stress (comment out)
* vm.c : fix indnet
```

2006-02-14(Tue) 08:04:33 +0900 Minero Aoki
<aamine@loveruby.net>

```
* lib/tempfile.rb: use Mutex instead of Thread.critic
* lib/rss/dublincore.rb: |x,| -> |x,_| to avoid YARV
* lib/rexml: imported from ruby CVS trunk HEAD.
* test/digest: ditto.
* test/fileutils: ditto.
* test/ostruct: ditto.
* test/erb: ditto.
* test/optparse: ditto.
* test/ruby/test_signal.rb: turn off a test to avoid
(tmp).
```

2006-02-14(Tue) 07:52:03 +0900 Minero Aoki

<aamine@loveruby.net>

```
* test/digest: imported from ruby CVS trunk HEAD.  
* test/fileutils: ditto.  
* test/ostruct: ditto.  
* test/erb: ditto.  
* test/optparse: ditto.
```

2006-02-14(Tue) 06:26:21 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c, parse.y : support BEGIN{} (remove local  
* test/ruby/beginmainend.rb : fix to apply YARV's spe  
* test/ruby/test_beginendblock.rb : enable BEGIN{} te  
* signal.c : exit at double segv  
* insns.def (preexe) : remove instruction "preexe"
```

2006-02-14(Tue) 05:53:56 +0900 Minero Aoki
<aamine@loveruby.net>

```
* eval.c (ruby_cleanup): th->errinfo contains a NODE  
break'ing, check it before refering klass.
```

2006-02-14(Tue) 05:45:07 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def : fix stack calc of send  
* sample/test.rb : remove SEGV causing code
```

2006-02-14(Tue) 02:24:21 +0900 Minero Aoki

<aamine@loveruby.net>

```
* test/ruby/test_module.rb: list order is not a matte  
* test/csv: imported from ruby CVS trunk HEAD.
```

2006-02-14(Tue) 02:06:25 +0900 Minero Aoki
<aamine@loveruby.net>

```
* test/ruby/test_beginendblock.rb: unlock tests.  
* test/ruby/beginmainend.rb: new file (imported from  
trunk HEAD).  
* test/ruby/endblockwarn.rb: new file (imported from  
trunk HEAD).  
* test/ruby/test_file.rb: new file (imported from rub  
HEAD).
```

2006-02-14(Tue) 01:42:11 +0900 Koichi Sasada
<ko1@atdot.net>

```
* error.c : fix include file positon  
* test/ruby/test_signal.rb : skip test_exit_action on
```

2006-02-14(Tue) 01:36:57 +0900 Minero Aoki
<aamine@loveruby.net>

```
* test/ruby/test_class.rb: new file (imported from ru
```

2006-02-14(Tue) 01:32:23 +0900 Minero Aoki
<aamine@loveruby.net>

```
* test/ruby/test_module.rb: ignore PP mixins.
```

2006-02-14(Tue) 01:24:56 +0900 Minero Aoki
<aamine@loveruby.net>

```
* test/ruby/test_lambda.rb: removed (->(){...} syntax  
obsolete).
```

2006-02-14(Tue) 01:20:54 +0900 Minero Aoki
<aamine@loveruby.net>

```
* test/ruby/test_module.rb: import many tests from ru
```

2006-02-14(Tue) 01:06:57 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : fix to avoid stack consistency error  
* yarvtest/test_exception.rb : add a test for above
```

2006-02-14(Tue) 00:42:47 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.h, vm_macro.def : rename VM_CALL_SUPER to  
* insns.def (send) : set a flag of super as fcall  
* yarvtest/test_class.rb : add a test for above
```

2006-02-14(Tue) 00:31:24 +0900 Minero Aoki
<aamine@loveruby.net>

```
* test/ruby/test_eval.rb: fix typo.  
* test/ruby/test_signal.rb: unlock tests.
```

2006-02-13(Mon) 23:53:27 +0900 Koichi Sasada
<ko1@atdot.net>


```
* insns.def, vm_macro.def : fix NODE_ZSUPER dispatch  
fix error message when super without suitable method  
  
* yarvcore.h : add VM_CALL_SUPER definition  
  
* yarvtest/test_method.rb : add a test of Module#priv
```

2006-02-13(Mon) 22:49:42 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def : traverse all iseq to find super method  
  
* yarvtest/test_class.rb : add a test for above  
  
* yarvcore.c : add clear iseq->defined_method_id  
  
* signal.c : fix to prohibit double segv handler kick
```

2006-02-13(Mon) 22:09:12 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : support NODE_DECL, NODE_CLASS with NODE  
  
* yarvtest/test_class.rb : add tests for above
```

2006-02-13(Mon) 21:20:57 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : fix indent  
  
* compile.c : fix to prohibit "redo" from eval expres
```

2006-02-13(Mon) 20:36:06 +0900 Koichi Sasada
<ko1@atdot.net>

```
* vm.c : fix constant search bug ([yarv-dev:788])
```

```
* yarvtest/test_class.rb : add a test of [yarv-dev:78
```

2006-02-13(Mon) 18:09:28 +0900 Koichi Sasada
<ko1@atdot.net>

```
* test/ruby/test_clone.rb : enable tests with Class#clone
```

```
* test/ruby/test_marshal.rb : ditto
```

2006-02-13(Mon) 17:42:37 +0900 Koichi Sasada
<ko1@atdot.net>

```
* class.c : support Class#clone
```

```
* compile.c, insns.def : remove popcref
```

```
* yarvcore.h, vm.c, insns.def : remove yarv_thread_t#
```

```
* eval.c, eval_intern.h, eval_load.c : ditto
```

```
* yarvtest/test_class.rb : add tests for singleton class
```

```
* gc.c : remove "FRAME *" unused variable
```

```
* insnhelper.h : fix COPY_CREF
```

```
* rb/mklog.rb : add default message
```

```
* vm_macro.def : support NODE_ZSUPER as method type
```

2006-02-13(Mon) 00:11:17 +0900 Koichi Sasada
<ko1@atdot.net>

```
* blockinlining.c : refactoring with CFLAGS+=-Wunused
```

```
* eval.c : ditto
```

```
* eval_intern.h : ditto
```

```
* eval_load.c : ditto
```

```
* eval_method.h : ditto
* eval_proc.c : ditto
* eval_thread.c : ditto
* insns.def : ditto
* parse.y : ditto
* thread.c : ditto
* vm.c : ditto
```

2006-02-13(Mon) 02:32:34 +0900 Minero Aoki
<aamine@loveruby.net>

```
* test/ruby/test_const.rb: show better message.
* test/ruby/test_eval.rb: ditto.
* test/ruby/test_module.rb: new file.
```

2006-02-12(Sun) 22:22:35 +0900 Koichi Sasada
<ko1@atdot.net>

```
* array.c : revert last commit
* ascii.c : ditto
* bignum.c : ditto
* class.c : ditto
* compar.c : ditto
* defines.h : ditto
* dir.c : ditto
* dln.c : ditto
```

```
* dlh.h : ditto
* enum.c : ditto
* enumerator.c : ditto
* error.c : ditto
* euc_jp.c : ditto
* file.c : ditto
* gc.c : ditto
* hash.c : ditto
* intern.h : ditto
* io.c : ditto
* lex.c : ditto
* main.c : ditto
* marshal.c : ditto
* math.c : ditto
* missing.h : ditto
* node.h : ditto
* numeric.c : ditto
* object.c : ditto
* oniguruma.h : ditto
* pack.c : ditto
* prec.c : ditto
* process.c : ditto
* random.c : ditto
* range.c : ditto
```

```
* rb/mklog.rb : ditto
* re.c : ditto
* regcomp.c : ditto
* regenc.c : ditto
* regenc.h : ditto
* regerror.c : ditto
* regex.h : ditto
* regexec.c : ditto
* regint.h : ditto
* regparse.c : ditto
* regparse.h : ditto
* ruby.c : ditto
* ruby.h : ditto
* rubyio.h : ditto
* rubysig.h : ditto
* signal.c : ditto
* sjis.c : ditto
* sprintf.c : ditto
* st.c : ditto
* st.h : ditto
* string.c : ditto
* struct.c : ditto
* time.c : ditto
```

```
* utf8.c : ditto
* util.c : ditto
* util.h : ditto
* variable.c : ditto
* version.c : ditto
```

2006-02-12(Sun) 21:33:10 +0900 Koichi Sasada
<ko1@atdot.net>

```
* array.c : fix to ruby's indent
* ascii.c : ditto
* bignum.c : ditto
* blockinlining.c : ditto
* call_cfunc.h : ditto
* class.c : ditto
* compar.c : ditto
* compile.c : ditto
* compile.h : ditto
* debug.c : ditto
* debug.h : ditto
* defines.h : ditto
* dir.c : ditto
* disasm.c : ditto
* dln.c : ditto
* dln.h : ditto
```

```
* enum.c : ditto
* enumerator.c : ditto
* error.c : ditto
* euc_jp.c : ditto
* eval.c : ditto
* eval_error.h : ditto
* eval_intern.h : ditto
* eval_jump.h : ditto
* eval_load.c : ditto
* eval_method.h : ditto
* eval_proc.c : ditto
* eval_safe.h : ditto
* eval_thread.c : ditto
* file.c : ditto
* gc.c : ditto
* hash.c : ditto
* insnhelper.h : ditto
* insns.def : ditto
* intern.h : ditto
* io.c : ditto
* lex.c : ditto
* main.c : ditto
* marshal.c : ditto
* math.c : ditto
```

```
* missing.h : ditto
* node.h : ditto
* numeric.c : ditto
* object.c : ditto
* oniguruma.h : ditto
* opt_insn_unif.def : ditto
* opt_operand.def : ditto
* pack.c : ditto
* prec.c : ditto
* process.c : ditto
* random.c : ditto
* range.c : ditto
* re.c : ditto
* re.h : ditto
* regcomp.c : ditto
* regenc.c : ditto
* regenc.h : ditto
* regerror.c : ditto
* regex.h : ditto
* regexec.c : ditto
* regint.h : ditto
* regparse.c : ditto
* regparse.h : ditto
```



```
* ruby.c : ditto
* ruby.h : ditto
* rubyio.h : ditto
* rubysig.h : ditto
* signal.c : ditto
* sjis.c : ditto
* sprintf.c : ditto
* st.c : ditto
* st.h : ditto
* string.c : ditto
* struct.c : ditto
* test.rb : ditto
* thread.c : ditto
* thread_pthread.h : ditto
* thread_win32.h : ditto
* time.c : ditto
* utf8.c : ditto
* util.c : ditto
* util.h : ditto
* variable.c : ditto
* version.c : ditto
* vm.c : ditto
* vm.h : ditto
* vm_dump.c : ditto
```

```
* vm_evalbody.h : ditto
* vm_macro.def : ditto
* yarv.h : ditto
* yarv_version.h : ditto
* yarvcore.c : ditto
* yarvcore.h : ditto
```

2006-02-12(Sun) 15:53:21 +0900 Koichi Sasada
<ko1@atdot.net>

```
* lib/abbrev.rb : added
* lib/base64.rb : ditto
* lib/cgi-lib.rb : ditto
* lib/csv.rb : ditto
* lib/date2.rb : ditto
* lib/eregex.rb : ditto
* lib/ipaddr.rb : ditto
* lib/irb.rb : ditto
* lib/irb/cmd/chws.rb : ditto
* lib/irb/cmd/fork.rb : ditto
* lib/irb/cmd/help.rb : ditto
* lib/irb/cmd/load.rb : ditto
* lib/irb/cmd/nop.rb : ditto
* lib/irb/cmd/pushws.rb : ditto
* lib/irb/cmd/subirb.rb : ditto
```

```
* lib/irb/completion.rb : ditto
* lib/irb/context.rb : ditto
* lib/irb/ext/change-ws.rb : ditto
* lib/irb/ext/history.rb : ditto
* lib/irb/ext/loader.rb : ditto
* lib/irb/ext/math-mode.rb : ditto
* lib/irb/ext/multi-irb.rb : ditto
* lib/irb/ext/save-history.rb : ditto
* lib/irb/ext/tracer.rb : ditto
* lib/irb/ext/use-loader.rb : ditto
* lib/irb/ext/workspaces.rb : ditto
* lib/irb/extend-command.rb : ditto
* lib/irb/frame.rb : ditto
* lib/irb/help.rb : ditto
* lib/irb/init.rb : ditto
* lib/irb/input-method.rb : ditto
* lib/irb/lc/error.rb : ditto
* lib/irb/lc/help-message : ditto
* lib/irb/lc/ja/ CVS/Entries : ditto
* lib/irb/lc/ja/ CVS/Repository : ditto
* lib/irb/lc/ja/ CVS/Root : ditto
* lib/irb/lc/ja/error.rb : ditto
* lib/irb/lc/ja/help-message : ditto
```

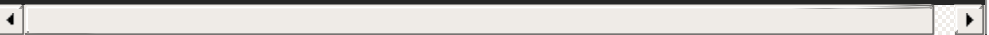
```
* lib/irb/locale.rb : ditto
* lib/irb/notifier.rb : ditto
* lib/irb/output-method.rb : ditto
* lib/irb/ruby-lex.rb : ditto
* lib/irb/ruby-token.rb : ditto
* lib/irb/slex.rb : ditto
* lib/irb/version.rb : ditto
* lib/irb/workspace.rb : ditto
* lib/irb/ws-for-case-2.rb : ditto
* lib/irb/xmp.rb : ditto
* lib/jcode.rb : ditto
* lib/logger.rb : ditto
* lib/mailread.rb : ditto
* lib/mathn.rb : ditto
* lib/parsedate.rb : ditto
* lib/pathname.rb : ditto
* lib/ping.rb : ditto
* lib/pstore.rb : ditto
* lib/resolv-replace.rb : ditto
* lib/resolv.rb : ditto
* lib/rss.rb : ditto
* lib/rss/0.9.rb : ditto
* lib/rss/1.0.rb : ditto
* lib/rss/2.0.rb : ditto
```

```
* lib/rss/content.rb : ditto
* lib/rss/converter.rb : ditto
* lib/rss/dublincore.rb : ditto
* lib/rss/image.rb : ditto
* lib/rss/maker.rb : ditto
* lib/rss/maker/0.9.rb : ditto
* lib/rss/maker/1.0.rb : ditto
* lib/rss/maker/2.0.rb : ditto
* lib/rss/maker/base.rb : ditto
* lib/rss/maker/content.rb : ditto
* lib/rss/maker/dublincore.rb : ditto
* lib/rss/maker/image.rb : ditto
* lib/rss/maker/syndication.rb : ditto
* lib/rss/maker/taxonomy.rb : ditto
* lib/rss/maker/trackback.rb : ditto
* lib/rss/parser.rb : ditto
* lib/rss/rexmlparser.rb : ditto
* lib/rss/rss.rb : ditto
* lib/rss/syndication.rb : ditto
* lib/rss/taxonomy.rb : ditto
* lib/rss/trackback.rb : ditto
* lib/rss/utils.rb : ditto
* lib/rss/xml-styleSheet.rb : ditto
```

```
* lib/rss/xmlparser.rb : ditto
* lib/rss/xmlscanner.rb : ditto
* lib/rubyunit.rb : ditto
* lib/scanf.rb : ditto
* lib/shell.rb : ditto
* lib/singleton.rb : ditto
* lib/tsort.rb : ditto
* lib/weakref.rb : ditto
* eval_jump.c : removed
```

2006-02-12(Sun) 15:39:09 +0900 Koichi Sasada
<ko1@atdot.net>

```
* parse.y : fix to remove including env.h
* yarvtest/test_exception.rb : fix syntax (add 'end')
```



2006-02-12(Sun) 15:14:44 +0900 Koichi Sasada
<ko1@atdot.net>

```
* env.h : removed
* common.mk : remove env.h dependency
* compile.c, eval_intern.h : remove include env.h
* vm.c : ditto
* ruby.h, gc.c, error.c : remove T_SCOPE, T_VARMAP
* parse.y, eval.c : use rb_parse_in_eval() instead of
* yarvcore.c, yarvcore.h : add a prase_in_eval member
* insns.def : add push value to throw instruction
```

```
for stack consistency

* yarvtest/test_exception.rb : add a test for above

* test/ruby/test_gc.rb : fix typo
```

2006-02-12(Sun) 05:05:02 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval.c, eval_intern.h, eval_load.c, eval_proc.c, no  
insnhelper.h, insns.def, vm.c, yarvcore.c, yarvcore.h  
change cref data structure and unify ruby_class and r  
and some refactoring
```

2006-02-11(Sat) 23:41:11 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def (methoddef) : fix method declaration in m  
* thread.c : Thread.critical to show warning (no effe
```

2006-02-11(Sat) 20:20:18 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def : fix [yarv-dev:831]  
* yarvtest/test_class.rb : add a test for above
```

2006-02-11(Sat) 14:29:01 +0900 Koichi Sasada
<ko1@atdot.net>

```
* rb/mklog.rb : use svk  
* error.c : remove newline  
* eval.c (rb_block_call) : added  
* eval_thread.c : remove some unused functions, comme
```

```
* thread.c : add comments (move from eval_thread.c) a
* thread.c (rb_thread_select) : supported
* thread_pthread.h (native_mutex_trylock) : added (ma
* thread_win32.h (native_mutex_trylock) : added
* yarvcore.c : remove unused code
* array.c : import ruby 1.9
* compar.c : ditto
* dln.c : ditto
* enum.c : ditto
* enumerator.c : ditto
* ext/digest/digest.c : ditto
* ext/digest/digest.h : ditto
* ext/digest/sha2/sha2.c : ditto
* ext/etc/etc.c : ditto
* ext/win32ole/win32ole.c : ditto
* hash.c : ditto
* intern.h : ditto
* io.c : ditto
* main.c : ditto
* missing.h : ditto
* missing/flock.c : ditto
* missing/isinf.c : ditto
* missing/vsnprintf.c : ditto
```



```
* lib/cgi.rb : ditto
* lib/complex.rb : ditto
* lib/delegate.rb : ditto
* lib/erb.rb : ditto
* lib/fileutils.rb : ditto
* lib/matrix.rb : ditto
* lib/mkrf.rb : ditto
* lib/optparse.rb : ditto
* lib/ostruct.rb : ditto
* lib/pp.rb : ditto
* lib/timeout.rb : ditto
* lib/tmpdir.rb : ditto
* lib/test/unit/autorunner.rb : ditto
* node.h : ditto
* object.c : ditto
* parse.y : ditto
* ruby.c : ditto
* sample/test.rb : ditto
* sprintf.c : ditto
* st.c : ditto
* test/ruby/test_whileuntil.rb : ditto
* test/runner.rb : ditto
* time.c : ditto
* lib/net/.document : added
```

```
* lib/net/ftp.rb : ditto
* lib/net/http.rb : ditto
* lib/net/https.rb : ditto
* lib/net/imap.rb : ditto
* lib/net/pop.rb : ditto
* lib/net/protocol.rb : ditto
* lib/net/smtp.rb : ditto
* lib/net/telnet.rb : ditto
* lib/open-uri.rb : ditto
```

2006-02-10(Fri) 08:07:34 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c, insns.def, yarvcore.h : support defined?  
defined?(protected_method) (separate DEFINE_METHOD /  
* yarvtest/test_syntax.rb : add a test for above  
* compile.c (iseq_compile_each) : fix NODE_RETURN bug  
(double ensure invoke)  
* yarvtest/test_flow.rb : add a test for above  
* eval.c (get_errinfo) : fix to search $!  
* yarvtest/test_exception.rb : add tests for above  
* eval_safe.h : support $SAFE  
* ext/socket/socket.c : import ruby 1.9  
* gc.c (gc_mark_children) : fix making T_VALUE  
* test/ruby/test_gc.rb : use GC.stress
```

```
* signal.c (sighandler) : send interrupt signal if th
* test/ruby/test_proc.rb : remove assert false
* test/ruby/test_readpartial.rb : change fail message
* test/ruby/test_signal.rb : remove assert false
* thread.c (thread_start_func_2) : set local_lfp/loca
at thread creation
* thread_pthread.h : export native_thread_interrupt
* thread_win32.h : export native_thread_interrupt
* version.h : import ruby 1.9
* vm.c (lfp_svar), yarvcore.h : fix to use Thread loc
* yarvtest/test_thread.rb : add a test for above
* win32/Makefile.sub : import ruby 1.9
* win32/dir.h : ditto
* win32/setup.mak : ditto
* win32/win32.c : ditto
* yarvtest/yarvtest.rb : fix to remove using ARGV
```

2006-02-10(Fri) 01:04:58 +0900 Yukihiro Matsumoto
<matz@ruby-lang.org>

```
* gc.c (rb_gc_call_finalizer_at_exit): turn on during
invoking finalizers.
* gc.c (rb_gc_finalize_deferred): ditto.
```

2006-02-08(Wed) 23:17:44 +0900 Minero Aoki
<aamine@loveruby.net>

```
* test/ruby/test_proc.rb: method names were wrongly d
```

2006-02-08(Wed) 21:30:01 +0900 Minero Aoki
<aamine@loveruby.net>

```
* ext/nkf: added (imported from ruby CVS trunk HEAD).  
* ext/nkf/depend: new file (rev 1.5).  
* ext/nkf/extconf.rb: new file (rev 1.2).  
* ext/nkf/nkf.c: new file (rev 1.12).  
* ext/nkf/test.rb: new file (rev 1.7).  
* ext/nkf/nkf-utf8/nkf.c: new file (rev 1.17).  
* ext/nkf/nkf-utf8/config.h: new file (rev 1.4).  
* ext/nkf/nkf-utf8/utf8tbl.c: new file (rev 1.6).  
* ext/nkf/lib/kconv.rb: new file (rev 1.13).  
* test/nkf: added (imported from ruby CVS trunk HEAD)  
* test/nkf/test_kconv.rb: new file (rev 1.1).  
* test/nkf/test_nkf.rb: new file (rev 1.1).
```

2006-02-08(Wed) 21:07:36 +0900 Minero Aoki
<aamine@loveruby.net>

```
* lib/find.rb: new file (imported from ruby CVS trunk  
rev 1.15).
```

2006-02-07(Tue) 17:58:18 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c, insns.def : support BEGIN{} and add pree
```

```
* insns.def : fix getspecial/setspecial instructions
to catch up svar change

* test/ruby/test_system.rb : remove stopper

* thread.c (rb_thread_fd_writable) : add a debug outp

* thread.c (rb_thread_wait_fd) : add a debug output

* vm.c (lfp_svar) : refactoring and fix some problems

* vm_dump.c (yarv_bug) : add branch

* yarv.h : remove unused declarations

* yarvcore.c (vm_free) : VM object should not free by
```

2006-02-07(Tue) 14:42:25 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval.c, eval_load.c : remove rb_thread_start_1()

* eval.c : fix some prototypes and indents

* eval_thread.c, thread.c : move some functions
from eval_thread.c to thread.c

* signal.c (sighandler) : add line braek in error mes

* yarvcore.c, yarvcore.h, thread.c : support ThreadGr

* ruby.h, gc.c, vm.c : make new basic type RValue and
RValue includes three values in itself. RValue is us
svar
```

2006-02-06(Mon) 23:51:41 +0900 Minero Aoki
<aamine@loveruby.net>

```
* test/ruby/test_hash.rb: import many tests from rubi
```

2006-02-04(Sat) 18:36:41 +0900 Minero Aoki
<aamine@loveruby.net>

```
* test/ruby/test_array.rb: import many tests from rub
```

2006-02-04(Sat) 17:47:44 +0900 Minero Aoki
<aamine@loveruby.net>

```
* test/ruby/test_signal.rb (test_exit_action): lib/ti  
not implemented yet.
```

2006-02-04(Sat) 17:42:31 +0900 Minero Aoki
<aamine@loveruby.net>

```
* test/ruby/test_readpartial.rb: lib/timeout.rb is no  
yet.
```

2006-02-04(Sat) 16:22:38 +0900 Minero Aoki
<aamine@loveruby.net>

```
* test/ruby/test_pipe.rb: remove useless require.  
* test/ruby/test_signal.rb: turn off the test case wh  
segmentation fault (tmp).
```

2006-02-04(Sat) 08:19:50 +0900 Koichi Sasada
<ko1@atdot.net>

```
* common.mk : add dependency to yarvcore.h on signal.  
* compile.c (iseq_compile_each) : fix [yarv-dev:795]  
(prohibit "break", "next" jump from eval)  
* eval.c : fix indent  
* eval_thread.c, thread.c : remove some functions and
```

```
* insns.def, vm.c : fix [yarv-dev:799] and [yarv-dev:
* yarvtest/test_class.rb : add a test for above
* test/ruby/test_gc.rb : remove GC.debug_flag control
* test/ruby/test_readpartial.rb : disable
* test/ruby/test_signal.rb : disable
* thread.c : fix thread_debug() and many bugs
* thread.c (yarv_thread_s_new) : move living_threads
* thread.c (yarv_thread_join) : fix
* thread_pthread.h : add type native_thread_data_t (d
and support interrupt blocking thread
* thread_pthread.h (native_thread_apply_priority) : a
* thread_win32.h : add type native_thread_data_t (dum
and support interrupt blocking thread
* yarvcore.h : use win32 thread system on cygwin and
some struct members
* yarvtest/test_thread.rb : added
```

2006-02-03(Fri) 00:08:09 +0900 Minero Aoki
<aamine@loveruby.net>

```
* test/ruby/test_string.rb: import many tests from ru
```

2006-02-02(Thu) 23:20:13 +0900 Minero Aoki
<aamine@loveruby.net>

```
* test/ruby/envutil.rb: new file (imported from ruby
* test/ruby/marshaltestlib.rb: ditto.
```

```
* test/ruby/test_array.rb: ditto.  
* test/ruby/test_beginendblock.rb: ditto.  
* test/ruby/test_clone.rb: ditto.  
* test/ruby/test_dir.rb: ditto.  
* test/ruby/test_env.rb: ditto.  
* test/ruby/test_file.rb: ditto.  
* test/ruby/test_float.rb: ditto.  
* test/ruby/test_fnmatch.rb: ditto.  
* test/ruby/test_hash.rb: ditto.  
* test/ruby/test_io.rb: ditto.  
* test/ruby/test_marshal.rb: ditto.  
* test/ruby/test_math.rb: ditto.  
* test/ruby/test_pack.rb: ditto.  
* test/ruby/test_path.rb: ditto.  
* test/ruby/test_pipe.rb: ditto.  
* test/ruby/test_rand.rb: ditto.  
* test/ruby/test_range.rb: ditto.  
* test/ruby/test_readpartial.rb: ditto.  
* test/ruby/test_regexp.rb: ditto.  
* test/ruby/test_settracefunc.rb: ditto.  
* test/ruby/test_signal.rb: ditto.  
* test/ruby/test_sprintf.rb: ditto.  
* test/ruby/test_string.rb: ditto.  
* test/ruby/test_stringchar.rb: ditto.
```



```
* test/ruby/test_struct.rb: ditto.  
* test/ruby/test_symbol.rb: ditto.  
* test/ruby/test_system.rb: ditto.  
* test/ruby/test_time.rb: ditto.  
* test/ruby/ut_eof.rb: ditto.
```

2006-02-02(Thu) 22:53:44 +0900 Minero Aoki
<aamine@loveruby.net>

```
* test/ruby/test_proc.rb: test [yarv-dev:777].
```

2006-02-01(Wed) 03:51:39 +0900 Koichi Sasada
<ko1@atdot.net>

```
* gc.c : add GC.debug_flag= method  
* insns.def : support method definition in method  
* yarvtest/test_method.rb : add tests for above
```

2006-01-29(Sun) 11:40:26 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval_proc.c (proc_alloc) : fix [yarv-dev:777]  
* yarvtest/test_proc.rb : add a test for above  
* insns.def : fix [yarv-dev:782] and add YARV_CHECK_I  
* yarvtest/test_class.rb : add a test for above  
* thread_win32.h : fix [yarv-dev-en:23]  
* vm.c (th_call0) : add YARV_CHECK_INTS()
```

2006-01-09(Mon) 11:56:34 +0900 Minero Aoki
<aamine@loveruby.net>

```
* yarvcore.h: add prototype (remove warning).
* vm.c (th_invoke_proc): make save variables volatile
* eval.c (eval): initialize local variables (remove w
* eval_thread.c (rb_exec_recursive): ditto.
* yarvcore.c (thread_mark): ditto.
* vm.c (th_invoke_proc): ditto.
* eval.c: remove useless prototypes.
```

2006-01-09(Mon) 10:25:12 +0900 Minero Aoki
<aamine@loveruby.net>

```
* eval_thread.c: rb_thread_join is required to build
Linux.
* compile.c: unify coding style.
* yarvcore.c: ditto.
```

2006-01-06(Fri) 09:21:34 +0900 Minero Aoki
<aamine@loveruby.net>

```
* vm.c: coding style change only.
```

2006-01-04(Wed) 14:12:47 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval.c (ruby_init), eval_intern.h : use POP_TAG_INI
* eval_thread.c : remove unused functions and comment
```

```
* intern.h : expose rb_make_exception()
* signal.c : support signal
* thread.c (yarv_thread_execute_interrupts) : added
* thread_pthread.h (thread_timer) : set interrupt_flag
current running threads
* vm.c (th_invoke_proc) : jump with JUMP_TAG() if some
occures
* yarv.h : add yarv_set_current_running_thread_raw()
* yarvcore.c : add yarv_segv() and segv() method for
* yarvcore.c (Init_yarvcore) : set yarv_thread_t#running
* yarvcore.h : fix yarv_thread_t members
```

2006-01-03(Tue) 22:25:04 +0900 Koichi Sasada
<ko1@atdot.net>

```
* disasm.c (insn_operand_intern) : fix to add child i
* eval.c, gc.c : remove obsolete static variables (ru
ruby_dyna_vars, ruby_frame)
* eval.c (rb_mod_s_constants) : use ruby_cref()
* eval.c (eval) : use th_restore_class()
* eval_proc.c (rb_f_binding) : use th_store_class()
* insns.def (concatarray) : fix insn ([expr, *nil] =>
* vm.c (th_set_env), insnhelper.h : remove macro
* vm.c (eval_get_cvar_base) : use get_cref
* vm.c (th_make_proc) : use th_store_class()
* vm_macro.def (macro_eval_invoke_func) : fix option
```

```
* vm_macro.def (macro_eval_invoke_func) : raise stack
* yarvcore.h : add yarv_stored_klass_t type
* yarvcore.c : fix mark functions around yarv_stored_
```

2006-01-01(Sun) 05:14:26 +0900 Minero Aoki
<aamine@loveruby.net>

```
* lib/benchmark.rb: new file (imported from original
1.10).
```

2006-01-01(Sun) 03:51:10 +0900 Minero Aoki
<aamine@loveruby.net>

```
* yarvcore.c: add prototype.
* re.c: remove warning: long -> unsigned long.
* debug.c: adjust coding style.
* yarv.h: ditto.
```

2006-01-01(Sun) 03:43:33 +0900 Minero Aoki
<aamine@loveruby.net>

```
* variable.c: add prototype.
* eval.c: ditto.
* eval_load.c: ditto.
```

2006-01-01(Sun) 02:41:21 +0900 Koichi Sasada
<ko1@atdot.net>

```
* common.mk : add address analyse to vtune rule
* rb/vtlh.rb : added for above
```

```
* rb/insns2vm.rb, template/vm.inc.tpl : insert #line
to reference above

* vm_macro.def (macro_eval_invoke_cfunc) : fix indent

* yarvtest/test_method.rb : fix indent, spacing
and add a test for alias
```

2005-12-31(Sat) 12:42:05 +0900 Koichi Sasada
<ko1@atdot.net>

```
* common.mk : add Intel VTune rule (make vtune)

* eval.c, yarvcore.h : fix to remove yarv_thread_t#lo

* parse.y (top_local_init_gen) : fix a problem ([yarv

* yarvtest/test_eval.rb : add a test for above

* vm.c (thread_eval) : remove unused function

* yarvcore.c (Init_yarvcore) : remove YARVCore::Threa

* yarvcore.c (thread_eval) : remove unused function
```

2005-12-31(Sat) 06:05:00 +0900 Koichi Sasada
<ko1@atdot.net>

```
* vm.c (eval_search_super_class) : pass block to meth

* vm_macro.def (macro_eval_invoke_method) : ditto

* yarvtest/test_method.rb : add a test for above
```

2005-12-31(Sat) 03:11:14 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval.c (eval), eval_proc.c (rb_f_binding) : save kl
binding and use it at eval
```

```
* eval_intern.h : ditto
* yarvtest/test_eval.rb : add tests for above
* yarvcore.c (th_get_special_cref) : added
* yarvcore.h : add a prototype of above
* vm.c (th_get_cref) : refactoring
* vm.c (eval_get_ev_const) : fix SEGV at A::B (A is n
([yarv-dev:758])
* yarvtest/test_bin.rb : add a test for above
* rb/mklog.rb : use external diff command and show fu
```

2005-12-30(Fri) 19:07:51 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.c, yarvcore.h, eval.c, eval_proc.c : suppo
Ruby's Binding
* yarvcore.c : support TOPLEVEL_BINDING
* yarvtest/test_eval.rb : add tests for above
```

2005-12-30(Fri) 13:12:28 +0900 Minero Aoki
<aamine@loveruby.net>

```
* test/ruby/test_eval.rb: more tests for
module_eval/instance_eval.
```

2005-12-30(Fri) 05:06:49 +0900 Koichi Sasada
<ko1@atdot.net>

```
* common.mk : add dependency (yarvcore.h) for gc.c
* eval.c, eval_intern.h, eval_load.c, eval_method.h,
insns.def, insnhelper.h, vm.c, yarvcore.c, yarvcore.h
```

```
re-write class reference
* yarvtest/test_eval.rb : added
* yarvtest/test_proc.rb :
```

2005-12-29(Thu) 12:27:12 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c, yarvcore.h :
remvoe needless yarv_iseq_t#rewind_frame_size
```

2005-12-29(Thu) 11:17:58 +0900 Koichi Sasada
<ko1@atdot.net>

```
* common.mk : add dependency to test-all rule
* eval.c (rb_sourceline), vm.c (th_get_sourceline) :
fix to skip process if iseq is ifunc
* test/ruby/test_lambda.rb : assert(fail, ...) instead
* test/ruby/test_proc.rb : ditto
* vm_dump.c : fix stack dump (iseq name)
* vm_macro.def : store proc (block proc) to cfp#proc
* yarvcore.c : mark above on thread_mark
* eval.c (exec_under) : replace block#self ([yarv-dev
```

2005-12-29(Thu) 01:56:46 +0900 Koichi Sasada
<ko1@atdot.net>

```
* vm.c : fix setting of Proc cref ([yarv-dev:741])
* yarvcore.c : fix indent
```

2005-12-29(Thu) 00:17:03 +0900 Koichi Sasada

<ko1@atdot.net>

```
* disasm.c : show (block) local variable simple (not  
* gc.c : fix syntax error
```

2005-12-28(Wed) 23:35:06 +0900 Koichi Sasada
<ko1@atdot.net>

```
* class.c (method_entry) : fixed for undefed method (  
* compile.c : fix errinfo dvar id (#$!)  
and fix NODE_ERRINFO compilation  
* eval_proc.c, yarvcore.c : support YARVCore::VM::Pro  
* insns.def : remove useless TODO comments  
* insns.def : fix to use strict array conversion on  
checkarrayinclude  
* insns.def : fix defined?(yield) ([yarv-dev:744])  
* yarvcore.h : change yarv_iseq_t layout
```

2005-12-28(Wed) 16:49:55 +0900 Minero Aoki
<aamine@loveruby.net>

```
* test/ruby/test_eval.rb: add TODO comment.  
* test/ruby/test_iterator.rb: rename YARVCore::VM::Pr  
(tmp).  
* test/ruby/test_lambda.rb: use assert_fail.  
* test/ruby/test_proc.rb: ditto.
```

2005-12-28(Wed) 16:28:35 +0900 Minero Aoki
<aamine@loveruby.net>


```
* test/ruby/test_clone.rb: removed (tmp).  
* test/ruby/test_eval.rb: define missing method Object (tmp).  
* test/ruby/test_lambda.rb: turn off tests for "->".  
* test/ruby/test_proc.rb: turn off tests for |&b|.  
* test/ruby/test_proc.rb: turn off tests for $SAFE se
```

2005-12-28(Wed) 15:31:46 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : fix calculation of stack_max  
* eval.c (rb_iter) : fix block/retry handling  
* yarvtest/test_flow.rb : add tests for above  
* insns.def : fix block passing on super (super(&nil))  
* vm_macro.def, insns.def : fix convert method of obj  
* yarvtest/test_method.rb : fix a test for above  
* vm.c : fix backtrace generate algorithm
```

2005-12-28(Wed) 10:36:45 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c, compile.h : refactoring (remove self pas  
* disasm.c : support showing ID of method/dynamic loc  
* rb/allload.rb : add verbose version (it's enable by  
* template/insns.inc.tmpl, template/insns_info.inc.tm  
template/minsns.inc.tmpl, template/opt_sc.inc.tmpl,  
template/optinsn.inc.tmpl, template/optunifs.inc.tmpl  
template/vmtc.inc.tmpl : fix a comment
```

```
* variable.c (mod_av_set) : fix to clear inline cache
* eval_method.h : fix to clear inline method cache
* vm.c, rb/insns2vm.rb, template/insns_info.inc.tmpl,
insns.def, vm_evalbody.h, vm_macro.def :
fix operands types (ulong -> num_t, ...)
* vm_macro.def : fix to check SPECIAL_CONST_P() at sp
([yarv-dev:722])
* yarvcore.c : fix to throw syntax error
* yarvcore.h, eval.c, eval_error.h, eval_jump.h :
add yarv_vm_t#exit_code to fix problem at cleanup ([y
* insns.def : fix to invoke zsuper in method defined
([yarv-dev:704])
* yarvtest/test_class.rb : add tests for above
* yarvtest/test_method.rb : fix comments
```

2005-12-27(Tue) 01:52:07 +0900 Koichi Sasada
<ko1@atdot.net>

```
* array.c, intern.h, insns.def : expose rb_ary_replac
in insns.def
* eval.c : fix to use SCOPE_* to NOEX_*
* eval_intern.h : remove SCOPE_*
and fix SCOPE_TEST() and SCOPE_SET(f)
* eval_load.c : save and store klass and visibility
at require and load
* eval_method.h : fix undefed method node ([yarv-dev-
* eval_proc.c : fix define_method ([yarv-dev:704])
* insnhelper.h, vm.h : remove GET_VM_STATE_VERSION(),
INC_VM_STATE_VERSION() and move these to vm.h
```

```
* insns.def : supportintg visibility
* node.h : remove NOEX_RECV
* variable.c, vm.c : add rb_vm_change_state() and use
remove_const
* vm.c, insns.def, yarvcore.h, yarvcore.c : add eval_
eval_pop_cref() and th_cref_init to manage current vi
* yarv.h : add a prototype of rb_vm_change_state()
* yarvcore.h, insns.def : add defined_method_id and s
super in define_method scope
* yarvtest/test_class.rb : add tests for above
```

2005-12-26(Mon) 20:44:38 +0900 Minero Aoki
<aamine@loveruby.net>

```
* test/ruby/test_basicinstructions.rb: new file.
```

2005-12-26(Mon) 08:40:02 +0900 Koichi Sasada
<ko1@atdot.net>

```
* vm.c (eval_get_ev_const) : fix to skip nil
```

2005-12-26(Mon) 08:27:15 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insnhelper.h : fix GET_CVAR_EV_KLASS [yarv-dev:703]
```

2005-12-26(Mon) 07:51:01 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : add emptystack insn for all NODE_RETURN
and optimize it if it's not needed
```

```
* yarvtest/test_flow.rb : add a test for above
```

2005-12-26(Mon) 07:08:22 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval.c, gc.c : add "gc_debug_flag" to debug gc
* insns.def : add emptstack

* compile.c, rb/insns2vm.rb, template/insns_info.inc.
change interface of insn_stack_increase

* compile.c : fix return from ensure in method [yarv-
* yarvtest/test_flow.rb : add tests for above
```

2005-12-26(Mon) 02:15:02 +0900 Minero Aoki
<aamine@loveruby.net>

```
* test/ruby/test_alias.rb: do not use unimplemented d
```


2005-12-26(Mon) 02:00:11 +0900 Minero Aoki
<aamine@loveruby.net>

```
* test/runner.rb: new file.
* test/ruby/test_alias.rb: new file.
* test/ruby/test_clone.rb: new file.
* test/ruby/test_eval.rb: new file.
* test/ruby/test_iterator.rb: new file.
* test/ruby/test_lambda.rb: new file.
* test/ruby/test_proc.rb: new file.
* test/ruby/test_super.rb: new file.
* test/ruby/test_assignment.rb: new file.
```

```
* test/ruby/test_bignum.rb: new file.  
* test/ruby/test_call.rb: new file.  
* test/ruby/test_case.rb: new file.  
* test/ruby/test_condition.rb: new file.  
* test/ruby/test_const.rb: new file.  
* test/ruby/test_defined.rb: new file.  
* test/ruby/test_exception.rb: new file.  
* test/ruby/test_gc.rb: new file.  
* test/ruby/test_ifunless.rb: new file.  
* test/ruby/test_method.rb: new file.  
* test/ruby/test_trace.rb: new file.  
* test/ruby/test_variable.rb: new file.  
* test/ruby/test_whileuntil.rb: new file.
```

2005-12-25(Sun) 07:40:08 +0900 Koichi Sasada
<ko1@atdot.net>

```
* blockinlining.c, compile.c : fix block inlining  
* rb/insns2vm.rb : fix to support tracing stack depth  
with operands unification  
* vm_dump.c : fix to print Qundef on stack dump
```



2005-12-25(Sun) 01:45:55 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def, compile.c, rb/insns2vm.rb, template/insn  
trace stack depth at compile time  
and use it as cont_sp for exception handling
```

```
* yarvtest/test_exception.rb : add tests for above
* yarvtest/test_flow.rb : ditto
* Merry Xmas :)
```

2005-12-24(Sat) 19:34:04 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c, compile.h : fix ADD_CATCH_ENTRY and add
* eval_jump.h : fix catch to remove illegal error
```

2005-12-24(Sat) 09:05:23 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval_method.h : change data structure for RClass#m_
* class.c, eval.c, eval_proc.c : fix for above change
* node.h, gc.c : change NODE_FBODY, NODE_METHOD membe
for above changes
* insns.def : support private/protected visibility
* vm_macro.def : ditto
* vm.c : ditto
* thread.c : fix typo
* thread_pthread.h : fix typo
* thread_win32.h : fix typo
* eval.c, yarvcore.h : add yarv_thread_t#method_missi
to pass method_missing reason and use it to build err
* compile.c : use ADD_CALL instead of ADD_SEND for
NODE_X(D)STR, NODE_CONST (func)
```

2005-12-22(Thu) 02:45:27 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarv_version.h, Changes : 0.3.3
```

2005-12-20(Tue) 04:04:45 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : fix self::Const access  
* yarvtest/test_bin.rb : add a test for above
```

2005-12-20(Tue) 01:52:52 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def : fix to expand VALUES value  
* yarvtest/test_massign.rb : add a test for above
```

2005-12-20(Tue) 01:32:35 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def, insnhelper.h : fix cvar in singleton met  
* yarvtest/test_bin.rb : add tests for above
```

2005-12-20(Tue) 01:03:34 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c, yarvcore.h : support all defined?() synt  
* compile.c : fix NODE_COLON2  
* yarvtest/test_bin.rb : add or fix tests for above  
* win32/* : update all
```

2005-12-17(Sat) 10:46:08 +0900 Minero Aoki
<aamine@loveruby.net>

```
* vm_macro.def: fix printf type mismatch for LP64 sys
* parse.y: introduce descriptive macro for special va
  lvtbl->dvars.
```

2005-12-17(Sat) 09:39:27 +0900 Minero Aoki
<aamine@loveruby.net>

```
* vm_macro.def (macro_eval_invoke_method): fix printf
  for LP64 system.
```

2005-12-14(Wed) 03:49:40 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : change rescue/ensure iseq name
* eval.c, intern.h : fix a prototype
* insns.def, yarvcore.h : add trace_function
* vm.c : fix deadly bug (illegal pointer cast)
* vm_dump.c : remove unused local variables
* vm_macro.def : add parameter size check
* yarvtest/test_bin.rb : comment out 2 assertions
```

2005-12-13(Tue) 03:55:27 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval_proc.c : fix indent
* insns.def : fix getspecial instruction to return nil
  if no entry
```



```
* yarvtest/test_syntax.rb : add a test for above
* lib/un.rb : added
* template/*.tmpl : fix typo
```

2005-12-13(Mon) 01:38:17 +0900 Minero Aoki
<aamine@loveruby.net>

```
* yarv.h: add prototypes.
* intern.h: ditto.
* eval.c: ditto.
* debug.c: ditto.
* thread_pthread.h: fix printf type mismatch for LP64
(Linux/AMD64).
* variable.c: ditto.
* object.c: ditto.
* gc.c: ditto.
* process.c: ditto.
* error.c: ditto.
* vm.c: ditto.
* vm.h: ditto.
* vm_dump.c: ditto.
* disasm.c: ditto.
* marshal.c: ditto.
* eval_thread.c: ditto.
```

2005-12-11(Sun) 22:00:34 +0900 Koichi Sasada

<ko1@atdot.net>

```
* insns.def : call "inherited" method when a class is
* yarvcore.h : fix yarv_iseq_t field layout
* common.mk : add dependence on yarvcore.h to eval*.o
* compile.c : fix NODE_POSTEXE logic
* insnhelper.h : use GC_GUARDED_PTR_REF instead of ma
* eval_proc.c : fix indent
* configure : re-autoconf
```

2005-12-10(Sat) 03:57:20 +0900 Koichi Sasada
<ko1@atdot.net>

```
* common.mk : fix blockinlining.o build rule
* insns.def : remove logic for zsuper
* template/optinsn.inc.tmpl :
* vm.c : remove thread_yield_light_prepare, thread_yi
* compile.c : support NODE_ZSUPER with optargs, resta
* yarvtest/test_class.rb : add tests for above
```

2005-12-09(Fri) 01:13:37 +0900 Koichi Sasada
<ko1@atdot.net>

```
* array.c, numeric.c, range.c : add prototype of
block inlining function
* blockinlining.c, vm_opts.h.base : add block inlinin
* common.mk, debug.h, debug.c : add debug_breakpoint(
```

```
* compile.c : fix to use size_t on compile_data_alloc
fix illegal cast, fix to set arg_simple at compiling

* compile.c, vm.c : fix NODE_NEXT, NODE_BREAK logic

* yarvtest/test_flow.rb : add a test for above

* yarvcore.c, yarvcore.h, compile.c, eval.c : remove
yarv_iseq_t#root_iseq and add yarv_iseq_t#local_iseq
to use this member field

* eval_method.h : fix indent

* gc.c : fix indent

* insns.def, compile.c : remove "zsuper" instruction
instead). This is because NODE_ZSUPER represent with
instruction

* yarvcore.c : add proc_arity
```

2005-12-05(Mon) 03:58:30 +0900 Koichi Sasada
<ko1@atdot.net>

```
* array.c, blockinlining.c : support block inlining f

* disasm.c : fix catch table format

* insns.def : fix stack consistency error message

* vm.c : fix to skip pushing value at "next"

* yarvcore.h : move definition of
"struct iseq_compile_data_ensure_node_stack" to compi

* compile.c : fix ensure catch table creation

* yarvtest/test_flow.rb : add tests for above
```

2005-12-03(Sat) 22:27:08 +0900 Koichi Sasada
<ko1@atdot.net>

```
* blockinlining.c, compile.c, yarvcore.c, yarvcore.h,  
numeric.c, range.c : collect block inlining logic to
```

2005-12-03(Sat) 20:24:07 +0900 Koichi Sasada
<ko1@atdot.net>

```
* blockinlining.c, common.mk : add blockinlining.c  
  
* yarvcore.c, yarvcore.h, blockinlining.c, compile.c,  
gc.c, node.h, numeric.c, range.c :  
support block inlining for Integer#times, Range#each  
  
* compile.c : fix to set block redo/next point at las  
and fix NODE_OP_ASGN1  
  
* compile.c, vm.c : add specialized instruction "opt_  
  
* disasm.c : fix to show block, and to show catch typ  
and change node_name logic  
  
* eval_thread.c : fix function type declaration  
  
* insns.def : add instruction "putundef", "opt_checke  
to support block inlining and add stack check routine  
  
* lib/cgi.rb : add global variable $CGI_DONTINPUT  
  
* opt_operand.def : add some operand unification rule  
  
* rb/insns2vm.rb : fix operand unification logic for  
  
* vm.c : fix exception handling routine (collect stac  
  
* vm_macro.def : fix macro_eval_invoke_bmethod  
  
* yarvsubst.c : removed  
  
* yarvtest/test_syn.rb : rename to yarvtest/test_synt  
  
* yarvtest/yarvtest.rb : remove tempfile explicitly
```

2005-11-30(Wed) 01:13:57 +0900 Koichi Sasada

<ko1@atdot.net>

```
* common.mk : add vm_opts.h rule  
* vm.c, insns.def : fix proc creation under class and  
environment
```

2005-11-29(Tue) 16:39:07 +0900 Koichi Sasada

<ko1@atdot.net>

```
* eval.c, eval_proc.c, vm.c, vm_macro.def :  
support define_method and invoke NODE_BMETHOD method
```

2005-11-29(Tue) 13:18:06 +0900 Koichi Sasada

<ko1@atdot.net>

```
* compile.c : add iseq_add_mark_object, iseq_add_mark  
and use it to mark objects on iseq  
* compile.h, compile.c : remove cast on NEW_CHILD_ISEQ  
and interface  
* compile.c, disasm.c, insns.def, vm_macro.def, rb/in  
add BLOCKISEQ parameter type  
* gc.c : fix garbage_collect to return true if only a  
* vm.c : fix insertion order of proc/env  
* vm_evalbody.h : add typedef yarv_iseq_t *BLOCKISEQ  
* yarvcore.c, yarvcore.c : add idTimes  
* yarvcore.c : fix proc_mark, env_mark around iseq ma
```

2005-11-28(Mon) 09:02:57 +0900 Koichi Sasada

<ko1@atdot.net>

```
* compile.c, insns.def, vm_evalbody.h : support super  
with splat argument and block (and zsuper with block)
```

```
* yarvtest/test_class.rb : add tests for above
* compile.c, yarvcore.h, yarvcore.c, insns.def, time.
add opt_succ insn
* eval_method.h : fix indent
* eval_thread.c : apply cast to vanish a warning
* lib/tempfile.rb, lib/tmpdir.rb : added
* vm.c : eval_method_missing added
* vm_macro.def : refactoring
```

2005-11-21(Mon) 21:21:33 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c, compile.h, yarvcore.c : remove "iseqobj"
variables and rename to "iseq"
```

2005-11-21(Mon) 07:31:50 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : fix block parameter error
* ext/* : added
* lib/optparse* : added
* benchmark/bm_so_sieve.rb : fix parameter
```

2005-11-21(Mon) 03:47:28 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : optimize condition in literal
* thread_win32.h : fix win32 thread function prototyp
```

2005-11-20(Sun) 17:58:24 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : fix NODE_AND/OR bug
* eval.c : support rb_frame_this_func()
```

2005-11-20(Sun) 12:32:31 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c, yarvcore.c, yarvcore.h : support NODE_OP
* compile.h : add macro ADD_CALL
* debug.c : add debug_v() and change to use only print
on debug_id()
* sample/test.rb :
* vm.c : fix make_proc_from_block
```

2005-11-19(Sat) 14:55:17 +0900 Koichi Sasada
<ko1@atdot.net>

```
* import ruby 1.9.0 (2005-11-18)
```

2005-11-19(Sat) 06:08:37 +0900 Koichi Sasada
<ko1@atdot.net>

```
* lib/test : added
```

2005-11-19(Sat) 05:48:50 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : useless jump elimination (if/unless des
* eval.c : rb_iter_break support,
fix rb_iterate (clear errinfo if break)
```

```
* eval_proc.c : support rb_node_arity (YARV_METHOD_NO
* insns.def : change variable name
* vm.c : fix th_invoke_yield and add th_iter_break()
* vm_dump.c : fix yarv_bug()
* yarvcore.c : fix proc_mark to check IFUNC node and
global ruby method SDR() for debug
* yarvtest/test_syn.rb : add a test for all condition
```

2005-11-15(Tue) 05:52:58 +0900 Koichi Sasada
<ko1@atdot.net>

```
* lib/forwardable.rb : added
* common.mk : remove "vm.o : CFLAGS += -fno-crossjump
* compile.c, yarvcore.h, insns.def : add FCALL/VCALL
* compile.c, insns.def : add onceinlinecache instruct
* eval.c : support $!, $@, raise (== raise $!)
* opt_operand.def : add some unification rule (send f
* vm.c : fix return process
* vm_macro.def : fix option prameters
* yarvtest/test_method.rb : add tests for above
```

2005-11-15(Tue) 00:42:49 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval.c : support rb_frame_pop() and rb_frame_callee
add rb_sourcefile(), rb_sourceline(),
* compile.c : support postposition while/until,
fix block parameter index
```



```
* yarvtest/test_syn.rb : add tests for above
* yarvcore.c : fix env_mark
* vm.h, yarvcore.h : move vm.h#cmethod_info to
  yarvcore.h#yarv_cmethod_info
* vm.c : add th_get_sourceline()
* eval_intern.h : fix PASS_PASSED_BLOCK()
* eval_load.c : fix re-enter require (temporality)
* insns.def : permit re-open class when superclass is
```

2005-11-11(Fri) 01:20:15 +0900 Koichi Sasada
<ko1@atdot.net>

```
* common.mk : add "allload" rule
* compile.c, yarvcore.h, insns.def, vm_macro.def, dis
  change arg_rest, arg_block offset (1)
* insns.def : add postexe instruction
* insns.def, vm.c : support rest block parameter
* yarvtest/test_block.rb : add tests for above
* rb/allload.rb : get path from ARGV
* vm_opts.h.base : set default off
```

2005-11-01(Tue) 08:28:19 +0900 Koichi Sasada
<ko1@atdot.net>

```
* benchmark/other-lang/eval.rb : fix path
* lib/English.rb, lib/cgi.rb, lib/complex.rb, lib/del
  added
```

2005-11-01(Tue) 08:18:33 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : push and pop values after checkincludea  
stack caching
```

2005-10-31(Mon) 15:37:09 +0900 Koichi Sasada
<ko1@atdot.net>

```
* benchmark/bm_app_mandelbrot.rb : added  
* benchmark/bm_app_factorial.rb : fixed parameter  
* benchmark/bm_so_count_words.rb, benchmark/run_rite.  
real file  
* common.mk : add "ext" rule, add some dependencies a  
to bench-each rule (renamed from bench-item)  
* compile.c : fix get_root_iseq_object (check iseq ty  
support splat case/when. support //o (regexp)  
* eval.c : support *_eval, fix rb_obj_call_init to pa  
* eval_jump.h : support throw/catch  
* eval_load.c : save klass_nest_stack when require  
* eval_method.h : fix ruby_cbase()  
* insnhelper.h : GET_EV_KLASS checks toplevel or not  
* insns.def, yarvcore.c : fix singleton method defini  
super class's method  
* lib/shellwords.rb : use String() instead of String.  
* vm.c : check class iseq or not when making Proc and  
add eval_search_super_klass function  
* vm.h : CMETHOD_INFO_P to yarvcore.h  
* vm_macro.def : splat if object type is T_ARRAY
```

```
* vm_opts.h, vm_opts.h.base : rename to vm_opts.h.base
insns2vm.rb will copy it to build directory

* yarvcore.c : add Proc#[ ]

* yarvcore.h : change INITIAL_ISEQ_COMPILE_DATA_STORAGE
to 512

* yarvtest/test_* : invalidate splat non array code (

* yarvtest/yarvtest.rb : use tempfile instead of pope
```

2005-10-28(Fri) 09:11:53 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvtest/test_method.rb : fix test
```

2005-10-28(Fri) 08:43:29 +0900 Koichi Sasada
<ko1@atdot.net>

```
* benchmark/run_rite.rb : add -I options to run bench

* common.mk : pass options to some rules with RUNOPT
and add -I options

* compile.c : fix massign with constant

* yarvtest/test_massign.rb : add tests for above

* eval_load.c : fix load_wait()

* eval_method.h : support ruby_cbase()

* lib/*.rb : add or modify libraries to run on yarv
* parse.y : change to ANSI C style

* vm.c : fix making proc process under cfunc/ifunc en

* vm_macro.def : fix block pass

* yarvtest/test_method.rb : add tests for above
```

```
* yarvcore.c : add yarv_obj_is_proc()  
* eval.c : fix rb_obj_is_proc to use yarv_obj_is_proc
```

2005-10-27(Thu) 11:50:15 +0900 Koichi Sasada
<ko1@atdot.net>

```
* some files : import from ruby 1.9.0 (2005-10-12)
```

2005-10-16(Sun) 14:50:02 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def, compile.c, yarvcore.h, yarvcore.c : add
```

2005-10-11(Tue) 17:01:13 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarv_version.h, Changes : 0.3.2
```

2005-10-11(Tue) 13:35:25 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def : add YARV_CHECK_INTS()  
* thread.c, thread_pthread.h, thread_win32.h : kick t  
when another thread kicked  
* vm.c : remove debug print  
* vm_opts.h : add OPT_CALL_THREADED_CODE  
* yarvtest/yarvtest.rb : remove "\r" from answer
```

2005-10-07(Fri) 09:36:36 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.h : add member variable "interrupt_flag" t
```

2005-10-05(Wed) 21:20:13 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eva.c, eval_thread.c, ruby.h, eval_error.h, eval_ju  
eval_load.c, thread.c, error.c, compile.h : remove ru  
* thread_win32.h, thread_pthread.h : set stack size t  
* vm.c : fix making env routine  
* vm_dump.c, vm.h : support frame type "EVAL" and fix  
* yarvcore.c : fix some mark/free routine
```

2005-10-05(Wed) 09:08:11 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval.c, eval_intern.h, vm.c, eval_jump.h, yarvcore.  
re-define PUSH/POP/EXEC/JUMP_TAG to use thread local  
* inits.c, yarvcore.c : fix bootstrap
```

2005-10-03(Mon) 22:28:24 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : fix NODE_COLON2 bugs  
* compile.h : fix debug routine  
* disasm.c : add space between insn and operand  
* insns.def : add comment of clasdef, singletonclass  
* vm.c, yarv.h : fix invoke_light routine  
* yarvcore.c : fix to mark each threads
```

2005-10-02(Sun) 05:55:34 +0900 Koichi Sasada
<ko1@atdot.net>

```
* thread_pthread.h : add "system_working" global vari
```

2005-10-02(Sun) 01:23:44 +0900 Koichi Sasada
<ko1@atdot.net>

```
* thread.c : add raw gets (for test), and fix indent
```

2005-10-01(Sat) 23:06:21 +0900 Koichi Sasada
<ko1@atdot.net>

```
* thread_win32.h, common.mk : add thread_win32.h  
* thread.c : support _WIN32 thread  
* thread.c, thread_pthread.h : fix some interface  
* eval_thread.c : remove debug print  
* gc.c : fix stack region  
* win32/Makefile.sub : add -MD flag to LDFLAGS  
* yarvcore.c : fix mark and sweep debug print  
* yarvcore.h : fix VM#living_threads data type to st
```

2005-10-01(Sat) 00:25:28 +0900 Koichi Sasada
<ko1@atdot.net>

```
* thread.c, yarvcore.h : rename GIL (Global Interpret  
GVL (Global VM Lock)  
* thread_pthread.h : fix pthread mutex initialize
```

2005-09-30(Fri) 20:11:19 +0900 Koichi Sasada

<ko1@atdot.net>

```
* thread.c : support join with timeout
* yarvcore.h : use GET_VM()
```

2005-09-30(Fri) 14:59:29 +0900 Koichi Sasada
<ko1@atdot.net>

```
* thread.c, common.mk : add thread.c
* thread.c, gc.c, eval_thread.c, yarvcore.c, yarvcore
support native thread (on pthread)
* insns.def : add YARV_CHECK_INTS() check
* yarv.h : add GET_VM() macro
```

2005-09-29(Thu) 22:43:08 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval_intern.h, eval_thread.c : move thread_status t
* yarvcore.c : fix thread/vm value
* yarvcore.h : add some parameter to yarv_thread_t
```

2005-09-29(Thu) 01:52:33 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c, yarvcore.h : add line number on last end
* vm.c : fix line no detection
```

2005-09-28(Wed) 00:02:10 +0900 Koichi Sasada
<ko1@atdot.net>

```
* common.mk, eval_load.c, eval.c, eval_intern.h : add
* disasm.c : fix around block local variables
* eval_proc.c : fix typo
```

2005-09-27(Tue) 16:45:20 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval.c : remove debug print
```

2005-09-27(Tue) 16:41:47 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval.c : support Kernel.local_variables
* parse.y, yarvcore.c : move some functions
(rb_(backref|lastline)_(get|set)) from parse.y to yar
* yarvcore.h : fix typo of YARV_PREVIOUS_CONTROL_FRAME
```

2005-09-26(Mon) 18:51:29 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval.c, compile.c, parse.y, vm.c, yarvcore.h :
eval() works with binding (Env)
* vm.c : add th_set_eval_stack
* yarvtest/test_syn.rb : remove an assert "defined?(1
```

2005-09-25(Sun) 19:30:59 +0900 Koichi Sasada
<ko1@atdot.net>

```
* benchmark/bm_vm2_send.rb : added
* common.mk : add rule "bench-item"
```



```
* eval_intern.h : add PASS_PASSED_BLOCK()
* eval_proc.c : support some functions
* rb/mklog.rb : added
* vm.c : fix prototype style and coding style
* yarv.h : add some prototypes of functions
* yarvcore.c, yarvcore.h, eval.c : yarv_thread_t#ifun
and add yarv_proc_t#safe_level
```

2005-09-25(Sun) 11:01:17 +0900 Koichi Sasada
<ko1@atdot.net>

```
* some files : import from ruby 1.9.0 (2005-09-25)
* eval*, vm.c, vm_macro.def : remove frame, scope, ..
* yarvcore.c : remove yarv_block_given_p()
* yarvcore.h, insnhelper.h : move some macro from ins
to use these in eval.c
```

2005-09-24(Sat) 15:51:42 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval* : remove dependency to ruby_dyna_vars and rub
```

2005-09-23(Fri) 20:39:14 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval_*. [ch] : split eval.c to some files
* *. [ch] : import ruby 1.9.0 (2004-09-23)
* parse.y : remove dependency to ruby_dyna_vars and r
```

2005-09-15(Thu) 16:51:06 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c, yarvcore.h : fix "for" scope
* yarvtest/test_block.rb : add tests for above
```

2005-09-14(Wed) 06:11:43 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.h, vm_evalbody.h, vm.h, vm_dump.c,
compile.c, yarvcore.c : use #ifdef insted of #if for
vm options
* vm_opts.h : fix default options
```

2005-09-10(Sat) 14:10:08 +0900 Koichi Sasada
<ko1@atdot.net>

```
* vm_opts.h : added
* yarvcore.h, rb/insns2vm.h : use vm_opts.h
```

2005-09-10(Sat) 04:53:22 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.h, insns.def, compile.c : add DEFINED_YIELD
* yarvtest/test_yield.rb : add test_1_ary_and_1_param
* insns.def : fix splat and svalue
* vm.c : fix to perform with proc with ifunc (incomplete)
* sample/test.rb : added (comment out unsupported feature)
* common.mk : add rule "runtest"
```

2005-09-09(Fri) 19:32:11 +0900 Koichi Sasada

<ko1@atdot.net>

```
* insns.def, compile.c : add splatarray
* yarvtest/test_massign.rb : add tests for above
```

2005-08-31(Wed) 22:55:15 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.c (yarvcore_eval_parsed): fix to return va
* yarv_version.h, Changes : 0.3.1
```

2005-08-20(Sat) 10:19:27 +0900 Koichi Sasada
<ko1@atdot.net>

```
* rb/ir.rb : add some check
* import today's ruby HEAD
```

2005-08-18(Thu) 23:29:52 +0900 Koichi Sasada
<ko1@atdot.net>

```
* common.mk : fix object file extension
* rb/ir.rb : added (import ruby script)
* rb/diff.rb : removed
* import today's ruby HEAD
```

2005-08-18(Thu) 12:59:38 +0900 Koichi Sasada
<ko1@atdot.net>

```
* common.mk : rule test -> test2, test1 -> test
* compile.c : fix when clause bug and splat arugment
```

2005-08-17(Wed) 05:22:31 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : fix block local parameter setting routi  
massign in block parameter initialize  
  
* yarvtest/test_yield.rb : add tests for above  
  
* insns.def, compile.c : support array concat (ex: "[  
  
* yarvtest/test_bin.rb : add tests for above
```

2005-08-16(Tue) 19:51:19 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : support nested massign  
  
* yarvtest/test_massign.rb : add tests for above
```

2005-08-16(Tue) 10:25:29 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval.c : support rb_yield_0 with 0 args
```

2005-08-16(Tue) 09:09:21 +0900 Koichi Sasada
<ko1@atdot.net>

```
* lib/fileutils.rb : imported  
  
* insns.def : fix yield argument (same as last commit  
  
* yarvtest/test_yield.rb : add tests for above
```

2005-08-16(Tue) 08:29:47 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval.c : fix to support rb_yield_0 with multiple va
```

```
* common.mk : add parse, run1p ruelse
* compile.c : support yield with ARGSCAT/SPLAT
* vm.c, insns.def : fix yield arguments to do compati
* yarvtest/test_yield.rb : added for above
```

2005-08-16(Tue) 06:00:17 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def : fix to set klass_nest_stack on singleto
method definition
* yarvtest/test_method.rb : add a test for above
```

2005-08-16(Tue) 05:34:48 +0900 Koichi Sasada
<ko1@atdot.net>


```
* test1.rb : added. gdb and run1 rule run this scrip
* compile.c : fix error handled variable access
* yarvtest/test_exception.rb : add tests for above
```

2005-08-16(Tue) 04:26:08 +0900 Koichi Sasada
<ko1@atdot.net>

```
* base ruby : ruby 1.9.0 (2005-08-15)
```

2005-08-16(Tue) 03:54:17 +0900 Koichi Sasada
<ko1@atdot.net>

```
* common.mk, Makefile.in : move some rules to common.
* rb/diff.rb : added
* yarvtest/yarvtest.rb : fix to compare output last v
```



2005-08-15(Mon) 18:27:58 +0900 Koichi Sasada
<ko1@atdot.net>

```
* Changes : 0.3.0
```

2005-08-15(Mon) 17:56:09 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.c : fix to add prototype  
* all files : propset svn:eol-style native
```

2005-08-15(Mon) 10:48:53 +0900 Koichi Sasada
<ko1@atdot.net>

```
* eval.c : support rb_load
```

2005-08-15(Mon) 09:42:01 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.h : define SDR()  
* vm_dump.c : stack_dump_raw() -> vm_stack_dump_raw()  
* yarvtest/yarvtest.rb : add rite test scheme  
* benchmark/run_rite.rb : added  
* yarvcore.c, inits.c : add Init_vm()  
* yarv.h : add some prototype declarations, GET_THREA  
* eval.c : remove unused functions  
* eval.c : support Kernel.eval, some schemes (same as
```



2005-08-15(Mon) 00:53:28 +0900 Koichi Sasada

<ko1@atdot.net>

```
* yarv_version.h : move configurations to yarvcore.h
* yarvcore.c : remove VALUE yarv_get_current_running_
add yarv_thread_t *yarv_get_current_running_thread(),
* yarvcore.h : yarv_thread_t#vm -> vm_value
* compile.c : fix "break from nested classes"
* yarvext/extconf.rb : use have_func instead of defin
* depend : fix pass
* eval.c : change to kick VM
* version.c : fix to show yarv version
* common.mk : fix dependent
* inits.c : fix to kick Init_yarvcore
```

2005-08-14(Sun) 02:05:15 +0900 Koichi Sasada
<ko1@atdot.net>

```
* README : add description
* yarvext/depend : move to topdir/depend
```

2005-08-14(Sun) 01:50:43 +0900 Koichi Sasada
<ko1@atdot.net>

```
* merge yarv to ruby (prepare)
* make yarvext/ to build as extension
```

2005-08-13(Sat) 09:36:26 +0900 Koichi Sasada
<ko1@atdot.net>

```
* evalc.patch, insns.def, compile.c : fix to support
```

```
ruby HEAD.
```

```
* 0.2.3
```

2005-08-08(Mon) 19:13:02 +0900 Koichi Sasada
<ko1@atdot.net>

```
* version.h, Changes : 0.2.2
```

2005-08-08(Mon) 17:17:50 +0900 Koichi Sasada
<ko1@atdot.net>

```
* vm.h, vm.c, insns.def, yarvcore.h, yarvcore.c :  
remove yarv_iseq_t#iseq_dt and add yarv_iseq_t#encode  
use yarv_iseq_t#encoded anytime
```

```
* vm_evalbody.h, vm.h, extconf.rb, version.h :  
support call threaded code (incomplete)
```

2005-08-01(Mon) 05:26:12 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.c : support yield with multiple values
```

```
* compile.c : fix dynavars
```

```
* yarvcore.h : fix to mark defined method
```

2005-07-31(Sun) 23:27:24 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.c, vm.c, insns.def : fix search object pat
```

```
* compile.c : fix "for" statement
```

```
* vm_macro.def : fix rest, opt arguments
```


2005-07-31(Sun) 14:52:06 +0900 Koichi Sasada
<ko1@atdot.net>

```
* vm_macro.def : fix block parameter
* compile.c : fix to unuse compile_data->in_ensure
* insns.def : add orphan check when return
```

2005-07-31(Sun) 03:25:05 +0900 Koichi Sasada
<ko1@atdot.net>

```
* vm.c, compile.c, yarvcore.h, insns.def :
support jump from rescue/ensure/class/module
* test/test_flow.rb : add tests for above fix
```

2005-07-30(Sat) 04:44:33 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.h : struct iseq_compile_data_ensure_node_s
* compile.c : insert ensure clause before break/next/
* vm.c : fix return/break handling
* yarv.h, vm.c : fix lightweight yield
* vm.c, insns.def, vm_macro.def : change arguments of
* test/test_flow.rb : added
* test/yarvtest.rb : add ae_flow
* compile.c, vm_macro.def : add tail-call/tail-recurs
(experimental)
```

2005-07-29(Fri) 20:14:11 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : make_name_for_block and make_name_with_
are added

* insns.def : fix if unmatched size arg size to yield

* test/test_block.rb : add test for above fix

* vm.c : add th_backtrace_each and fix backtrace nota

* yarvcore.c : set top level iseq name to "<main>"
```

2005-07-29(Fri) 13:20:19 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.h : fix yarv_iseq_t to pass VC (cl)

* vm_dump.c : ditto

* compile.h : ditto

* insnhelper.h : ditto

* vm_evalbody.h : include 'math.h'

* insns.def, vm.c : raise error when yield without bl

* vm.c : implement thread_backtrace

* vm.c, yarvsubst.c, yarv.h : implement thread_yield_
thread_yield_light_invoke

* yarvcore.c : Integer#times uses yarv specific versi
```

2005-07-28(Thu) 21:35:09 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.c : add another mark function for thread/s

* vm_evalbody.h : fix register allocation for x86_64

* vm.h : use asm for tc on x86_64
```

2005-07-28(Thu) 20:17:09 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.c : add mark/free message to debug gc
* insnhelper.h, insns.def, vm_macro.def : remove and
add new RESTORE_REGS
* vm_evalbody.h : fix register allocation
```

2005-07-28(Thu) 02:00:42 +0900 Koichi Sasada
<ko1@atdot.net>

```
* vm.c, etc : change VM stack structure. re-write all
vm functions to do it
* vm_macro.def : added
```

2005-07-08(Fri) 01:36:49 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def : don't use fmod on AMD64
```

2005-07-08(Fri) 00:14:22 +0900 Koichi Sasada
<ko1@atdot.net>

```
* Changes : added
```

2005-07-07(Thu) 23:54:37 +0900 Koichi Sasada
<ko1@atdot.net>

```
* version.h : 0.3.0
```

2005-07-07(Thu) 23:52:03 +0900 Koichi Sasada
<ko1@atdot.net>

```
* 0.2.1 : released
```

2005-07-07(Thu) 23:50:22 +0900 Koichi Sasada
<ko1@atdot.net>

```
* version.h : 0.2.1
```

2005-07-07(Thu) 23:47:55 +0900 Koichi Sasada
<ko1@atdot.net>

```
* rb/insns2vm.rb, extconf.rb : add --[enable|disable]  
and --disable-opts  
  
* vm.h : DISPATCH_ARCH_DEPEND_WAY is only enabled on
```

2005-07-06(Wed) 13:20:27 +0900 Koichi Sasada
<ko1@atdot.net>

```
* depend, rb/eval.rb : add ITEMS option to benchmark  
  
* benchmark/* : changed  
  
* benchmark/other-lang/* : added
```

2005-07-04(Mon) 04:02:15 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.h, yarvcore.c : add idDIV, idMOD, idEq, id  
  
* compile.c, insns.def : add specialized insn for abo  
  
* test/test_bin.rb : add tests for above
```

2005-07-03(Sun) 20:31:09 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.c, yarvcore.h : remove cYarvThrowObject (u
* yarvcore.c, yarvcore.h, insns.def :
thread_object#stack_mark_poinetr
* depend, rb/eval.rb : BOPT, TOPT -> OPT
```

2005-07-03(Sun) 13:53:47 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c, compile.h : INSN_OBJECT, LABEL_OBJECT ->
ISEQ_LINK_ELEMENT, ISEQ_LINK_ANCHOR -> LINK_ELEMENT,
and some fixes
* tmpl/optinsn.inc.tmpl : ditto
* yarvcore.c, yarvcore.h : remove label_object, insn_
prepare_iseq_build, cleanup_iseq_build are added
* insns.def : remove unused variable from send
```

2005-07-02(Sat) 04:19:22 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def : add GC protect for opt_aset
```

2005-07-02(Sat) 03:49:17 +0900 Koichi Sasada
<ko1@atdot.net>

```
* extconf.rb : add option -fno-reorder-blocks to vm.a
* insns.def : fix opt_aset bugs
* test/test_bin.rb : add tests for aset, aref
```

2005-07-02(Sat) 03:05:12 +0900 Koichi Sasada
<ko1@atdot.net>

```
* benchmark/run.rb : fix output
* vm_evalbody.h : add register for x86_64
* rb/asm_parse.rb : fix to shor size and length
```

2005-07-02(Sat) 02:56:31 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : move specialized instruction point (new
* insns.def : add opt_aref, opt_aset
```

2005-07-01(Fri) 11:04:11 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.h : fix to pass VALUE type to new_insn_body
* insnhelper.h : add cast
* compile.c : fix getdynamic argument (0 == Qfalse ->
```

2005-06-30(Thu) 23:34:10 +0900 Koichi Sasada
<ko1@atdot.net>

```
* rb/eval.rb : add and fix some rules
* rb/insns2vm.rb : generate all
* benchmark/run.rb : add -r (ruby only) option
```

2005-06-30(Thu) 23:25:23 +0900 Koichi Sasada
<ko1@atdot.net>

```
* tmpl/vmtc.inc.tpl : add const prefix
* /rb/asm_parse.rb, extconf.rb : added and make assem
```

```
* opt_operand.def : add send operands unification
* insnhelper.h : add HEAP_CLASS_OF(obj)
* insns.def : fix opt_plus, opt_ltlr
* vm_evalbody.h : move _tag
* benchmark/run.rb : fix file select
```

2005-06-30(Thu) 06:07:04 +0900 Koichi Sasada
<ko1@atdot.net>

```
* extconf.rb : add collect-usage-analysis option
* opt_operand.def, opt_insn_unif.def : add some rules
```

2005-06-29(Wed) 23:28:44 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarcvcore.h, extconf.rb, vm.h, compile.c :
DISPATCH_DIRECT_THREADED_CODE, DISPATCH_THREADED_CODE
-> OPT_DIRECT_THREADED_CODE, OPT_INDIRECT_THREADED_
if at least one of then is defined, OPT_THREADED_CODE
* benchmark/* : fix name and parameters
* rb/eval.rb : added for YARV evaluation
```

2005-06-29(Wed) 16:16:52 +0900 Koichi Sasada
<ko1@atdot.net>

```
* benchmark/run.rb : fix output format
* call_cfunc.inc -> call_cfunc.h
* vm.h : add sign by asm statement
```

2005-06-28(Tue) 22:28:40 +0900 Koichi Sasada

<ko1@atdot.net>

```
* vm.c : fix method search
```

2005-06-28(Tue) 22:26:34 +0900 Koichi Sasada
<ko1@atdot.net>

```
* extconf.rb : fix options
```

2005-06-28(Tue) 21:50:58 +0900 Koichi Sasada
<ko1@atdot.net>

```
* benchmark/run.rb : fix output format
```

2005-06-28(Tue) 21:34:54 +0900 Koichi Sasada
<ko1@atdot.net>

```
* depend : add option TOPT to test rules  
* benchmark/run.rb : fix output format
```

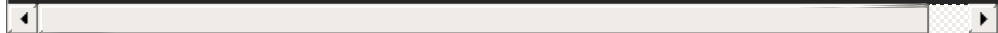
2005-06-28(Tue) 21:15:54 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : fix opt_case_dispatch instruction  
* benchmark/run.rb : output all usertimes when exit b
```



2005-06-28(Tue) 20:35:55 +0900 Koichi Sasada
<ko1@atdot.net>

```
* extconf.rb, compile.c, tmp1/optinsn.inc.tpl, vm.c  
change extconf options
```



2005-06-28(Tue) 13:20:59 +0900 Koichi Sasada

<ko1@atdot.net>

```
* benchmark/run.rb : add -y, --yarv-only option
* depend : add BOPT to tbench rule
```

2005-06-27(Mon) 23:31:12 +0900 Koichi Sasada
<ko1@atdot.net>

```
* depend : add gdb rule
* vm.h : use inline assembler for x86 (to support gcc
```

2005-06-27(Mon) 20:04:10 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.c, compile.c, disasm.c : remove unused var
* vm.h, insnhelper.h, debug.h : fix to reduce warning
* vm.c, vm_dump.c : move VM state dump (debug) functi
* depend : adde reconf rule
* insnhelper.h :
* vm_evalbody.inc : rename to vm_evalbody.h
```

2005-06-27(Mon) 16:50:31 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns2vm.rb : fix generating unif insn
* compile.c : add useless pop/swap insn elimination w
* depend : remove compiled.o dependency
```

2005-06-26(Sun) 14:06:22 +0900 Koichi Sasada

<ko1@atdot.net>

```
* benchmark/run.rb : use tmpfile instead of popen
* rb/insns2vm.rb : fix generating insn unification lo
* opt_insn_unif.def : add some unification rules
* compile.c : add verify_list function and fix unific
```

2005-06-22(Wed) 12:58:26 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.h, yarvcore.c, insns.def, compile.c : add
* test/test_bin.rb : add test_fact
```

2005-06-21(Tue) 22:34:07 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.h, compile.[ch], tmp/optinsn.inc.tmp1, rb
change data structure (don't use Ruby's array to repr
instruction sequence)
* disasm.c : add separator
```

2005-06-14(Tue) 07:48:58 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : support "for" statement
* test/test_block.rb : add test for above
* yarvcore.[ch] : add global id idEach
```

2005-06-08(Wed) 22:30:44 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : add if/unless(L1) jump (L2) :L1 => unle  
optimize (condition reversal) and fix typo
```

2005-06-07(Tue) 08:29:41 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.c : fix to remove compiler warning  
* version.h : 0.2.1
```

2005-06-07(Tue) 08:16:22 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.h : iseq_link_element changed to double li  
* disasm.c : support dump struct iseq_link_element  
* compile.c : use double linked list instead of array  
for intermediate representation
```

2005-06-06(Mon) 15:38:44 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.h, yarvcore.c : add link structure to insn  
* compile.h, compile.c : remove some variables in fun  
of iseq_compile_each and some optimization (now worki
```

2005-06-04(Sat) 16:12:59 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : fix previous commit
```

2005-06-04(Sat) 15:56:21 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : fix stack caching (after jump state)
```

2005-06-04(Sat) 09:12:13 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : fix some point for previous commit
```

2005-06-04(Sat) 07:31:21 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c, insns.def : optimize case/when statement  
(dispatch on constant time)  
  
* yarvcore.h, disasm.c, rb/insns2vm.rb : fixed for abo  
(CDHASH)  
  
* test/test_syn.rb : add test for above
```

2005-06-04(Sat) 03:41:29 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.h, yarvcore.c : add some temporary variabl  
(it'll be vanished)  
  
* compile.c : NODE_CASE optimize (use topn instead of
```

2005-06-03(Fri) 00:54:38 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : apply flow optimization for while/until
```

2005-03-04(Fri) 19:34:32 +0900 Koichi Sasada
<ko1@atdot.net>

```
* rb/insns2vm.rb : fix category (comment)
```

```
* depend : remove space between target name and colo
```

2005-03-04(Fri) 15:55:51 +0900 Koichi Sasada
<ko1@atdot.net>

```
* tmpl/yarvarch.ja : fix typo
```

2005-03-04(Fri) 13:30:19 +0900 Koichi Sasada
<ko1@atdot.net>

```
* depend : add a rule for jitcompile.o  
* vm.h : fix a macro argument  
* version.h : 0.2.0
```

2005-03-03(Thu) 08:35:14 +0900 Koichi Sasada
<ko1@atdot.net>

```
* extconf.rb : remove vm_evalbody.inc call_cfunc.inc
```

2005-03-03(Thu) 00:54:15 +0900 Koichi Sasada
<ko1@atdot.net>

```
* tmpl/insns.inc.tmpl : fixed typo  
* insns.def : store th->pc to current pc
```

2005-03-03(Thu) 00:31:47 +0900 Koichi Sasada
<ko1@atdot.net>

```
* tmpl/yarvarch.ja, doc/yarv.rb : write current archi
```

2005-03-01(Tue) 13:50:04 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.c (yarvcore_eval_parsed) : added
(separated from yarvcore_eval)

* yarvcore.c, compile.c : iseq_translate_direct_threa
is moved to compile.c

* depend : add rule for yasmdata.rb

* rb/yasm.rb : support top-level and method-level ass
```

2005-02-26(Sat) 08:09:57 +0900 Koichi Sasada
<ko1@atdot.net>

```
* rb/insns2vm.rb, compile.c, vm.h : change type long

* tmp1/yasmdata.rb.tmp1 : added

* rb/insns2vm.rb : add yasmdata_rb method

* rb/yasm.rb : fix some interface (incomplete)

* compile.c : iseq_setup added

* yarvcore.c : YARVCore::InstructionSequence::Instruc
```

2005-02-24(Thu) 07:45:37 +0900 Koichi Sasada
<ko1@atdot.net>

```
* rb/yasm.rb : added
```

2005-02-24(Thu) 01:13:33 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : remove useless statements
```

2005-02-24(Thu) 00:46:44 +0900 Koichi Sasada
<ko1@atdot.net>

```
* rb/insns2vm.rb (InsnInfo) : add @is_sc attr and remove  
is_sc method  
  
* compile.c : fix NODE_CASE/NODE_WHEN bug (cond at 'w'  
must not be popped)  
  
* compile.c : support NODE_OP_ASGN1 to &&= and ||=  
  
* test/test_bin.rb : add tests for above
```

2005-02-23(Wed) 09:17:01 +0900 Koichi Sasada
<ko1@atdot.net>

```
* vm.c, yarvcore.c : thread_svar added and fix svar l
```

2005-02-21(Mon) 08:38:02 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.h : make type "struct iseq_compile_data"  
  
* yarvcore.h : iseq_object#insn_info_ary to iseq_obje
```

2005-02-21(Mon) 05:24:01 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.c (compile_string) : remove null check of
```

2005-02-19(Sat) 03:52:45 +0900 Koichi Sasada
<ko1@atdot.net>

```
* version.h : 0.1.1
```

2005-02-18(Fri) 20:57:18 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.h, yarvcore.c : add idLTLT, idMethodMissin
* compile.c : support lval (or others) block paramet
* test/test_block.rb : add tests for above
* insns.def (send) : support method_missing
* test/test_method.rb : add tests for above
* insns.def : opt_ltlit and
```

2005-02-18(Fri) 08:54:40 +0900 Koichi Sasada
<ko1@atdot.net>

```
* benchmark/runc.rb : added
* benchmark/contrib/pentomino.rb : added opt_ltlit
and Float, String plus specialization
```

2005-02-18(Fri) 07:49:42 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : remove debug print
* rb/aotcompile.rb : skip if yarvcore.so is not creat
```

2005-02-18(Fri) 06:46:13 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : fix block passing
and block argument
```

2005-02-18(Fri) 05:52:41 +0900 Koichi Sasada
<ko1@atdot.net>

```
* vm.c : thread_get_ev_const, thread_get_ev_defined i
(separated from insns.def)
```



```
* insnhelper.h : GET_EV_KLASS(klass) is added
(separated from insns.def)

* yarvcore.h, insns.def, compile.c : support defined?

* test/test_syn.rb : tests for above is added

* compile.c, insns.def : support block passed method

* test/test_method.rb : tests for above is added

* compile.h : CALL_ARGS_SPLAT is removed
```

2005-02-16(Wed) 13:32:37 +0900 Koichi Sasada
<ko1@atdot.net>

```
* disasm.c : fix ID to String method

* compile.c : NODE_SUPER, NODE_ZSUPER check 'poped'
and NODE_RETURN check outer type
and NODE_DREGX_ONCE supported (temporarily)

* test/test_syn.rb : add a test

* test/test_jump.rb : add a test
```

2005-02-16(Wed) 06:07:41 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.[hc] : use Symbol instead of Fixnum to repr

* rb/insns2vm.rb : add attr_reader :insns, :insn_map

* vm.h, rb/insns2vm.rb : END_INSN have one arg

* jitcompile.c : jit compiler framework (experimental)

* rb/aotcompile.rb : refactoring

* compiled.c : add constant pool

* vm_evalbody.inc, call_cfunc.inc, vm.c : separated f
```

```
* insns.def : fix return val
* depend : add rules for compiled.o
```

2005-02-14(Mon) 13:09:01 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insnhelper.h, yarvcore.h: move YARV_METHOD_NODE to
* yarvcore.h : add 2 members jit_compiled and iseq_or
to struct iseq_object
* yarvcore.c : add yarv_jitcompile and global functio
* insns.def : insn opt_call_native_compiled added
* jitcompile.c : added
```

2005-02-12(Sat) 05:38:51 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def (putstring) : fixed to duplicate string o
* rb/insns2vm.rb, tmpl/optunifs.inc.tmpl, compile.c :
instructions unification (aka super instruction)
* opt_insn_unif.def : added for above
* benchmark/bm_unif1.rb : added to measure efficiency
* depend : fixed for above
* extconf.rb : add option --(enable|disalbe)-opt-insn
```

2005-02-11(Fri) 12:14:39 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.c, vm.c, insns.def : permit to access svar
cfunc environment
```

```
* test/test_method.rb : add tests for above
```

2005-02-09(Wed) 19:31:06 +0900 Koichi Sasada
<ko1@atdot.net>

```
* ite.rb : added (ruby -rite [script file])
```

2005-02-09(Wed) 02:25:43 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.[hc] : add member compile_data (hash) to i  
* compile.c, yarvcore.h : check label is already set  
* compile.c, extconf.rb : support __goto__ and __labe
```

2005-01-25(Tue) 12:49:27 +0900 Koichi Sasada
<ko1@atdot.net>

```
* test/test_block.rb : add break test to test_times
```

2005-01-25(Tue) 03:34:04 +0900 Koichi Sasada
<ko1@atdot.net>

```
* extconf.rb : check ruby version if yarv patch is ap  
* evalc.patch : fixed for rb_call_super and above che
```

2005-01-25(Tue) 03:21:48 +0900 Koichi Sasada
<ko1@atdot.net>

```
* rb/insns2vm.rb : refactoring (mainly, make InsnsDef  
to represent each instruction information)  
* depend, rb/makedocs.rb : fixed for above
```

```
* yarvcore.c (thread_call_super) : added
* vm.c (thread_call_super) : added
* vm.h : add struct cmethod_info
* insns.def, vm.c : use cmethod_info to represent C m
* insns.def : use iseq_object#klass_nest_stack
to search super/zsuper's class
* prosym.rb : removed
* ToDo : write todo things on wiki
```

2005-01-18(Tue) 23:44:47 +0900 Koichi Sasada
<ko1@atdot.net>

```
* benchmark/run.rb : check ENV['RUBY'] to use ruby bi
```

2005-01-10(Mon) 08:44:40 +0900 Koichi Sasada
<ko1@atdot.net>

```
* version.h : 0.1.0
```

2005-01-09(Sun) 22:01:29 +0900 Koichi Sasada
<ko1@atdot.net>

```
* repository : svn propset svn:eol-style native *.c *
```

2005-01-09(Sun) 21:48:38 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.c : FREE_UNLESS_NULL, MARK_UNLESS_NULL mac
* yarvcore.c : some insn/label methods are added
```

```
* yarvcore.h : add structure member "insns_ary" to is
* vm.c, insns.def (thread_eval_body) : return values
* prosym.rb : added
* insns.def : add YARV_AOT_COMPILED and some procedur
* depend : add compiled.c
* compiled.c : added to build compiled Ruby program (
by AOT compiler
* rb/aotcompile.rb : AOT compiler
* aotct.rb, rb/aotctest.rb : test and benchmark AOT c
* rb/allload.rb : added
```

2005-01-09(Sun) 08:30:38 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.c (yarv_yield_values) : added
* vm.c (thread_call0) : change interface. substitute
yarv environment
* yarvcore.c (yarv_call0) : fix for above
* yarvcore.c (yarv_call0_cfunc) : removed
* yarvcore.c : change passing items for yarv_setup
* evalc.patch : fix for above
* benchmark/bm_lists.rb : fix (unsupport block passin
* benchmark/run.rb : use full path to ruby
* insns.def (yield): raise error if argc > expected a
```

2005-01-08(Sat) 16:07:48 +0900 Koichi Sasada
<ko1@atdot.net>

```
* extconf.rb : add descriptions
* compile.c : fix bugs (getinlinecache operands)
* yarvcore.c : initial value of yarvGlobalStateVersion
to 1
```

2005-01-08(Sat) 14:39:04 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.c, vm.c, evalc.patch : support making back
(incompatible with current ruby interpreter)
```

2005-01-08(Sat) 11:25:46 +0900 Koichi Sasada
<ko1@atdot.net>

```
* evalc.patch : commit for previous commit change
* yarvcore.h, compile.c, insns.def : MC to IC (inline
and changed to using IC by set/getinlinecache
```

2005-01-08(Sat) 10:04:33 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.c : add global variable sym[IC]FUNC
* yarvcore.c (yarv_iterate, yarv_call0_cfunc) : added
(each called from rb_iterate, rb_call0 with NODE_CFUNC)
* vm.c (stack_dump_raw) : fixed to prints more detail
* vm.c (stack_dump_th, stack_dump_thobj) : added to
dumps thread_object states (for VALUE, struct pointer)
* vm.c (thread_dump_regs) : added
* vm.c (thread_call0, thread_call0_cfunc, thread_invoke,
thread_invoke_yield_cfunc), insns.def (yield, send) :
fixed, added to support IFUNC
```

```
* vm.c, yarvcore.c, insns.def : change type purpose
thread_object#block_ptr (it holds IFUNC block informa
so this type was changed to 'NODE *')

* vm.c (stack_dump_each) : fixed for above

* test/test_block.rb (test_ifunc) : test for above

* vm.c (get_block_objec, thread_make_env_object) : fi

* test/test_bin.rb (test_xstr) : remove %xls` test
```

2005-01-06(Thu) 21:35:18 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarv : trying to support NODE_IFUNC (rb_iterate)
```

2005-01-05(Wed) 06:50:42 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.h, insns.def, disasm.c, rb/insns2vm.rb, co
support inline method cache

* extconf.rb : add **-inline-method-cache (default: e

* test/test_method.rb : add a test for above

* benchmark/bm_poly_method.rb : added

* yarvcore.c : add option string
```

2005-01-04(Tue) 17:15:41 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def, compile.c : add compile_array and duparr
to optimize only literal array creation

* benchmark/bm_array.rb : added
```

2005-01-04(Tue) 10:02:40 +0900 Koichi Sasada
<ko1@atdot.net>

```
* README : fix version
```

2005-01-04(Tue) 09:57:25 +0900 Koichi Sasada
<ko1@atdot.net>

```
* ToDo : reflect current status
```

2005-01-04(Tue) 09:43:54 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : support NODE_VALUES, NODE_ARGSCAT, NODE  
* test/test_massign.rb : add tests for above  
* benchmark/bm_swap.rb : added
```

2005-01-04(Tue) 06:25:45 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.h : COMPILER_ERROR break contol (instead of  
* compile.c : support NODE_MASGN  
* insns.def : change expandarray for massign and add  
* test/test_massign.rb : added
```

2005-01-03(Mon) 21:20:28 +0900 Koichi Sasada
<ko1@atdot.net>

```
* vm.c : store block when create proc  
* test/test_proc.rb : add a test for above change  
* yarvcore.c : add global function "once"
```

2005-01-02(Sun) 00:40:08 +0900 Koichi Sasada
<ko1@atdot.net>

```
* benchmark/bm_super.rb : fix bug (remove infinite lo
```

2005-01-01(Sat) 23:45:49 +0900 Koichi Sasada
<ko1@atdot.net>

```
* benchmark/bm_z?super.rb : added
```

2005-01-01(Sat) 23:37:38 +0900 Koichi Sasada
<ko1@atdot.net>

```
* benchmark/bmx_so_object.rb : rename to benchmark/bm
```

2005-01-01(Sat) 23:19:02 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : support NODE_OP_ASGN2, NODE_OP_ASGN_AND  
NODE_SUPER, NODE_ZSUPER, NODE_MATCH
```

```
* insns.def : support super, zsuper (currently, super  
handle with block)
```

```
* test/test_bin.rb : add test for op_asgin2, op_assgi
```

```
* test/test_class.rb : add test for super, zsuper
```

2005-01-01(Sat) 20:39:29 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : support NODE_MATCH
```

```
* yarvcore.c : fix yarv_svar bug (fix condition bound
```

```
* insnhelper.h : save cfp/lfp/dfp vars to thread_obje
```

2005-01-01(Sat) 20:03:10 +0900 Koichi Sasada
<ko1@atdot.net>

```
* version.h : 0.0.1
* yarvcore.h : add idIntern declaration
* insns.def : add getspecial, setspecial.
implement getclassvariable, setclassvariable.
store lfp before reg match (opt_regexpmatch1)
* compile.c : support ditto, flipflop
* yarvcore.c : support svar
* test/test_syn.rb : add test for flipflop
* test/test_bin.rb : add test for dsym, cvar, backref
```

2005-01-01(Sat) 09:09:32 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def : add getspecial insn
* compile.c : support NODE_NTH_REF, NODE_BACK_REF
```

2005-01-01(Sat) 06:53:38 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def, compile.c : support alias, undef
* test/test_method.rb : test for above
* rb/insns2vm.rb : fix enbug
```

2005-01-01(Sat) 06:00:32 +0900 Koichi Sasada
<ko1@atdot.net>

```
* test/test_jump.rb : add test (next with value)
* yarvcore.h, yarvcore.c, compile.c, compile.h :
raise compile error exception instead of rb_bug
* yarvcore.c, evalc.patch : support "require"
* test.rb : restore $" after evaluation with ruby
* rb/insns2vm.rb : remove unnecessary each
```

2004-12-17(Fri) 18:56:38 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def : fix newhash
```

2004-12-15(Wed) 13:29:27 +0900 Koichi Sasada
<ko1@atdot.net>

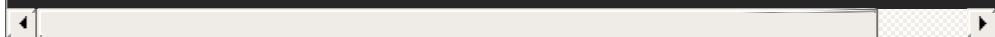
```
* yarvcore.c : add version string
* compile.c : fix rescure clause bug
```

2004-12-14(Tue) 22:46:30 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def : add reput insn
* vm.h : show stack cache registers when stack dump
* rb/insns2vm.rb, compile.c : fix stack caching bugs
```

2004-12-14(Tue) 00:51:58 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns2vm.rb, compile.c, tmpl/opt_sc.inc.tmpl : fix
* rb/mixc-asm.rb : added
```



2004-12-14(Tue) 00:17:02 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.h, yarvcore.c, compile.c : fix SC bugs  
(SC state management)  
  
* extconf.rb : add option -[enable|disable]-opt-stack  
  
* insns2vm.rb : accept CPPFLAGS options  
  
* vm.c : support restore register for pc
```

2004-12-13(Mon) 16:53:42 +0900 Koichi Sasada
<ko1@atdot.net>

```
* rb/insns2vm.rb : add macro INSN_IS_SC()
```

2004-12-11(Sat) 10:51:44 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def, compile.c : support singleton method def  
  
* test/test_method.rb : add test for above
```

2004-12-11(Sat) 03:17:54 +0900 Koichi Sasada
<ko1@atdot.net>

```
* benchmark/*.rb : modify  
  
* extconf.rb : add $cleanfiles
```

2004-12-08(Wed) 13:01:38 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c, insns.def : change to disable stack cach
```

2004-12-07(Tue) 19:37:13 +0900 Koichi Sasada
<ko1@atdot.net>

```
* rb/insns2vm.rb : add default after
* insns.def : fix to work on stack caching
```

2004-12-07(Tue) 15:07:13 +0900 Koichi Sasada
<ko1@atdot.net>

```
* depend : add some dependency to *.inc files
* vm.c : add "register" and asm("regname") descriptor
* rb/insns2vm.rb, compile.c : add stack caching support
* tmpl/opt_sc.inc.tmpl : added to above change
* rb/makedocs.rb : fix file path
* extconf.rb : fix option selection
```

2004-12-06(Mon) 11:20:11 +0900 Koichi Sasada
<ko1@atdot.net>

```
* extconf.rb : add vm.asm target if compiler is gcc
```

2004-12-06(Mon) 09:56:24 +0900 Koichi Sasada
<ko1@atdot.net>

```
* vm.h : rename method_frame's member block to block_
* extconf.rb : add "-fno-crossjumping" option when compiler
is gcc
* opt_operand.def : add unification insn send
* rb/insns2vm.rb : define symbol instead of declare c
variable (for more optimize on VC)
```

```
* insns.def : move enter point in send
```

2004-12-06(Mon) 04:53:51 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c, opt_operand.def, rb/insns2vm.rb, depend  
support operand unification
```

2004-12-05(Sun) 03:16:10 +0900 Koichi Sasada
<ko1@atdot.net>

```
* vm.c, insns.def : speed up throw/catch scheme
```

2004-12-05(Sun) 01:47:05 +0900 Koichi Sasada
<ko1@atdot.net>

```
* vm.c : fix catch handler bugs  
* test/test_jump.rb : test_complex_jump added
```

2004-12-03(Fri) 20:39:05 +0900 Koichi Sasada
<ko1@atdot.net>

```
* benchmark/contrib/mcq.rb : added  
(from URABE Syouhei)
```

2004-12-03(Fri) 20:35:28 +0900 Koichi Sasada
<ko1@atdot.net>

```
* vm.c : support break in rb_yield block
```

2004-12-03(Fri) 14:26:35 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : support block local variable in current  
ruby specification (patche from Kent Sibilev)
```

```
* insns.def : support attr_* (patch from Kent Sibilev
```

2004-12-02(Thu) 21:04:27 +0900 Koichi Sasada
<ko1@atdot.net>

```
* opt_operand.def : added
```

2004-12-02(Thu) 13:20:41 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.c, vm.h, vm.c, insns.def, insnhelper.h, ya  
add usage analisys framework
```

```
* disasm.c : insn_operand_intern to separate function
```

```
* benchmark/run.rb : run each benchmark on another pr
```

2004-12-01(Wed) 10:26:49 +0900 Koichi Sasada
<ko1@atdot.net>

```
* vm.c : yield check block is given
```

```
* benchmark/bm_lists.rb : rename to bmx_lists.rb  
(because it's not work ... bug?)
```

```
* insns.def : opt_* support other type calc
```

2004-11-30(Tue) 16:14:54 +0900 Koichi Sasada
<ko1@atdot.net>

```
* benchmark/bm_so_array.rb : added
```

```
* benchmark/bm_so_matrix.rb : added
```

2004-11-30(Tue) 14:11:30 +0900 Koichi Sasada
<ko1@atdot.net>

```
* rb/getrev.rb : added
* yarvcore.c : add YARVCore::REV, YARVCore::DATE cons
```

2004-11-30(Tue) 13:05:42 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : support NODE_OP_ASGN1 (incomplete)
* insns.def : add dupn
```

2004-11-30(Tue) 08:52:01 +0900 Koichi Sasada
<ko1@atdot.net>

```
* version.h : 0.0.0.f
```

2004-11-30(Tue) 08:43:59 +0900 Koichi Sasada
<ko1@atdot.net>

```
* test/test_class.rb : add test_initialize and test_t
* yarvsubst.c : use rb_funcall instead of yarv_funcall
* evalc.patch : fix ruby's patch
* benchmark/bm_so_*.rb : change naming rule. "bm_so_*
language shootout
* depend : tbench target item is ITEM env val (default
* vm.c : show raw address if environment is in heap a
* vm.c : thread_call0 added
* vm.c : fix thread_yield_light_invoke
* yarv.h, yarvcore.c : remove yarv_funcall
```

2004-11-29(Mon) 11:37:08 +0900 Koichi Sasada

<ko1@atdot.net>

```
* test/test/test_proc.rb : add test test_nestproc
* yarvsubst.c : comment out yarv_Array_each
* insns.def : restore lfp/dfp after call_cfunc
* vm.c : fix stack dump routine
* vm.c : impliment thread_funcall (temporarily)
* yarv.h : add IS_YARV_WORKING(), SET_YARV_START(), S
* yarvcore.c : remove check with yarv_in_work
* evalc.patch : added
```

2004-11-27(Sat) 00:19:52 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.c : free -> ruby_xfree
```

2004-11-26(Fri) 02:11:11 +0900 Koichi Sasada
<ko1@atdot.net>

```
* vm,c : fix bug
```

2004-11-22(Mon) 11:19:48 +0900 Koichi Sasada
<ko1@atdot.net>

```
* benchmark/bm_ackermann.rb, bm_proc.rb, bm_simpleite
bm_so_exception.rb, bm_wc.rb, wc.input added
```

2004-11-22(Mon) 02:31:56 +0900 Koichi Sasada
<ko1@atdot.net>

```
* test/test_proc.rb : add some test
```

```
* yarvcore.c, vm.c : support yield in C method (as rb  
* vm.c (thread_yield_light_(prepare|invoke)) : support  
yield  
* yarv.h : added  
* yarvcore.c, yarv.h : support yarv_is_working, yarv_  
yarv_yield, yarv_funcall (only dummy function)  
* vm.c : thread_eval_body changed return value  
* yarvsubst.c : added and add yarv_Integer_times, yar  
* yarvcore.h : block_ptr is added to struct thread_ob  
* insns.def : pass block when C method call  
* insnhelper.h : add GET_ISEQOBJ(cfp) macro
```

2004-11-21(Sun) 07:25:49 +0900 Koichi Sasada
<ko1@atdot.net>

```
* vm.c : support Proc#call  
* test/test_proc.rb : added
```

2004-11-19(Fri) 18:04:10 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def, vm.c : support creating Proc object
```

2004-11-15(Mon) 14:19:27 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def (send) : use clear_local_size to specify  
clear local table vars.  
* insns.def : block represent data shares lfp, dfp wi
```

2004-11-13(Sat) 18:19:41 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.h, insns.def : add VM_CALL_ARGS_SPLAT_BIT  
VM_CALL_ARGS_BLOCKARG_BIT  
  
* compile.c, compile.h : add ADD_SEND, ADD_SEND_R
```

2004-11-10(Wed) 08:26:25 +0900 Koichi Sasada
<ko1@atdot.net>

```
* add "vm_" prefix to (block_object, proc_object, env
```

2004-11-03(Wed) 15:52:14 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.h, yarvcore.c, disasm.c, compile.c, insns.  
fix to move x86_64 (illegal cast, etc)
```

2004-11-01(Mon) 04:45:54 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.h, compile.c, debug.c, version.h :  
redesign gc debug scheme (GC_CHECK())  
  
* yarvcore.c : mark iseqobj->current_block on GC  
  
* insns.def, compile.c : last "throw" in ensure/rescu  
use operand throwobj and before this insn, use "getdy  
  
* benchmark/bm_temp.rb : move to benchmark bmx_temp.r  
  
* depend : change some targets
```

2004-10-25(Mon) 19:57:58 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : push exception iseq to iseqobj->iseq_ma  
to mark for GC
```

2004-10-10(Sun) 16:25:03 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : remove $_, $' area from method local fr  
and provide that's special method local variables poi  
  
* disasm.c : change environment showing format  
  
* yarvcore.(h|c) : add YarvProc, YarvEnv  
  
* yarvcore.h : add arg_block field to iseq_object  
and init -1 as default value
```

2004-09-30(Thu) 19:50:48 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c, insns.def : support passing splat argume  
  
* compile.c, insns.def : support rest argument  
  
* compile.c, insns.def : support optional argument in  
  
* test/test_method.rb : add tests for above
```

2004-09-29(Wed) 10:50:03 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : fix rescue clause popped  
  
* benchmark/bm_random.rb : move to benchmark/bmx_rand
```

2004-09-29(Wed) 01:25:35 +0900 Koichi Sasada
<ko1@atdot.net>

```
* many many files: change stack frame design
```

2004-09-16(Thu) 08:51:37 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c, yarvcore.h : support 'return' from metho  
in ensure clause
```

2004-09-13(Mon) 21:56:40 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : support inline cache constant access  
on NODE_COLON2, NODE_COLON3  
  
* depend : add 'vtest' rule(verbose test)
```

2004-09-13(Mon) 10:58:44 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c, yarvcore.h : support redo/next/break in  
while/until
```

2004-09-13(Mon) 08:50:19 +0900 Koichi Sasada
<ko1@atdot.net>

```
* test/test_jump.rb : added(correctly)  
  
* benchmark/bm_(ensure|rescue|simplereturn).rb added
```

2004-09-12(Sun) 23:30:20 +0900 Koichi Sasada
<ko1@atdot.net>

```
* test/test_jump.rb : added  
  
* insns.def, compile.c : add 'putnil' insn  
  
* compile.c : use '===' when rescue check
```

```
* insns.def : remove 'rescuecheck' insn
* compile.c : support retry in begin/rescue clause
* ToDo : added
```

2004-09-08(Wed) 12:34:04 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvcore.h, yarvcore.c : add idThrow*
* insns.def, compile.c, vm.c : support retry, break,
next, redo, return(imcomplete)
```

2004-09-03(Fri) 13:40:08 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : add nop after rescue body
* insns.def, vm.c : support stack rewind when thrown
```

2004-09-01(Wed) 17:31:01 +0900 Koichi Sasada
<ko1@atdot.net>

```
* test/test_exception.rb : added
```

2004-09-01(Wed) 13:15:14 +0900 Koichi Sasada
<ko1@atdot.net>

```
* vm.c, insns.def : implementing exception handling
```

2004-09-01(Wed) 00:18:54 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def : add 'throw' insn
* compile.c : support 'rescue' and 'ensure' clause
```

```
* yarvcore.c, yarvcore.h : add 'catch_table' to iseq_
```

2004-08-30(Mon) 19:06:12 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.h : NEW_ISEQOBJ don't pass self as parent  
* compile.c : use NEW_CHILD_ISEQOBJ explicitly
```

2004-08-29(Sun) 21:09:55 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : trying to implement rescue/ensure  
* insns.def : fix yield bug(lfp, dfp link)
```

2004-08-28(Sat) 13:52:15 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : fix dvar bug  
* test/test_block.rb : add test  
* insns.def, insnhelper.h : remove unused source code
```

2004-08-28(Sat) 08:51:26 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : support NODE_DASGN  
* test/test_block.rb : add test
```

2004-08-28(Sat) 08:13:04 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c, insns.def : support access to instance v
```

```
* test/test_class.rb : add test of instance variable
* benchmark/bm_block.rb : added
```

2004-08-28(Sat) 07:48:43 +0900 Koichi Sasada
<ko1@atdot.net>

```
* test/test_block.rb : fix block parameter name
```

2004-08-28(Sat) 07:27:52 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c, insns.def : support method call with block
and yield and add some functions
* compile.c, insns.def : support dynavars accessor
* test/test_block.rb : added
* vm.c : fix block parameter stack dump
```

2004-08-27(Fri) 23:56:47 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c(iseq_compile) : remove parameter iseqtype
(this information can access via self)
```

2004-08-27(Fri) 17:13:35 +0900 Koichi Sasada
<ko1@atdot.net>

```
* test/test_bin.rb : add test(absolute path constant)
* yarvcore.h, compile.c(iseq_compile) : change parameter
* insns.def(classdef) : fix bug
```


2004-08-27(Fri) 04:53:13 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def : support setconstant, getconstant, class  
moduledef  
  
* vm.h : fix debug levels and so on  
  
* vm.h : foo_WORD -> foo_WC  
  
* test/test_class.rb : added
```

2004-08-25(Wed) 17:51:50 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def : fix getconstant/setconstant/classdef
```

2004-08-25(Wed) 14:27:10 +0900 Koichi Sasada
<ko1@atdot.net>

```
* debug.[ch] : added  
  
* compile.c, disasm.c : use debug interface  
  
* compile.c : support some nodes  
  
* compile.c, rb/insns2vm.rb : remove TS_CPATH  
  
* insns.def : modify classdef/moduledef/singletonclas  
and add popcref  
  
* and others...
```

2004-08-18(Wed) 20:16:45 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : fix case/when statement with empty else  
  
* insns.def : enable compile
```

```
* yarvcore.h : add class search path scheme
* test/test_syn.rb : add switch/case test case
* tmp1/yarvarch.ja : update documents
```

2004-05-22(Sat) 01:30:44 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvutil.rb : add eval_in_wrap
* test/test_*.rb : change to use eval_in_wrap
```

2004-05-20(Thu) 02:50:32 +0900 Koichi Sasada
<ko1@atdot.net>

```
* support global variables
* benchmark/bm_*.rb : add some benchmarks
* compile.c : support NODE_ATTRASGN
* compile.c : add debugi(...)
```

2004-05-19(Wed) 23:19:38 +0900 Koichi Sasada
<ko1@atdot.net>

```
* test/test_method.rb : added
```

2004-05-19(Wed) 22:56:09 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def : fix typo
* benchmark/run.rb : sort benchmark order by filename
* extconf.rb : use --enable/disable-xxx
* version.h : ditto(don't touch to change yarv option
```

2004-05-19(Wed) 21:18:55 +0900 Koichi Sasada
<ko1@atdot.net>

```
* yarvutil.rb : added
* test.rb, test/*, benchmark/run.rb : use yarvutil.rb
* version.h : USE_OPTIMIZED_REGEXP_MATCH added
* yarvcore.h : add idEqTilde
* yarvcore.c(yarvcore_parse, yarvcore_eval) : require
parameter
* test/test_bin.rb : add regexp test
* benchmark/bm_regexp.rb : added
```

2004-05-19(Wed) 13:57:31 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : add compile_dstr(self, node)
* compile.c : support NODE_MATCH2, NODE_MATCH3, NODE_
* insns.def : add toregexp
```

2004-05-18(Tue) 10:12:20 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : support NODE_XDSTR
* test/test_bin.rb : add test for above change
```

2004-05-18(Tue) 09:46:33 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def(send) : store regs before call_cfunc
```

2004-05-18(Tue) 08:55:17 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : support NODE_DSTR, NODE_EVSTR
* compile.c : support NODE_XSTR
* insns.def : add tostring operation
* rb/makedocs.rb : fix directory path
* depend : add tbench rule
* yarvcore.h : add 'exten ID idBackquote'
```

2004-05-18(Tue) 00:09:48 +0900 Koichi Sasada
<ko1@atdot.net>

```
* version.h : add USE_OPTIMIZED_BASIC_OPERATION
* yarvcore.h(struct thread_object) : add 'VALUE stat_
```

2004-05-17(Mon) 11:28:55 +0900 Koichi Sasada
<ko1@atdot.net>

```
* version.h, insns.def, yarvcore.c : add FAKE_INLINE_
```

2004-05-17(Mon) 09:05:53 +0900 Koichi Sasada
<ko1@atdot.net>

```
* compile.c : fix generating opt_* insn process
```

2004-05-17(Mon) 08:58:49 +0900 Koichi Sasada
<ko1@atdot.net>

```
* benchmark/(bm_tarai.rb, bm_fib.rb) : added
```

2004-05-17(Mon) 08:20:12 +0900 Koichi Sasada
<ko1@atdot.net>

```
* benchmark/(bm_tak.rb, bm_reccount.rb) : added
* insns.def : test method cache(incomplete)
* insns.def : add expandarray insn
* yarvcore.c(iseq_init) : add parameter 'parent'
```

2004-05-17(Mon) 01:49:48 +0900 Koichi Sasada
<ko1@atdot.net>

```
* benchmark/run.rb, bm_factorial.rb, bm_whileloop.rb
* insns.def(send) : set id to ruby_frame->orig_func
* check behavior on mswin32 and cygwin
* insns.def(send) : check stack overflow
```

2004-05-16(Sun) 08:00:55 +0900 Koichi Sasada
<ko1@atdot.net>

```
* change frame structure(debugging)
```

2004-05-14(Fri) 15:06:02 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns2vm.rb : support file name arguments
```

2004-05-14(Fri) 04:33:09 +0900 Koichi Sasada
<ko1@atdot.net>

```
* insns.def : support (easy) constant
```

2004-05-12(Wed) 01:51:48 +0900 Koichi Sasada
<ko1@atdot.net>

```
* rb/insns2vm.b : set directory prefix
* disasm.c : fix bug
```

2004-05-12(Wed) 00:00:17 +0900 Koichi Sasada
<ko1@atdot.net>

```
* vm.h, compiler.h, version.h : move *DEBUG defs to v
```

2004-05-11(Tue) 23:00:11 +0900 Koichi Sasada
<ko1@atdot.net>

```
* vm.h, version.h, yarvcore.h : move gcc ver check to
and include version.h from yarvcore.h
```

2004-05-11(Tue) 19:16:26 +0900 Koichi Sasada
<ko1@atdot.net>

```
* 0.0.0.d : imported
```

Local variables: add-log-time-format: (lambda ())

```
(let* ((time (current-time))
      (diff (+ (cadr time) 32400))
      (lo (% diff 65536))
      (hi (+ (car time) (/ diff 65536))))
  (format-time-string "%Y-%m-%d(%a) %H:%M:%S +900" (li
```

indent-tabs-mode: t tab-width: 8 end:

Generated by [RDoc](#) 3.12.2.

Generated with the [Darkfish Rdoc Generator](#) 3.

Contributing to Ruby

Ruby has a vast and friendly community with hundreds of people contributing to a thriving open-source ecosystem. This guide is designed to cover ways for participating in the development of CRuby.

There are plenty of ways for you to help even if you're not ready to write code or documentation. You can help by reporting issues, testing patches, and trying out beta releases with your applications.

How To Report

If you've encountered a bug in Ruby please report it to the redmine issue tracker available at bugs.ruby-lang.org. Do not report security vulnerabilities here, there is a [separate channel](#) for them.

There are a few simple steps you should follow in order to receive feedback on your ticket.

- If you haven't already, [sign up for an account](#) on the bug tracker.
- Try the latest version.
If you aren't already using the latest version, try installing a newer stable release. See [Downloading Ruby](#).
- Look to see if anyone already reported your issue, try [searching on redmine](#) for your problem.
- If you can't find a ticket addressing your issue, [create a new one](#).
- Choose the target version, usually current. Bugs will be first fixed in the current release and then [backported](#)
- Fill in the Ruby version you're using when experiencing this issue (`ruby -v`).
- Attach any logs or reproducible programs to provide additional information. Reproducible scripts should be as small as possible.
- Briefly describe your problem. A 2-3 sentence description will help give a quick response.
- Pick a category, such as core for common problems, or lib for a standard library.
- Check the [Maintainers list](#) and assign the ticket if there

is an active maintainer for the library or feature.

- If the ticket doesn't have any replies after 10 days, you can send a reminder.
- Please reply to feedback requests. If a bug report doesn't get any feedback, it'll eventually get rejected.

Reporting to downstream distributions

You can reports downstream issues for the following distributions via their bugtracker:

- [debian](#)
- [freebsd](#)
- [redhat](#)
- [macports](#)
- etc (add your distribution bug tracker here)

Platform Maintainers

For platform specific bugs in Ruby, you can assign your ticket the current maintainer for a specific platform.

The current active platform maintainers are as follows:

mswin32, mswin64 (Microsoft Windows)

NAKAMURA Usaku (usa)

mingw32 (Minimalist GNU for Windows)

Nobuyoshi Nakada (nobu)

IA-64 (Debian GNU/Linux)

TAKANO Mitsuhiro (takano32)

Symbian OS

Alexander Zavorine (azov)

AIX

Yutaka Kanemoto (kanemoto)

FreeBSD

Akinori MUSHA (knu)

Solaris

Naohisa Goto (ngoto)

RHEL, CentOS

KOSAKI Motohiro kosaki

Mac OS X

Kenta Murata (mrkn)

cygwin, bcc32, djgpp, wince, ...

none. (Maintainer WANTED)

Reporting Security Issues

Security vulnerabilities receive special treatment since they may negatively affect many users. There is a private mailing list that all security issues should be reported to and will be handled discretely. Email the security@ruby-lang.org list and the problem will be published after fixes have been released. You can also encrypt the issue using [the PGP public key](#) for the list.

Reporting Other Issues

If you're having an issue with the website, or maybe the mailing list, you can contact the webmaster to help resolve the problem.

The current webmaster is:

- Hiroshi SHIBATA (hsbt)

You can also report issues with the ruby-lang.org website on the issue tracker:

- [issue tracker](#)

Resolve Existing Issues

As a next step beyond reporting issues you can help the core team resolve existing issues. If you check the Everyone's Issues list in GitHub Issues, you'll find lots of issues already requiring attention. What can you do for these? Quite a bit, actually:

When a bug report goes for a while without any feedback, it goes to the bug graveyard which is unfortunate. If you check the [issues list](#) you'll find lots of delinquent bugs that require attention.

You can help by verifying the existing tickets, try to reproduce the reported issue on your own and comment if you still experience the bug. Some issues lack attention because of too much ambiguity, to help you can narrow down the problem and provide more specific details or instructions to reproduce the bug. You might also try contributing a failing test in the form of a patch, which we will cover later in this guide.

It may also help to try out patches other contributors have submitted to redmine, if gone without notice. In this case the `patch` command is your friend, see `man patch` for more information. Basically this would go something like this:

```
cd path/to/ruby/trunk
patch -p0 < path/to/patch
```

You will then be prompted to apply the patch with the associated files. After building ruby again, you should try to run the tests and verify if the change actually worked or fixed the bug. It's important to provide valuable feedback on the patch that can help reach the overall goal, try to answer some of these questions:

- What do you like about this change?
- What would you do differently?
- Are there any other edge cases not tested?
- Is there any documentation that would be affected by this change?

If you can answer some or all of these questions, you're on the right track. If your comment simply says "+1", then odds are that other reviewers aren't going to take it too seriously. Show that you took the time to review the patch.

How To Request Features

If there's a new feature that you want to see added to Ruby, you'll need to write a convincing proposal and patch to implement the feature.

For new features in CRuby, use the ['Feature' tracker](#) on ruby-trunk. For non-CRuby dependent features, features that would apply to alternate Ruby implementations such as JRuby and Rubinius, use the [CommonRuby tracker](#).

When writing a proposal be sure to check for previous discussions on the topic and have a solid use case. You will need to be persuasive and convince Matz on your new feature. You should also consider the potential compatibility issues that this new feature might raise.

Consider making your feature into a gem, and if there are enough people who benefit from your feature it could help persuade ruby-core. Although feature requests can seem like an alluring way to contribute to Ruby, often these discussions can lead nowhere and exhaust time and energy that could be better spent fixing bugs. Choose your battles.

A good template for feature proposal should look something like this:

| |
|--|
| Abstract |
| Summary of your feature |
| Background |
| Describe current behavior and why it is problem. |

Related work, such as solutions in other language helps us to understand the problem.

Proposal

Describe your proposal in details

Details

If it has complicated feature, describe it

Usecase

How would your feature be used? Who will benefit from it?

Discussion

Discuss about this proposal. A list of pros and cons will help start discussion.

Limitation

Limitation of your proposal

Another alternative proposal

If there are alternative proposals, show them.

See also

Links to the other related resources

Slideshow

On Ruby Developer Meeting Japan, committers discuss about Feature Proposals together at Tokyo. We'll judge proposals accept, reject, or feedback. If you have a stalled proposal, making a slide to submit is good way to get feedback.

Slides should be:

- One-page slide
- Include a corresponding ticket number
- MUST include a figure and/or short example code
- SHOULD have less sentence in natural language (try to write less than 140 characters)
- It is RECOMMENDED to itemize: motivation/use case, proposal, pros/cons, corner case
- PDF or Image (Web browsers can show it)

Please note:

- Even if the proposal is generally acceptable, it won't be accepted without writing corner cases in the ticket
- Slide's example: `DevelopersMeeting20130727Japan`

Backport Requests

When a new version of Ruby is released it starts at patch level 0 (p0), and bugs will be fixed first on the trunk branch. If its determined that a bug exists in a previous version of Ruby that is still in the bug fix stage of maintenance, then a patch will be backported. After the maintenance stage of a particular Ruby version ends, it goes into “security fix only” mode which means only security related vulnerabilities will be backported. Versions in End-of-life (EOL) will not receive any updates and it is recommended you upgrade as soon as possible.

If a major security issue is found or after a certain amount of time since the last patch level release, a new patch-level release will be made.

When submitting a backport request please confirm the bug has been fixed in newer versions and exists in maintenance mode versions. There is a backport tracker for each major version still in maintenance where you can request a particular revision merged in the affected version of Ruby.

Each major version of Ruby has a release manager that should be assigned to handle backport requests. You can find the list of release managers on the [wiki](#).

Branch Maintainers

A branch maintainer maintains a branch and releases a new release of Ruby. The branch depends on the associated version of Ruby, such as `ruby_1_8_7` for version 1.8.7. The current branch maintainers are as follows:

trunk

unnecessary

ruby_2_0_0

Chikanaga Tomoyuki (nagachika)

ruby_1_9_3

NAKAMURA Usaku (usa)

ruby_1_9_2, ruby_1_9_1

unmaintained

ruby_1_8

unmaintained

ruby_1_8_7

unmaintained

ruby_1_8_6 ...

unmaintained

Running tests

In order to help resolve existing issues and contributing patches to Ruby you need to be able to run the test suite.

CRuby uses subversion for source control, you can find installation instructions and lots of great info to learn subversion on the svnbook.red-bean.com. For other resources see the [ruby-core documentation on ruby-lang.org](http://ruby-core.org/ruby-core).

This guide will use git for contributing. The [git homepage](http://git-scm.com) has installation instructions with links to documentation for learning more about git. There is a mirror of the subversion repository on [github](https://github.com).

Install the prerequisite dependencies for building the CRuby interpreter to run tests.

- C compiler
- autoconf
- bison
- gperf
- ruby - Ruby itself is prerequisite in order to build Ruby from source. It can be 1.8.

You should also have access to development headers for the following libraries, but these are not required:

- Tcl/Tk
- NDBM/QDBM
- GDBM
- OpenSSL
- readline/editline(libedit)

- zlib
- libffi
- libyaml
- libexecinfo (FreeBSD)

Now let's build CRuby:

- Checkout the CRuby source code:

```
git clone git://github.com/ruby/ruby.git ruby-trunk
```

- Generate the configuration files and build:

```
cd ruby-trunk
autoconf
mkdir build && cd build # its good practice to bui
mkdir ~/.rubies # we will install to .rubies/ruby-
../configure --prefix=~/.rubies/ruby-trunk
make && make install
```

After adding Ruby to your PATH, you should be ready to run the test suite:

```
make test
```

You can also use `test-all` to run all of the tests with the `RUNRUBY` interpreter just built. Use `TESTS` or `RUNRUBYOPT` to pass parameters, such as:

```
make test-all TESTS=-v
```

This is also how you can run a specific test from our build dir:

```
make test-all TESTS=drb/test_drb.rb
```

For older versions of Ruby you'll need to run the build setup again after checking out the associated branch in git, for example if you wanted to checkout 1.9.3:

```
git clone git://github.com/ruby/ruby.git --branch rub
```

Contributing Documentation

If you're interested in contributing documentation directly to CRuby there is a wealth of information available at documenting-ruby.org.

There is also the [Ruby Reference Manual](#) in Japanese.

Contributing A Patch

Deciding what to patch

Before you submit a patch, there are a few things you should know:

- Pay attention to the maintenance policy for stable and maintained versions of Ruby.
- Released versions in security mode will not merge feature changes.
- Search for previous discussions on ruby-core to verify the maintenance policy
- Patches must be distributed under Ruby's license.
- This license may change in the future, you must join the discussion if you don't agree to the change

To improve the chance your patch will be accepted please follow these simple rules:

- Bug fixes should be committed on trunk first
- Format of the patch file must be a unified diff (ie: diff -pu, svn diff, or git diff)
- Don't introduce cosmetic changes
- Follow the original coding style of the code
- Don't mix different changes in one commit

First thing you should do is check out the code if you haven't already:

```
git clone git://github.com/ruby/ruby.git ruby-trunk
```

Now create a dedicated branch:

```
cd ruby-trunk
git checkout -b my_new_branch
```

The name of your branch doesn't really matter because it will only exist on your local computer and won't be part of the official Ruby repository. It will be used to create patches based on the differences between your branch and trunk, or edge Ruby.

Coding style

Here are some general rules to follow when writing Ruby and C code for CRuby:

- Indent 4 spaces for C with tabs for eight-space indentation (emacs default)
- Indent 2 space tabs for Ruby
- Do not use TABs in ruby codes
- ANSI C style for 1.9+ for function declarations
- Follow C90 (not C99) Standard
- PascalStyle for class/module names.
- UNDERSCORE_SEPARATED_UPPER_CASE for other constants.
- Capitalize words.
- ABBRs should be all upper case.
- Do as others do

ChangeLog

Although not required, if you wish to add a ChangeLog entry for your change please note:

You can use the following template for the ChangeLog

entry on your commit:

```
Thu Jan  1 00:00:00 2004  Your Name  <yourmail@exampl
    * filename (function): short description of thi
      This should include your intention of this ch
      [bug:#number] [mailinglist:number]
    * filename2 (function2): additional description
```

This follows [GNU Coding Standards for Change Logs](#),
some other requirements and tips:

- Timestamps must be in JST (+09:00) in the style as above.
- Two spaces between the timestamp and your name.
Two spaces between your name and your mail address.
- One blank line between the timestamp and the description.
- Indent the description with TAB. 2nd line should begin with TAB+2SP.
- Write a entry (*) for each change.
- Refer to redmine issue or discussion on the mailing list.
- For GitHub issues, use [GH-#] (such as [Fixes GH-234])
- One blank line between entries.
- Do as other committers do.

You can generate the ChangeLog entry by running `make change`

When you're ready to commit, copy your ChangeLog entry into the commit message, keeping the same formatting and select your files:

```
git commit ChangeLog path/to/files
```

In the likely event that your branch becomes outdated, you will have to update your working branch:

```
git fetch origin
git rebase remotes/origin/master
```

Now that you've got some code you want to contribute, let's get set up to generate a patch. Start by forking the github mirror, check the [github docs on forking](#) if you get stuck here. here. You will only need a github account if you intend to host your repository on github.

Next copy the writable url for your fork and add it as a git remote, replace "my_username" with your github account name:

```
git remote add my_fork git@github.com:my_username/rub
# Now we can push our branch to our fork
git push my_fork my_new_branch
```

In order to generate a patch that you can upload to the bug tracker, we can use the github interface to review our changes just visit

github.com/my_username/ruby/compare/trunk...my_new_branch

Next, you can simply add '.patch' to the end of this URL and it will generate the patch for you, save the file to your computer and upload it to the bug tracker. Alternatively you can submit a pull request, but for the best chances to receive feedback add it is recommended you add it to redmine.

Since git is a distributed system, you are welcome to host your git repository on any [publically accessible hosting site](#), including [hosting your own](#) You may use the '[git format-patch](#)' command to generate patch files to upload

to redmine. You may also use the [‘git request-pull’](#) command for formatting pull request messages to redmine.

Generated by [RDoc 3.12.2](#).

Generated with the [Darkfish Rdoc Generator 3](#).

Contributors to Ruby

The following list might be incomplete. Feel free to add your name if your patch was accepted into Ruby.

A

Ayumu AIZAWA (ayumin)

- committer

AKIYOSHI, Masamichi (akiyoshi)

- committer
- He had maintained the VMS support on 2003-2004.

Muhammad Ali

- wrote rdoc for Fiber

Minero Aoki (aamine)

- committer
- He is the maintainer of:
 - fileutils
 - net/http, net/https
 - net/pop
 - net/smtp
 - racc
 - ripper
 - strscan

Wakou Aoyama (wakou)

- committer
- He was the maintainer of some standard libraries.

Koji Arai

- committer

arton

- He is the distributor of ActiveScriptRuby and experimental 1.9.0-x installers for win32.
- Wrote patches for win32ole, gc.c, tmpdir.rb

B

Daniel Berger

- a patch for irb
- documentation
- He wrote forwardable.rb

David Black (dblack)

- committer
- He is the maintainer of scanf

Ken Bloom

- a patch for REXML.

Oliver M. Bolzer

- a patch for soap

Alexey Borzenkov

- a patch for mkmf.rb

Richard Brown

- a patch for configure.in

Dirkjan Bussink

- a patch for date.rb

Daniel Bovensiepen

- documentation

- a patch for irb

C

Brian Candler

- a patch for `configure.in`, `net/telnet`

keith cascio

- a patch for `optparse.rb`

Frederick Cheung

- a patch for `test/ruby/test_symbol.rb`

Christoph

- patches for `set.rb`

Sean Chittenden

- patches for `net/http`, `cgi`

William D. Clinger

- `ruby_strtod` is based on his paper.

D

Ryan Davis (ryan)

- committer
- He wrote and is the maintainer of miniunit

Guy Decoux (ts)

- committer

Zach Dennis

Martin Duerst (duerst)

- committer
- M17N

Paul Duncan

- pathces for rdoc

Alexander Dymo

- a patch for lib/benchmark.rb

E

Yusuke Endoh (mame)

- committer
- He wrote and is the maintainer of base64 library (1.9)
- did much upon YARV compiler.

erlercw

- wrote Integer::gcd2

F

Frank S.Fejes

- a patch for net/pop

Fundakowski Feldman

- a patch for process.c

Mauricio Fernandez

- patches for parse.y

David Flanagan (davidflanagan)

- committer
- M17N

Takeyuki Fujioka (xibbar)

- committer
- He is the maintainer of cgi/*

FUKUMOTO, Atsushi

- a patch for tracer.rb

Shota Fukumori (sorah)

- committer
- #4415 parallel unit/test

Tadayoshi Funaba (tadf)

- committer

- He wrote and is the maintainer of
 - `date`
 - `parsedate` (1.8)
- He ported `rational.rb` and `complex.rb`, which 1.8 contains, into `rational.c` and `complex.c` of 1.9.

G

David M. Gay

- `ruby_strtod`

Florian Gilcher

- `documentation`

GOTOU, Kentaro (gotoken)

- `committer`
- He wrote `benchmark.rb`
- He is the maintainer of:
 - `benchmark.rb`
 - `open3`

GOTOU, Yuuzou (gotoyuzo)

- `committer`

James Edward Gray II (jeg2)

- `committer`
- He wrote the faster implementation of CSV and is the maintainer of `csv`.
- Wrote documentation for `rdoc`

H

Phil Hagelberg

- patch for ruby-mode.el's documentation.

Kirk Haines (wyhaines)

- committer
- the maintainer of ruby_1_8_6 branch

Shinichiro Hamaji

- fixed memory leaks (marshal.c, string.c)

Shin-ichiro HARA

- the developer and the sysop of ruby-{dev,list,core,talk} archive.
- a patch for numeric.c

Chris Heath (traumdeutung)

- a patch for proc.c

HIROKAWA Hisashi

- fixed socket/socket.c

Daniel Hob

- He wrote:
 - SMTP-TLS support for net/smtp.
 - POP3S support

Eric Hodel (drbrain)

- committer
- He is the maintainer of:
 - rdoc
 - ri
 - rubygems

Erik Hollensbe

- a patch for delegate.rb

Johan Holmberg

- a patch for dir.c
- documentation

Erik Huelsmann

Dae San Hwang

- built a continuous integration environment on OpenSolaris.

|

Nobuhiro IMAI

- a patch for logger.rb

“incorporate”

- a patch for sprintf.c

Keiju Ishitsuka (keiju)

- committer
- He wrote and is the maintainer of:
 - cmath.rb (1.9)
 - complex.rb (1.8)
 - e2mmap.rb
 - forwardable.rb
 - irb
 - mathn
 - matrix.rb
 - mutex_m.rb
 - rational.rb (1.8)
 - sync.rb
 - shell/*
 - thwait.rb
 - tracer.rb

J

Curtis Jackson

- missing/dup2.c

Alan Johnson

- a patch for net/ftp

Lyle Johnson

- patches for nkf, bigdecimal, numeric.c

K

Yoshihiro Kambayashi

- a patch for enc/trans/single_byte.trans.
- He wrote supports for some encodings.

Yutaka Kanemoto

- patches for common.mk, AIX AF_INET6 support

Motoyuki Kasahara

- He wrote getoptlong.rb

Masahiro Kawato

- a patch for shellwords.rb

Wataru Kimura

- a patch for configure.in

Michael Klishin

- patch for make help.

Noritada Kobayashi

- a patch for optparse.rb

Shigeo Kobayashi (shigek)

- committer
- He is the maintainer of bigdecimal

KONISHI, Hiromasa (H_Konishi)

- committer
- He had maintained the bcc32 support in 2004.

Kornelius “murphy” Kalnbach

- documentation

K.Kosako (kosako)

- committer
- He wrote Oniguruma.

Takehiro Kubo

- patches for dl 64bit support.

L

Marc-Andre Lafortune (marcandre)

- committer
- patches for hash.c, array.c, thread.c, enumc, string.c, range.c and rdoc documentation.

Hongli Lai

- improved pstore.rb
- patch for tool/file2lastrev.rb.

raspberry lemon

- a patch for webrick/httpproxy.rb.

Christian Loew

- a patch for fileutils.rb

M

Shugo Maeda (shugo)

- committer
- A system administrator of ruby-lang.org servers.
- He wrote and is the maintainer of:
 - monitor.rb
 - net/ftp
 - net/imap

Stephan Maka (mathew)

- documentation

Yukihiro Matsumoto (matz)

- Matz – the founder, language designer of Ruby.
- committer
- Ruby itself, most of Ruby.
- He is the maintainer of:
 - singleton
 - timeout
 - gdbm
 - sdbm

Konrad Meyer

- documentation

Mib Software

- missing/vsnprintf.c

Todd C. Miller

- missing/strlcat.c
- missing/strlcpy.c

MIYASAKA, Masaru

- a patch for cgi.rb

Stefan Monnier

- regex.c was fixed with based on his Emacs21 patch.

Marcel Moolenaar

- patches for eval.c and gc.c.

moonwolf

- a patch for REXML, xmlrpc

Hiroshi Moriyama

- a patch for yaml.

Kyosuke Morohashi

- a patch for gem_prelude.rb

Kenta Murata

- patches for json, bignum.c

Akinori MUSA (knu)

- committer
- He wrote and is the maintainer of:
 - abbrev.rb

- generator (1.8)
- enumerator (1.8)
- set
- ipaddr.rb
- digest/*
- syslog
- He is the branch maintainer of ruby_1_8, the release manager of 1.8 series.

N

Hidetoshi NAGAI (nagai)

- committer
- He is the maintainer of tk/*

Nobuyoshi Nakada (nobu)

- committer
- a.k.a. the “patch monster”
- He wrote and is the maintainer of:
 - optparse
 - stringio
 - io/wait
 - iconv

Satoshi Nakagawa

- patches for util.c

Narihiro Nakamura (nari)

- committer
- a.k.a. authorNari
- working at GC

NAKAMURA, Hiroshi (nahi)

- committer
- He is the maintainer of:
 - csv.rb (1.8)
 - logger.rb
 - soap/* (1.8)

- wsdl/* (1.8)
- xsd/* (1.8)

NAKAMURA, Usaku (usa)

- committer
- a.k.a. unak
- He is the maintainer of mswin32 and mswin64 support.

NARUSE, Yui (naruse)

- committer
- a.k.a. “nurse”
- Did much upon m17n.
- He is the maintainer of:
 - json
 - nkf

Christian Neukirchen

- a patch for webrick/httputils

Michael Neumann (mneumann)

- committer
- He is the maintainer of
 - xmlrpc (1.8)
 - gserver (1.8)

NISHIO Hirokazu

- wrote a patch for CVE-2010-0541

Kazuhiro NISHIYAMA (kazu)

- committer

- a.k.a. znz

Go Noguchi

Martin Nordholts

- misc/rdebug.el

nmu

- a patch for socket

O

okkez

- He is a sysop of the Ruby Reference Manual Renewal Project.
- fixed ipaddr.rb, ext/etc

Haruhiko Okumura

- some of missing/* is based on his book:
 - missing/erf.c
 - missing/lgamma_r.c
 - missing/tgamma.c

OMAE, jun

- a patch for debug.rb

Eugene Ossintsev

- documentation

P

Heesob Park

- a patch for win32/win32.c.

pegacorn

- a patch for instruby.rb

Q

R

Gaston Ramos

- documentation

The Regents of the University of California

- missing/crypt.c
- missing/vsnprintf.c

Sam Roberts

- patch for socket
- documentation

Michal Rokos (michal)

- committer
- He was the maintainer of DJGPP support.

rubikitch

- a patch for io.c

Marcus Rueckert

- a patch for mkconfig.rb.

Run Paint Run Run

- patch for enc/unicode.c

- documentation

Sean Russell (ser)

- committer
- He wrote and is the maintainer of REXML.

S

Kazuo Saito (ksaito)

- committer
- M17N

Tadashi Saito

- patches for test/ruby/test_math.rb, thread_*.c, bignum.c
- working upon BigDecimal.
- did much upon documentation

Masahiro Sakai

- a patch for io.c

Laurent Sansonetti

- a patch for tool/ytab.sed

Jeff Saracco

- documentation

Koichi Sasada (ko1)

- committer
- He wrote YARV.

Hugh Sasse

- a patch for net/http
- documentation

Charlie Savage

- a patch for win32/Makefile.sub

Michael Scholz

- a patch for ruby-mode.el

Arthur Schreiber

- patch for net/http and rdoc.

Masatoshi SEKI (seki)

- committer
- He wrote and is the maintainer of:
 - drb/*
 - erb
 - rinda

Roman Shterenzon

- a patch for open-uri.

Kent Sibilev

Gavin Sinclair (gsinclair)

- committer

John W. Small

- He wrote gserver.rb

Yuki Sonoda (yugui)

- committer
- She is the maintainer of man/* manual pages and is the

release manager of 1.9 series.

- She wrote prime.rb.
- A developer and a sysop of redmine.ruby-lang.org.

SOUMA, Yutaka

- a patch for pack.c.

Tatsuki Sugiura

- WebDAV support for net/http

Masaki Suketa (suke)

- committer
- He is the maintainer of win32ole

sheepman

- patches for ruby.c, thread.c, stringio, enum.c, webrick, net/http

Siena. (siena)

- committer

Kirill A. Shutemov

- a patch for parse.y

Darren Smith

- a patch for golf_prelude.rb

Richard M. Stallman

- missing/alloca.c

Robin Stocker

- documentation

Adam Strzelecki

- a patch for compile.c

Masashi Sumi

- improved net/pop.rb

Eric Sunshine

- NeXT OpenStep, Rhapsody support

Kouhei Sutou (kou)

- committer
- He wrote and is the maintainer of `rss/*`

David Symonds

- documentation

T

TAKANO Mitsuhiro (takano32)

- committer
- He is the maintainer of IA-64 support.
- BigDecimal

TAKAO, Kouji (kouji)

- committer
- He is the maintainer of readline.

Nathaniel Talbott (ntalbott)

- committer
- He was the maintainer of test/unit, runit, rubyunit.

TANAKA, Akira (akr)

- committer
- Did much upon m17n.
- And he is the maintainer of:
 - open-uri
 - pathname
 - pp
 - resolv-replace
 - resolv
 - time
 - tsort

Takaaki Tateishi (ttate)

- committer

- He was the maintainer of dl

Technorama Ltd. (technorama)

- committer
- openssl

Andrew Thompson

- a patch for socket.c IRIX support.

Dave Thomas (dave)

- committer
- a.k.a. the Pragmatic Programmer.
- He wrote rdoc.

Tietew

- patches for win32 support

Masahiro Tomita

- a patch for cgi.rb

Jakub Travník

- a patch for eval.c

Tom Truscott

- missing/crypt.c

U

UEDA, Satoshi

- a patch for uri

Takaaki Uematsu (uema2)

- committer
- He was the maintainer of WinCE support.

UENO, Katsuhiko (katsu)

- committer
- He is the maintainer of zlib

Hajimu UMEMOTO

- He wrote ipaddr.rb

URABE, Shyouhei (shyouhei)

- committer
- a.k.a. mput.
- He is the branch maintainer of ruby_1_8_6 and ruby_1_8_7
- and is the release manager of 1.8.x-pXXX.

V

Joel VanderWerf

- a patch for numeric.c

Peter Vanbroekhoven

Corinna Vinschen

W

wanabe (wanabe)

- committer
- fixed YARV and Oniguruma.

Chun Wang

- a patch for time.rb

WATANABE, Hirofumi (eban)

- committer
- He is the maintainer of
 - ftools (1.8)
 - tmpdir
 - un
 - Win32API

WATANABE, Tetsuya

- a patch for ruby.c

William Webber (wew)

- committer

Jim Weirich (jim)

- committer
- He wrote Rake.

Nathan Weizenbaum

- fixed misc/ruby-mode.el.

why the lucky stiff (why)

- committer
- He is the maintainer of syck

Caley Woods

- documentation

Gary Wright

- documentation

X

Y

Akira Yamada (akira)

- committer
- He is the maintainer of ruby related packages at Debian project.

Keita Yamaguchi

- patches for enum.c, parse.y
- documentation

Hirokazu Yamamoto (ocean)

- committer

Hiroataka Yoshioka

- a patch for improving SEGV handling

Z

Aristarkh A Zagorodnikov

- a patch for io.c

Alexander Zavorine

- committer
- He is the maintainer for Symbian OS.

Chiyuan Zhang

- a patch for misc/ruby-mode.el.

Dee Zsombor (zunda)

- a patch for thread_pthread.c

Dan Zwell

- a patch for net/pop

Generated by [RDoc](#) 3.12.2.

Generated with the [Darkfish Rdoc Generator](#) 3.

DTrace Probes

A list of DTrace probes and their functionality. “Module” and “Function” cannot be defined in user defined probes (known as USDT), so they will not be specified. Probe definitions are in the format of:

```
provider:module:function:name(arguments)
```

Since module and function cannot be specified, they will be blank. An example probe definition for Ruby would then be:

```
ruby::function-entry(class name, method name, file n
```

Where “ruby” is the provider name, module and function names are blank, the probe name is “function-entry”, and the probe takes four arguments:

- class name
- method name
- file name
- line number

Probes List

Stability

Before we list the specific probes, let's talk about stability. Probe stability is declared in the probes.d file at the bottom on the pragma D attributes lines. Here is a description of each of the stability declarations.

Provider name stability

The provider name of “ruby” has been declared as stable. It is unlikely that we will change the provider name from “ruby” to something else.

Module and Function stability

Since we are not allowed to provide values for the module and function name, the values we have provided (no value) is declared as stable.

Probe name stability

The probe names are likely to change in the future, so they are marked as “Evolving”. Consumers should not depend on these names to be stable.

Probe argument stability

The parameters passed to the probes are likely to change in the future, so they are marked as “Evolving”. Consumers should not depend on

these to be stable.

Declared probes

Probes are defined in the probes.d file. Here are the declared probes along with when they are fired and the arguments they take:

```
ruby::method-entry(classname, methodname,  
filename, lineno);
```

This probe is fired just before a method is entered.

```
classname name of the class (a string)  
methodname name of the method about to be ex  
filename the file name where the method is _  
lineno the line number where the method is _
```

```
ruby::method-return(classname, methodname,  
filename, lineno);
```

This probe is fired just after a method has returned. The arguments are the same as “ruby::function-entry”.

```
ruby::cmethod-entry(classname, methodname,  
filename, lineno);
```

This probe is fired just before a C method is entered. The arguments are the same as “ruby::function-entry”.

```
ruby::cmethod-return(classname,
```


methodname, filename, lineno);

This probe is fired just before a C method returns. The arguments are the same as “ruby::function-entry”.

ruby::require-entry(requiredfile, filename, lineno);

This probe is fired on calls to `rb_require_safe` (when a file is required).

```
requiredfile is the name of the file to be r  
filename is the file that called "require" (  
lineno is the line number where the call to
```

ruby::require-return(requiredfile, filename, lineno);

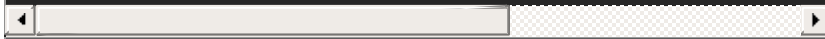
This probe is fired just before `rb_require_safe` (when a file is required) returns. The arguments are the same as “ruby::require-entry”. This probe will not fire if there was an exception during file require.

ruby::find-require-entry(requiredfile, filename, lineno);

This probe is fired right before `search_required` is called. `search_required` determines whether the file has already been required by searching loaded features (“\$”), and if not, figures out which file must be loaded.

```
requiredfile is the file to be required (str  
filename is the file that called "require" (
```

```
lineno is the line number where the call to
```



ruby::`find-require-return`(requiredfile, filename, lineno);

This probe is fired right after `search_required` returns. See the documentation for “`ruby::find-require-entry” for more details. Arguments for this probe are the same as “ruby::find-require-entry”.`

ruby::`load-entry`(loadedfile, filename, lineno);

This probe is fired when calls to “load” are made. The arguments are the same as “`ruby::require-entry”.`

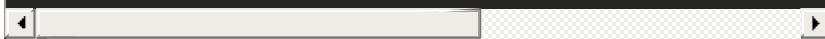
ruby::`load-return`(loadedfile, filename, lineno);

This probe is fired when “load” returns. The arguments are the same as “`ruby::load-entry”.`

ruby::`raise`(classname, filename, lineno);

This probe is fired when an exception is raised.

```
classname is the class name of the raised ex  
filename the name of the file where the exce  
lineno the line number in the file where the
```

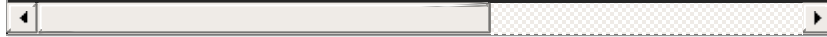


ruby::`object-create`(classname, filename, lineno);

This probe is fired when an object is about to be

allocated.

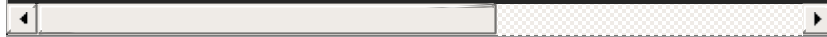
```
classname the class of the allocated object  
filename the name of the file where the object  
lineno the line number in the file where the
```



ruby::array-create(length, filename, lineno);

This probe is fired when an Array is about to be allocated.

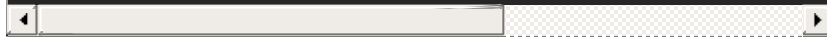
```
length the size of the array (long)  
filename the name of the file where the array  
lineno the line number in the file where the
```



ruby::hash-create(length, filename, lineno);

This probe is fired when a Hash is about to be allocated.

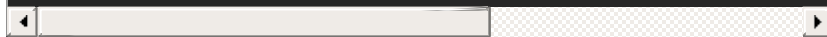
```
length the size of the hash (long)  
filename the name of the file where the hash  
lineno the line number in the file where the
```



ruby::string-create(length, filename, lineno);

This probe is fired when a String is about to be allocated.

```
length the size of the string (long)  
filename the name of the file where the string  
lineno the line number in the file where the
```

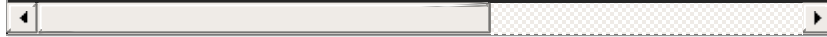


ruby::symbol-create(str, filename, lineno);

This probe is fired when a Symbol is about to be

allocated.

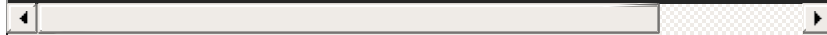
```
str the contents of the symbol (string)
filename the name of the file where the stri
lineno the line number in the file where the
```



ruby:::parse-begin(sourcefile, lineno);

Fired just before parsing and compiling a source file.

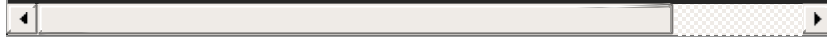
```
sourcefile the file being parsed (string)
lineno the line number where the source star
```



ruby:::parse-end(sourcefile, lineno);

Fired just after parsing and compiling a source file.

```
sourcefile the file being parsed (string)
lineno the line number where the source ende
```



ruby:::gc-mark-begin();

Fired at the beginning of a mark phase.

ruby:::gc-mark-end();

Fired at the end of a mark phase.

ruby:::gc-sweep-begin();

Fired at the beginning of a sweep phase.

ruby::gc-sweep-end();

Fired at the end of a sweep phase.

**ruby::method-cache-clear(class, sourcefile,
lineno);**

Fired when the method cache is cleared.

```
class is the classname being cleared, or "gl  
sourcefile the file being parsed (string)  
lineno the line number where the source ends
```

Generated by [RDoc 3.12.2](#).

Generated with the [Darkfish Rdoc Generator 3](#).

Pre-defined variables

\$!

The exception information message set by 'raise'.

\$@

Array of backtrace of the last exception thrown.

\$&

The string matched by the last successful match.

\$`

The string to the left of the last successful match.

\$'

The string to the right of the last successful match.

\$+

The highest group matched by the last successful match.

\$1

The Nth group of the last successful match. May be > 1.

\$~

The information about the last match in the current scope.

\$=

The flag for case insensitive, nil by default.

`$/`

The input record separator, newline by default.

`$\`

The output record separator for the print and IO#write. Default is nil.

`$,`

The output field separator for the print and Array#join.

`$;`

The default separator for String#split.

`$.`

The current input line number of the last file that was read.

`$<`

The virtual concatenation file of the files given on command line (or from \$stdin if no files were given).

`$>`

The default output for print, printf. \$stdout by default.

`$_`

The last input line of string by gets or readline.

`$0`

Contains the name of the script being executed. May

be assignable.

\$*

Command line arguments given for the script sans args.

\$\$

The process number of the Ruby running this script.

\$?

The status of the last executed child process. This value is thread-local.

\$:

Load path for scripts and binary modules by load or require.

\$“

The array contains the module names loaded by require.

\$DEBUG

The debug flag, which is set by the -d switch. Enabling debug output prints each exception raised to \$stderr (but not its backtrace). Setting this to a true value enables debug output as if -d were given on the command line. Setting this to a false value disables debug output.

\$LOADED_FEATURES

The alias to the \$“.

\$FILENAME

Current input file from \$<. Same as \$<.filename.

\$LOAD_PATH

The alias to the \$:

\$stderr

The current standard error output.

\$stdin

The current standard input.

\$stdout

The current standard output.

\$VERBOSE

The verbose flag, which is set by the -w or -v switch. Setting this to a true value enables warnings as if -w or -v were given on the command line. Setting this to nil disables warnings, including from Kernel#warn.

\$-0

The alias to \$/.

\$-a

True if option -a is set. Read-only variable.

\$-d

The alias of \$DEBUG. See \$DEBUG above for further discussion.

\$-F

The alias to \$:

\$-i

In in-place-edit mode, this variable holds the extension, otherwise nil.

\$-l

The alias to \$:

\$-l

True if option -l is set. Read-only variable.

\$-p

True if option -p is set. Read-only variable.

\$-v

An alias of \$VERBOSE. See \$VERBOSE above for further discussion.

\$-w

An alias of \$VERBOSE. See \$VERBOSE above for further discussion.

Pre-defined global constants

TRUE

The typical true value.

FALSE

The false itself.

NIL

The nil itself.

STDIN

The standard input. The default value for \$stdin.

STDOUT

The standard output. The default value for \$stdout.

STDERR

The standard error output. The default value for \$stderr.

ENV

The hash contains current environment variables.

ARGF

The alias to the \$<.

ARGV

The alias to the \$*.

DATA

The file object of the script, pointing just after `__END__`.

RUBY_VERSION

The ruby version string (VERSION was deprecated).

RUBY_RELEASE_DATE

The release date string.

RUBY_PLATFORM

The platform identifier.

Generated by [RDoc](#) 3.12.2.

Generated with the [Darkfish Rdoc Generator](#) 3.

Keywords

The following keywords are used by Ruby.

__ENCODING__

The script encoding of the current file. See Encoding.

__LINE__

The line number of this keyword in the current file.

__FILE__

The path to the current file.

BEGIN

Runs before any other code in the current file. See miscellaneous syntax

END

Runs after any other code in the current file. See miscellaneous syntax

alias

Creates an alias between two methods (and other things). See modules and classes syntax

and

Short-circuit Boolean and with lower precedence than `&&`

begin

Starts an exception handling block. See exceptions syntax

break

Leaves a block early. See control expressions syntax

case

Starts a case expression. See control expressions syntax

class

Creates or opens a class. See modules and classes syntax

def

Defines a method. See methods syntax

defined?

Returns a string describing its argument. See miscellaneous syntax

do

Starts a block.

else

The unhandled condition in case, if and unless expressions. See control expressions syntax

elsif

An alternate condition for an if expression. See control expressions syntax

end

The end of a syntax block. Used by classes, modules, methods, exception handling and control expressions.

ensure

Starts a section of code that is always run when an exception is raised. See exceptions syntax

false

Boolean false. See literals

for

A loop that is similar to using the `each` method. See control expressions syntax

if

Used for `if` and modifier `if` expressions. See control expressions syntax

in

Used to separate the iterable object and iterator variable in a `for` loop. See control expressions syntax

module

Creates or opens a module. See modules and classes syntax

next

Skips the rest of the block. See control expressions syntax

nil

A false value usually indicating “no value” or “unknown”. See literals

not

Inverts the following boolean expression. Has a lower precedence than !

or

Boolean or with lower precedence than ||

redo

Restarts execution in the current block. See control expressions syntax

rescue

Starts an exception section of code in a `begin` block. See exceptions syntax

retry

Retries an exception block. See exceptions syntax

return

Exits a method. See methods syntax

self

The object the current method is attached to. See methods syntax

super

Calls the current method in a superclass. See methods syntax

then

Indicates the end of conditional blocks in control structures. See control expressions syntax

true

Boolean true. See literals

undef

Prevents a class or module from responding to a method call. See modules and classes syntax

unless

Used for `unless` and modifier `unless` expressions. See control expressions syntax

until

Creates a loop that executes until the condition is true. See control expressions syntax

when

A condition in a `case` expression. See control expressions syntax

while

Creates a loop that executes while the condition is true. See control expressions syntax

yield

Starts execution of the block sent to the current method. See methods syntax

Generated by [RDoc](#) 3.12.2.
Generated with the [Darkfish Rdoc Generator](#) 3.

Maintainers

This page describes the current module, library, and extension maintainers of Ruby.

Module Maintainers

A module maintainer is responsible for a certain part of Ruby.

- The maintainer fixes bugs of the part. Particularly, they should fix security vulnerabilities as soon as possible.
- They handle issues related the module on the Redmine or ML.
- They may be discharged by the 3 months rule [ruby-core:25764].
- They have commit right to Ruby's repository to modify their part in the repository.
- They have “developer” role on the Redmine to modify issues.
- They have authority to decide the feature of their part. But they should always respect discussions on ruby-core/ruby-dev.

A submaintainer of a module is like a maintainer. But The submaintainer does not have authority to change/add a feature on his/her part. They need consensus on ruby-core/ruby-dev before changing/adding. Some of submaintainers have commit right, others don't.

Language core features including security

Yukihiro Matsumoto (matz)

Evaluator

Koichi Sasada (ko1)

Core classes

Yukihiro Matsumoto (matz)

Documentation

Zachary Scott (zzak)

Library Maintainers

lib/English.rb

unmaintained

lib/abbrev.rb

Akinori MUSHA (knu)

lib/base64.rb

- 1.8: *unmaintained*
- 1.9: Yusuke Endoh (mame)

lib/benchmark.rb

unmaintained

lib/cgi.rb, lib/cgi/*

Takeyuki Fujioka (xibbar)

lib/complex.rb

- 1.8: *unmaintained*
- 1.9: moved into core

lib/cmath.rb

- 1.8: 1.9 feature
- 1.9: *unmaintained*

lib/csv.rb

- 1.8: Hiroshi Nakamura (nahi)

- 1.9: James Edward Gray II (jeg2)

lib/date.rb, lib/date/*

Tadayoshi Funaba (tadf)

lib/drb.rb, lib/drb/*

Masatoshi SEKI (seki)

lib/debug.rb

unmaintained

lib/delegate.rb

unmaintained

lib/e2mmap.rb

Keiju ISHITSUKA (keiju)

lib/erb.rb

Masatoshi SEKI (seki)

lib/fileutils.rb

unmaintained

lib/find.rb

Kazuki Tsujimoto (ktsj)

lib/finalize.rb

- 1.8: *unmaintained*
- 1.9: *deprecated*

lib/forwardable.rb

Keiju ISHITSUKA (keiju)

lib/ftools.rb

- 1.8: *unmaintained*
- 1.9: *deprecated*

lib/generator.rb

- 1.8: Akinori MUSHA (knu)
- 1.9: moved into core

lib/getoptlong.rb

unmaintained

lib/getopts.rb

- 1.8: Akinori MUSHA (knu)
- 1.9: *deprecated*

lib/gserver.rb

James Edward Gray II (jeg2)

lib/ipaddr.rb

Akinori MUSHA (knu)

lib/irb.rb, lib/irb/*

Keiju ISHITSUKA (keiju)

lib/jcode.rb

- 1.8: *unmaintained*
- 1.9: *deprecated*

lib/logger.rb

Hiroshi Nakamura (nahi)

lib/mathn.rb

Keiju ISHITSUKA (keiju)

lib/matrix.rb

Marc-Andre Lafortune (marcandre)

lib/minitest/*

- 1.8: 1.9 feature
- 1.9: Ryan Davis (ryan)

lib/mkrf.rb

unmaintained

lib/monitor.rb

Shugo Maeda (shugo)

lib/mutex_m.rb

Keiju ISHITSUKA (keiju)

lib/net/ftp.rb

Shugo Maeda (shugo)

lib/net/imap.rb

Shugo Maeda (shugo)

lib/net/telnet.rb

unmaintained

lib/net/http.rb, lib/net/https

NARUSE, Yui (naruse)

lib/net/pop.rb

unmaintained

lib/net/smtp.rb

unmaintained

lib/observer.rb

unmaintained

lib/open-uri.rb

Tanaka Akira (akr)

lib/open3.rb

unmaintained

lib/optparse.rb, lib/optparse/*

Nobuyuki Nakada (nobu)

lib/ostruct.rb

Marc-Andre Lafortune (marcandre)

lib/parsearg.rb

- 1.8: *unmaintained*
- 1.9: *deprecated*

lib/parsedate.rb

- 1.8: Tadayoshi Funaba (tadf)
- 1.9: *deprecated*

lib/pathname.rb

Tanaka Akira (akr)

lib/ping.rb

- 1.8: *unmaintained*
- 1.9: *deprecated*

lib/pp.rb

Tanaka Akira (akr)

lib/prettyprint.rb

Tanaka Akira (akr)

lib/prime.rb

Yuki Sonoda (yugui)

lib/profile.rb

unmaintained

lib/profiler.rb

unmaintained

lib/pstore.rb

unmaintained

lib/racc/*

Aaron Patterson (tenderlove)

lib/rake/*

Eric Hodel (drbrain)

lib/rational.rb

- 1.8: *unmaintained*
- 1.9: moved into core

lib/rdoc/*

Eric Hodel (drbrain)

lib/readbytes.rb

- 1.8: *unmaintained*
- 1.9: *deprecated*

lib/resolv-replace.rb

Tanaka Akira (akr)

lib/resolv.rb

Tanaka Akira (akr)

lib/rexml/*

Kouhei Sutou (kou)

lib/rinda/*

Masatoshi SEKI (seki)

lib/rss/*

Kouhei Sutou (kou)

lib/rubygems.rb, lib/ubygems.rb, lib/rubygems/*

- 1.8: 1.9 feature
- 1.9: Eric Hodel (drbrain)

lib/rubyunit.rb, lib/runit/*

- 1.8: *unmaintained*
- 1.9: *deprecated*

lib/scanf.rb

David A. Black (dblack)

lib/set.rb

Akinori MUSHA (knu)

lib/securerandom.rb

Tanaka Akira (akr)

lib/shell.rb, lib/shell/*

Keiju ISHITSUKA (keiju)

lib/shellwords.rb

Akinori MUSHA (knu)

lib/singleton.rb

Yukihiro Matsumoto (matz)

lib/{soap|wsdl|xsd}/*

- 1.8: Hiroshi Nakamura (nahi)
- 1.9: *deprecated*

lib/sync.rb

Keiju ISHITSUKA (keiju)

lib/tempfile.rb

unmaintained

lib/test/*

Shota Fukumori (sorah)

lib/tmpdir.rb

unmaintained

lib/thread.rb

unmaintained

lib/thwait.rb

Keiju ISHITSUKA (keiju)

lib/time.rb

Tanaka Akira (akr)

lib/timeout.rb

Yukihiro Matsumoto (matz)

lib/tracer.rb

Keiju ISHITSUKA (keiju)

lib/tsort.rb

Tanaka Akira (akr)

lib/un.rb

WATANABE Hirofumi (eban)

lib/uri.rb, lib/uri/*

YAMADA, Akira (akira)

lib/weakref.rb

unmaintained

lib/webrick.rb, lib/webrick/*

Hiroshi Nakamura (nahi)

lib/xmlrpc/*

Kouhei Sutou (kou)

lib/yaml.rb, lib/yaml/*

Aaron Patterson (tenderlove)

Extension Maintainers

ext/Win32API

- 1.8: *unmaintained*
- 1.9: merged into dl

ext/bigdecimal

Kenta Murata (mrkn)

ext/continuation

- 1.8: 1.9 feature
- 1.9: Koichi Sasada (ko1)

ext/coverage

Yusuke Endoh (mame)

ext/dbm

unmaintained

ext/digest, ext/digest/*

Akinori MUSAHA (knu)

ext/dl

Aaron Patterson (tenderlove)

ext/dl/win32

NAKAMURA Usaku (usa)

ext/enumerator

- 1.8: Akinori MUSHA (knu)
- 1.9: moved into core

ext/etc

unmaintained

ext/fcntl

unmaintained

ext/fiber

- 1.8: 1.9 feature
- 1.9: Koichi Sasada (ko1)

ext/fiddle

Aaron Patterson (tenderlove)

ext/gdbm

Yukihiro Matsumoto (matz)

ext/iconv

Nobuyuki Nakada (nobu)

ext/io/wait

Nobuyuki Nakada (nobu)

ext/json

NARUSE, Yui (naruse)

ext/mathn/complex

- 1.8: 1.9 feature
- 1.9: Keiju ISHITSUKA (keiju)

ext/mathn/rational

- 1.8: 1.9 feature
- 1.9: Keiju ISHITSUKA (keiju)

ext/nkf

NARUSE, Yui (narse)

ext/objspace

unmaintained

ext/openssl

Martin Boßlet (emboss)

ext/psych

Aaron Patterson (tenderlove)

ext/pty

unmaintained

ext/racc

Aaron Patterson (tenderlove)

ext/readline

TAKAO Kouji (kouji)

ext/ripper

unmaintained

ext/sdbm

Yukihiro Matsumoto (matz)

ext/socket

- Tanaka Akira (akr)
- API change needs matz's approval

ext/stringio

Nobuyuki Nakada (nobu)

ext/strscan

unmaintained

ext/syck

unmaintained

ext/syslog

Akinori MUSHA (knu)

ext/thread

- 1.8: *unmaintained*
- 1.9: 1.8 feature

ext/tcltklib

deprecated

ext/tk

Hidetoshi NAGAI (nagai)

ext/win32ole

Masaki Suketa (suke)

ext/zlib

unmaintained

Marshal Format

The Marshal format is used to serialize ruby objects. The format can store arbitrary objects through three user-defined extension mechanisms.

For documentation on using Marshal to serialize and deserialize objects, see the Marshal module.

This document calls a serialized set of objects a stream. The Ruby implementation can load a set of objects from a String, an IO or an object that implements a `getc` method.

Stream Format

The first two bytes of the stream contain the major and minor version, each as a single byte encoding a digit. The version implemented in Ruby is 4.8 (stored as “x04x08”) and is supported by ruby 1.8.0 and newer.

Different major versions of the Marshal format are not compatible and cannot be understood by other major versions. Lesser minor versions of the format can be understood by newer minor versions. Format 4.7 can be loaded by a 4.8 implementation but format 4.8 cannot be loaded by a 4.7 implementation.

Following the version bytes is a stream describing the serialized object. The stream contains nested objects (the same as a Ruby object) but objects in the stream do not necessarily have a direct mapping to the Ruby object model.

Each object in the stream is described by a byte indicating its type followed by one or more bytes describing the object. When “object” is mentioned below it means any of the types below that defines a Ruby object.

true, false, nil

These objects are each one byte long. “T” is represents true, “F” represents false and “0” represents nil.

Fixnum and long

“i” represents a signed 32 bit value using a packed format. One through five bytes follows the type. The value loaded will always be a Fixnum. On 32 bit platforms

(where the precision of a Fixnum is less than 32 bits)
loading large values will cause overflow on CRuby.

The fixnum type is used to represent both ruby Fixnum objects and the sizes of marshaled arrays, hashes, instance variables and other types. In the following sections “long” will mean the format described below, which supports full 32 bit precision.

The first byte has the following special values:

“x00”

The value of the integer is 0. No bytes follow.

“x01”

The total size of the integer is two bytes. The following byte is a positive integer in the range of 0 through 255. Only values between 123 and 255 should be represented this way to save bytes.

“xff”

The total size of the integer is two bytes. The following byte is a negative integer in the range of -1 through -256.

“x02”

The total size of the integer is three bytes. The following two bytes are a positive little-endian integer.

“xfe”

The total size of the integer is three bytes. The following two bytes are a negative little-endian integer.

“x03”

The total size of the integer is four bytes. The following three bytes are a positive little-endian integer.

“xfd”

The total size of the integer is two bytes. The following three bytes are a negative little-endian integer.

“x04”

The total size of the integer is five bytes. The following four bytes are a positive little-endian integer. For compatibility with 32 bit ruby, only Fixnums less than 1073741824 should be represented this way. For sizes of stream objects full precision may be used.

“xfc”

The total size of the integer is two bytes. The following four bytes are a negative little-endian integer. For compatibility with 32 bit ruby, only Fixnums greater than -10737341824 should be represented this way. For sizes of stream objects full precision may be used.

Otherwise the first byte is a sign-extended eight-bit value with an offset. If the value is positive the value is determined by subtracting 5 from the value. If the value is negative the value is determined by adding 5 to the value.

There are multiple representations for many values. CRuby always outputs the shortest representation possible.

Symbols and Byte Sequence

“.” represents a real symbol. A real symbol contains the data needed to define the symbol for the rest of the stream as future occurrences in the stream will instead be

references (a symbol link) to this one. The reference is a zero-indexed 32 bit value (so the first occurrence of :hello is 0).

Following the type byte is byte sequence which consists of a long indicating the number of bytes in the sequence followed by that many bytes of data. Byte sequences have no encoding.

For example, the following stream contains the Symbol :hello:

```
"\x04\x08:\x0ahello"
```

“;” represents a Symbol link which references a previously defined Symbol. Following the type byte is a long containing the index in the lookup table for the linked (referenced) Symbol.

For example, the following stream contains [:hello, :hello]:

```
"\x04\b[\a:\nhello;\x00"
```

When a “symbol” is referenced below it may be either a real symbol or a symbol link.

Object References

Separate from but similar to symbol references, the stream contains only one copy of each object (as determined by object_id) for all objects except true, false, nil, Fixnums and Symbols (which are stored separately as described above) a one-indexed 32 bit value will be stored and reused when the object is encountered again. (The first object has an index of 1).

“@” represents an object link. Following the type byte is a long giving the index of the object.

For example, the following stream contains an Array of the object "hello" twice:

```
"\004\b[\a\"hello@\006"
```

Instance Variables

“l” indicates that instance variables follow the next object. An object follows the type byte. Following the object is a length indicating the number of instance variables for the object. Following the length is a set of name-value pairs. The names are symbols while the values are objects. The symbols must be instance variable names (:@name).

An Object (“o” type, described below) uses the same format for its instance variables as described here.

For a String and Regexp (described below) a special instance variable :E is used to indicate the Encoding.

Extended

“e” indicates that the next object is extended by a module. An object follows the type byte. Following the object is a symbol that contains the name of the module the object is extended by.

Array

“[” represents an Array. Following the type byte is a long indicating the number of objects in the array. The given number of objects follow the length.

Bignum

“l” represents a Bignum which is composed of three parts:

sign

A single byte containing “+” for a positive value or “-” for a negative value.

length

A long indicating the number of bytes of Bignum data follows, divided by two. Multiply the length by two to determine the number of bytes of data that follow.

data

Bytes of Bignum data representing the number.

The following ruby code will reconstruct the Bignum value from an array of bytes:

```
result = 0

bytes.each_with_index do |byte, exp|
  result += (byte * 2 ** (exp * 8))
end
```

Class and Module

“c” represents a Class object, “m” represents a Module and “M” represents either a class or module (this is an old-style for compatibility). No class or module content is included, this type is only a reference. Following the type byte is a byte sequence which is used to look up an existing class or module, respectively.

Instance variables are not allowed on a class or module.

If no class or module exists an exception should be raised.

For “c” and “m” types, the loaded object must be a class or module, respectively.

Data

“d” represents a Data object. (Data objects are wrapped pointers from ruby extensions.) Following the type byte is a symbol indicating the class for the Data object and an object that contains the state of the Data object.

To dump a Data object Ruby calls `_dump_data`. To load a Data object Ruby calls `_load_data` with the state of the object on a newly allocated instance.

Float

“f” represents a Float object. Following the type byte is a byte sequence containing the float value. The following values are special:

“inf”

Positive infinity

“-inf”

Negative infinity

“nan”

Not a Number

Otherwise the byte sequence contains a C double (loadable by `strtod(3)`). Older minor versions of Marshal also stored extra mantissa bits to ensure portability

across platforms but 4.8 does not include these. See

ruby-talk:69518

for some explanation.

Hash and Hash with Default Value

“{” represents a Hash object while “}” represents a Hash with a default value set (`Hash.new 0`). Following the type byte is a long indicating the number of key-value pairs in the Hash, the size. Double the given number of objects follow the size.

For a Hash with a default value, the default value follows all the pairs.

Module and Old Module

Object

“o” represents an object that doesn't have any other special form (such as a user-defined or built-in format). Following the type byte is a symbol containing the class name of the object. Following the class name is a long indicating the number of instance variable names and values for the object. Double the given number of pairs of objects follow the size.

The keys in the pairs must be symbols containing instance variable names.

Regular Expression

“/” represents a regular expression. Following the type

byte is a byte sequence containing the regular expression source. Following the type byte is a byte containing the regular expression options (case-insensitive, etc.) as a signed 8-bit value.

Regular expressions can have an encoding attached through instance variables (see above). If no encoding is attached escapes for the following regexp specials not present in ruby 1.8 must be removed: g-m, o-q, u, y, E, F, H-L, N-V, X, Y.

String

"" represents a String. Following the type byte is a byte sequence containing the string content. When dumped from ruby 1.9 an encoding instance variable (:E see above) should be included unless the encoding is binary.

Struct

"S" represents a Struct. Following the type byte is a symbol containing the name of the struct. Following the name is a long indicating the number of members in the struct. Double the number of objects follow the member count. Each member is a pair containing the member's symbol and an object for the value of that member.

If the struct name does not match a Struct subclass in the running ruby an exception should be raised.

If there is a mismatch between the struct in the currently running ruby and the member count in the marshaled struct an exception should be raised.

User Class

“C” represents a subclass of a String, Regexp, Array or Hash. Following the type byte is a symbol containing the name of the subclass. Following the name is the wrapped object.

User Defined

“u” represents an object with a user-defined serialization format using the `_dump` instance method and `_load` class method. Following the type byte is a symbol containing the class name. Following the class name is a byte sequence containing the user-defined representation of the object.

The class method `_load` is called on the class with a string created from the byte-sequence.

User Marshal

“U” represents an object with a user-defined serialization format using the `marshal_dump` and `marshal_load` instance methods. Following the type byte is a symbol containing the class name. Following the class name is an object containing the data.

Upon loading a new instance must be allocated and `marshal_load` must be called on the instance with the data.

Regular expressions (*regexps*) are patterns which describe the contents of a string. They're used for testing whether a string contains a given pattern, or extracting the portions that match. They are created with the `/pat/` and `%r{pat}` literals or the `Regexp.new` constructor.

A regexp is usually delimited with forward slashes (`/`). For example:

```
/hay/ =~ 'haystack'    #=> 0  
/y/.match('haystack') #=> #<MatchData "y">
```

If a string contains the pattern it is said to *match*. A literal string matches itself.

Here 'haystack' does not contain the pattern 'needle', so it doesn't match:

```
/needle/.match('haystack') #=> nil
```

Here 'haystack' contains the pattern 'hay', so it matches:

```
/hay/.match('haystack')    #=> #<MatchData "hay">
```

Specifically, `/st/` requires that the string contains the letter *s* followed by the letter *t*, so it matches *haystack*, also.

`=~` and `Regexp#match`

Pattern matching may be achieved by using `=~` operator or `Regexp#match` method.

`=~` operator

`=~` is Ruby's basic pattern-matching operator. When one operand is a regular expression and the other is a string then the regular expression is used as a pattern to match against the string. (This operator is equivalently defined by `Regexp` and `String` so the order of `String` and `Regexp` do not matter. Other classes may have different implementations of `=~`.) If a match is found, the operator returns index of first match in string, otherwise it returns `nil`.

```
/hay/ =~ 'haystack'    #=> 0  
'haystack' =~ /hay/   #=> 0  
/a/   =~ 'haystack'   #=> 1  
/u/   =~ 'haystack'   #=> nil
```

Using `=~` operator with a `String` and `Regexp` the `$~` global variable is set after a successful match. `$~` holds a `MatchData` object. `Regexp.last_match` is equivalent to `$~`.

`Regexp#match` method

The `match` method returns a `MatchData` object:

```
/st/.match('haystack') #=> #<MatchData "st">
```

Metacharacters and Escapes

The following are *metacharacters* (,), [,], {, }, ., ?, +, *. They have a specific meaning when appearing in a pattern. To match them literally they must be backslash-escaped. To match a backslash literally backslash-escape that: `<tt>\\</tt>`.

```
/1 \+ 2 = 3\?/.match('Does 1 + 2 = 3?') #=> #<MatchData
```

Patterns behave like double-quoted strings so can contain the same backslash escapes.

```
/\s\u{6771 4eac 90fd} /.match("Go to 東京都")  
#=> #<MatchData " 東京都">
```

Arbitrary Ruby expressions can be embedded into patterns with the `#{...}` construct.

```
place = "東京都"  
/#{place} /.match("Go to 東京都")  
#=> #<MatchData "東京都">
```

Character Classes

A *character class* is delimited with square brackets ([,]) and lists characters that may appear at that point in the match. `/[ab]/` means *a* or *b*, as opposed to `/ab/` which means *a* followed by *b*.

```
/W[aeiou]rd/.match("word") #=> #<MatchData "word">
```

Within a character class the hyphen (-) is a metacharacter denoting an inclusive range of characters. `[abcd]` is equivalent to `[a-d]`. A range can be followed by another range, so `[abcdwxyz]` is equivalent to `[a-dw-z]`. The order in which ranges or individual characters appear inside a character class is irrelevant.

```
/[0-9a-f]/.match('9f') #=> #<MatchData "9">  
/[9f]/.match('9f') #=> #<MatchData "9">
```

If the first character of a character class is a caret (^) the class is inverted: it matches any character *except* those named.

```
/[^a-eg-z]/.match('f') #=> #<MatchData "f">
```

A character class may contain another character class. By itself this isn't useful because `[a-z[0-9]]` describes the same set as `[a-z0-9]`. However, character classes also support the `&&` operator which performs set intersection on its arguments. The two can be combined as follows:

```
/[a-w&&[^c-g]z]/ # ([a-w] AND ([^c-g] OR z))
```

This is equivalent to:

```
/[abh-w]/
```

The following metacharacters also behave like character classes:

- `/./` - Any character except a newline.
- `./m` - Any character (the `m` modifier enables multiline mode)
- `/\w/` - A word character (`[a-zA-Z0-9_]`)
- `/\W/` - A non-word character (`[^a-zA-Z0-9_]`). Please take a look at [Bug #4044](#) if using `/\w/` with the `/i` modifier.
- `/\d/` - A digit character (`[0-9]`)
- `/\D/` - A non-digit character (`[^0-9]`)
- `/\h/` - A hexdigit character (`[0-9a-fA-F]`)
- `/\H/` - A non-hexdigit character (`[^0-9a-fA-F]`)
- `/\s/` - A whitespace character: `/[\t\r\n\f]/`
- `/\S/` - A non-whitespace character: `/[^ \t\r\n\f]/`

POSIX *bracket expressions* are also similar to character classes. They provide a portable alternative to the above, with the added benefit that they encompass non-ASCII characters. For instance, `/\d/` matches only the ASCII decimal digits (0-9); whereas `/[:digit:]/` matches any character in the Unicode *Nd* category.

- `/[:alnum:]/` - Alphabetic and numeric character
- `/[:alpha:]/` - Alphabetic character
- `/[:blank:]/` - Space or tab
- `/[:cntrl:]/` - Control character
- `/[:digit:]/` - Digit
- `/[:graph:]/` - Non-blank character (excludes spaces,

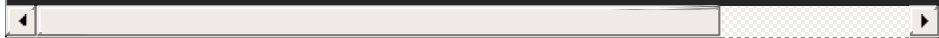
control characters, and similar)

- `/[[:lower:]]/` - Lowercase alphabetical character
- `/[[:print:]]/` - Like `[[:graph:]]`, but includes the space character
- `/[[:punct:]]/` - Punctuation character
- `/[[:space:]]/` - Whitespace character (`[[:blank:]]`, newline, carriage return, etc.)
- `/[[:upper:]]/` - Uppercase alphabetical
- `/[[:xdigit:]]/` - Digit allowed in a hexadecimal number (i.e., 0-9a-fA-F)

Ruby also supports the following non-POSIX character classes:

- `/[[:word:]]/` - A character in one of the following Unicode general categories *Letter*, *Mark*, *Number*, *Connector_Punctuation*
- `/[[:ascii:]]/` - A character in the ASCII character set

```
# U+06F2 is "EXTENDED ARABIC-INDIC DIGIT TWO"  
/[[:digit:]]/.match("\u06F2")    #=> #<MatchData "  
/[[:upper:]][[:lower:]]/.match("Hello") #=> #<Matc  
/[[:xdigit:]][[:xdigit:]]/.match("A6") #=> #<Matc
```



Repetition

The constructs described so far match a single character. They can be followed by a repetition metacharacter to specify how many times they need to occur. Such metacharacters are called *quantifiers*.

- * - Zero or more times
- + - One or more times
- ? - Zero or one times (optional)
- {*n*} - Exactly *n* times
- {*n*,} - *n* or more times
- {, *m*} - *m* or less times
- {*n*, *m*} - At least *n* and at most *m* times

At least one uppercase character ('H'), at least one lowercase character ('e'), two 'l' characters, then one 'o':

```
"Hello".match(/[[[:upper:]]+[[[:lower:]]+1{2}o/ ) #=> #<
```

Repetition is *greedy* by default: as many occurrences as possible are matched while still allowing the overall match to succeed. By contrast, *lazy* matching makes the minimal amount of matches necessary for overall success. A greedy metacharacter can be made lazy by following it with ?.

Both patterns below match the string. The first uses a greedy quantifier so '.+' matches '<a>'; the second uses a lazy quantifier so '.+?' matches '<a>':

```
/<.+>/ .match("<a><b>") #=> #<MatchData "<a><b>">  
/<.+?>/ .match("<a><b>") #=> #<MatchData "<a>">
```

A quantifier followed by + matches *possessively*: once it has matched it does not backtrack. They behave like greedy quantifiers, but having matched they refuse to “give up” their match even if this jeopardises the overall match.

Capturing

Parentheses can be used for *capturing*. The text enclosed by the n^{th} group of parentheses can be subsequently referred to with n . Within a pattern use the *backreference* $\backslash n$; outside of the pattern use `MatchData[n]`.

'at' is captured by the first group of parentheses, then referred to later with $\backslash 1$:

```
[csh](..) [csh]\1 in/.match("The cat sat in the hat")  
#=> #<MatchData "cat sat in" 1:"at">
```

`Regexp#match` returns a `MatchData` object which makes the captured text available with its `#[]` method:

```
[csh](..) [csh]\1 in/.match("The cat sat in the hat")
```

Capture groups can be referred to by name when defined with the $(?<name>)$ or $(?'name')$ constructs.

```
/\$(?<dollars>\d+)\.(?<cents>\d+)/.match("$3.67")  
=> #<MatchData "$3.67" dollars:"3" cents:"67">  
/\$(?<dollars>\d+)\.(?<cents>\d+)/.match("$3.67")[:do
```

Named groups can be backreferenced with $\backslash k<name>$, where *name* is the group name.

```
/(?<vowel>[aeiou]).\k<vowel>.\k<vowel>/.match('ototom')  
#=> #<MatchData "ototo" vowel:"o">
```

Note: A regexp can't use named backreferences and

numbered backreferences simultaneously.

When named capture groups are used with a literal regexp on the left-hand side of an expression and the =~ operator, the captured text is also assigned to local variables with corresponding names.

```
/\$(?<dollars>\d+)\.(?<cents>\d+)/ =~ "$3.67" #=> 0  
dollars #=> "3"
```

Grouping

Parentheses also *group* the terms they enclose, allowing them to be quantified as one *atomic* whole.

The pattern below matches a vowel followed by 2 word characters:

```
/[aeiou]\w{2}/.match("Caenorhabditis elegans") #=> #<
```

Whereas the following pattern matches a vowel followed by a word character, twice, i.e. `[aeiou]\w[aeiou]\w`: 'enor'.

```
/([aeiou]\w){2}/.match("Caenorhabditis elegans")  
#=> #<MatchData "enor" 1:"or">
```

The `(?:...)` construct provides grouping without capturing. That is, it combines the terms it contains into an atomic whole without creating a backreference. This benefits performance at the slight expense of readability.

The first group of parentheses captures 'n' and the second 'ti'. The second group is referred to later with the backreference `\2`:

```
/I(n)ves(ti)ga\2ons/.match("Investigations")  
#=> #<MatchData "Investigations" 1:"n" 2:"ti">
```

The first group of parentheses is now made non-capturing with `'?:'`, so it still matches 'n', but doesn't create the backreference. Thus, the backreference `\1` now refers to 'ti'.

```
/I(?:n)ves(ti)ga\1ons/.match("Investigations")  
#=> #<MatchData "Investigations" 1:"ti">
```

Atomic Grouping

Grouping can be made *atomic* with $(?>pat)$. This causes the subexpression pat to be matched independently of the rest of the expression such that what it matches becomes fixed for the remainder of the match, unless the entire subexpression must be abandoned and subsequently revisited. In this way pat is treated as a non-divisible whole. Atomic grouping is typically used to optimise patterns so as to prevent the regular expression engine from backtracking needlessly.

The `"` in the pattern below matches the first character of the string, then `.*` matches `Quote`. This causes the overall match to fail, so the text matched by `.*` is backtracked by one position, which leaves the final character of the string available to match `"`

```
/".*"/.match('"Quote"') #=> #<MatchData "\"Quote\"
```

If `.*` is grouped atomically, it refuses to backtrack `Quote`, even though this means that the overall match fails

```
/"(?>.*)"/.match('"Quote"') #=> nil
```

Subexpression Calls

The `\g<name>` syntax matches the previous subexpression named *name*, which can be a group name or number, again. This differs from backreferences in that it re-executes the group rather than simply trying to re-match the same text.

This pattern matches a (character and assigns it to the paren group, tries to call that the paren sub-expression again but fails, then matches a literal):

```
/\A(?<paren>\(\g<paren>*\))*\z/ =~ '()'
/\A(?<paren>\(\g<paren>*\))*\z/ =~ '(())' #=> 0
# ^1
#   ^2
#     ^3
#       ^4
#         ^5
#           ^6
#             ^7
#               ^8
#                 ^9
#                   ^10
```

1. Matches at the beginning of the string, i.e. before the first character.
2. Enters a named capture group called paren
3. Matches a literal (, the first character in the string
4. Calls the paren group again, i.e. recurses back to the second step
5. Re-enters the paren group
6. Matches a literal (, the second character in the string
7. Try to call paren a third time, but fail because doing so would prevent an overall successful match

8. Match a literal `)`, the third character in the string. Marks the end of the second recursive call
9. Match a literal `)`, the fourth character in the string
10. Match the end of the string

Alternation

The vertical bar metacharacter (|) combines two expressions into a single one that matches either of the expressions. Each expression is an *alternative*.

```
/\w(and|or)\w/.match("Feliformia") #=> #<MatchData "f"
/\w(and|or)\w/.match("furandi")    #=> #<MatchData "r"
/\w(and|or)\w/.match("dissemblance") #=> nil
```

Character Properties

The `\p{}` construct matches characters with the named property, much like POSIX bracket classes.

- `/\p{Alnum}/` - Alphanumeric character
- `/\p{Alpha}/` - Alphabetic character
- `/\p{Blank}/` - Space or tab
- `/\p{Cntrl}/` - Control character
- `/\p{Digit}/` - Digit
- `/\p{Graph}/` - Non-blank character (excludes spaces, control characters, and similar)
- `/\p{Lower}/` - Lowercase alphabetical character
- `/\p{Print}/` - Like `\p{Graph}`, but includes the space character
- `/\p{Punct}/` - Punctuation character
- `/\p{Space}/` - Whitespace character (`[:blank:]`, newline, carriage return, etc.)
- `/\p{Upper}/` - Uppercase alphabetical
- `/\p{XDigit}/` - Digit allowed in a hexadecimal number (i.e., 0-9a-fA-F)
- `/\p{Word}/` - A member of one of the following Unicode general category *Letter*, *Mark*, *Number*, *Connector_Punctuation*
- `/\p{ASCII}/` - A character in the ASCII character set
- `/\p{Any}/` - Any Unicode character (including unassigned characters)
- `/\p{Assigned}/` - An assigned character

A Unicode character's *General Category* value can also

be matched with $\backslash p\{Ab\}$ where Ab is the category's abbreviation as described below:

- $\backslash p\{L\}$ / - 'Letter'
- $\backslash p\{Ll\}$ / - 'Letter: Lowercase'
- $\backslash p\{Lm\}$ / - 'Letter: Mark'
- $\backslash p\{Lo\}$ / - 'Letter: Other'
- $\backslash p\{Lt\}$ / - 'Letter: Titlecase'
- $\backslash p\{Lu\}$ / - 'Letter: Uppercase'
- $\backslash p\{Lo\}$ / - 'Letter: Other'
- $\backslash p\{M\}$ / - 'Mark'
- $\backslash p\{Mn\}$ / - 'Mark: Nonspacing'
- $\backslash p\{Mc\}$ / - 'Mark: Spacing Combining'
- $\backslash p\{Me\}$ / - 'Mark: Enclosing'
- $\backslash p\{N\}$ / - 'Number'
- $\backslash p\{Nd\}$ / - 'Number: Decimal Digit'
- $\backslash p\{Nl\}$ / - 'Number: Letter'
- $\backslash p\{No\}$ / - 'Number: Other'
- $\backslash p\{P\}$ / - 'Punctuation'
- $\backslash p\{Pc\}$ / - 'Punctuation: Connector'
- $\backslash p\{Pd\}$ / - 'Punctuation: Dash'
- $\backslash p\{Ps\}$ / - 'Punctuation: Open'
- $\backslash p\{Pe\}$ / - 'Punctuation: Close'
- $\backslash p\{Pi\}$ / - 'Punctuation: Initial Quote'
- $\backslash p\{Pf\}$ / - 'Punctuation: Final Quote'
- $\backslash p\{Po\}$ / - 'Punctuation: Other'
- $\backslash p\{S\}$ / - 'Symbol'
- $\backslash p\{Sm\}$ / - 'Symbol: Math'
- $\backslash p\{Sc\}$ / - 'Symbol: Currency'

- `\p{Sc}`/ - 'Symbol: Currency'
- `\p{Sk}`/ - 'Symbol: Modifier'
- `\p{So}`/ - 'Symbol: Other'
- `\p{Z}`/ - 'Separator'
- `\p{Zs}`/ - 'Separator: Space'
- `\p{Zl}`/ - 'Separator: Line'
- `\p{Zp}`/ - 'Separator: Paragraph'
- `\p{C}`/ - 'Other'
- `\p{Cc}`/ - 'Other: Control'
- `\p{Cf}`/ - 'Other: Format'
- `\p{Cn}`/ - 'Other: Not Assigned'
- `\p{Co}`/ - 'Other: Private Use'
- `\p{Cs}`/ - 'Other: Surrogate'

Lastly, `\p{}` matches a character's Unicode *script*. The following scripts are supported: *Arabic, Armenian, Balinese, Bengali, Bopomofo, Braille, Buginese, Buhid, Canadian_Aboriginal, Carian, Cham, Cherokee, Common, Coptic, Cuneiform, Cypriot, Cyrillic, Deseret, Devanagari, Ethiopic, Georgian, Glagolitic, Gothic, Greek, Gujarati, Gurmukhi, Han, Hangul, Hanunoo, Hebrew, Hiragana, Inherited, Kannada, Katakana, Kayah_Li, Kharoshthi, Khmer, Lao, Latin, Lepcha, Limbu, Linear_B, Lycian, Lydian, Malayalam, Mongolian, Myanmar, New_Tai_Lue, Nko, Ogham, Ol_Chiki, Old_Italic, Old_Persian, Oriya, Osmanya, Phags_Pa, Phoenician, Rejang, Runic, Saurashtra, Shavian, Sinhala, Sundanese, Syloti_Nagri, Syriac, Tagalog, Tagbanwa, Tai_Le, Tamil, Telugu, Thaana, Thai, Tibetan, Tifinagh, Ugaritic, Vai, and Yi.*

Unicode codepoint U+06E9 is named “ARABIC PLACE OF SAJDAH” and belongs to the Arabic script:

```
/\p{Arabic}/.match("\u06E9") #=> #<MatchData "\u06E9">
```

All character properties can be inverted by prefixing their name with a caret (^).

Letter 'A' is not in the Unicode LI (Letter; Lowercase) category, so this match succeeds:

```
/\p{^LI}/.match("A") #=> #<MatchData "A">
```

Anchors

Anchors are metacharacter that match the zero-width positions between characters, *anchoring* the match to a specific position.

- `^` - Matches beginning of line
- `$` - Matches end of line
- `\A` - Matches beginning of string.
- `\Z` - Matches end of string. If string ends with a newline, it matches just before newline
- `\z` - Matches end of string
- `\G` - Matches point where last match finished
- `\b` - Matches word boundaries when outside brackets; backspace (0x08) when inside brackets
- `\B` - Matches non-word boundaries
- `(?=pat)` - *Positive lookahead* assertion: ensures that the following characters match *pat*, but doesn't include those characters in the matched text
- `(?!pat)` - *Negative lookahead* assertion: ensures that the following characters do not match *pat*, but doesn't include those characters in the matched text
- `(?<=pat)` - *Positive lookbehind* assertion: ensures that the preceding characters match *pat*, but doesn't include those characters in the matched text
- `(?<!pat)` - *Negative lookbehind* assertion: ensures that the preceding characters do not match *pat*, but doesn't include those characters in the matched text

If a pattern isn't anchored it can begin at any point in the

string:

```
/real/.match("surrealist") #=> #<MatchData "real">
```

Anchoring the pattern to the beginning of the string forces the match to start there. 'real' doesn't occur at the beginning of the string, so now the match fails:

```
/^real/.match("surrealist") #=> nil
```

The match below fails because although 'Demand' contains 'and', the pattern does not occur at a word boundary.

```
/\band/.match("Demand")
```

Whereas in the following example 'and' has been anchored to a non-word boundary so instead of matching the first 'and' it matches from the fourth letter of 'demand' instead:

```
/\Band.+/.match("Supply and demand curve") #=> #<Matc
```

The pattern below uses positive lookahead and positive lookbehind to match text appearing in tags without including the tags in the match:

```
/(?<=<b>)\w+(?=<\b>)/.match("Fortune favours the <b>  
#=> #<MatchData "bold">
```

Options

The end delimiter for a regexp can be followed by one or more single-letter options which control how the pattern can match.

- `/pat/i` - Ignore case
- `/pat/m` - Treat a newline as a character matched by `.`
- `/pat/x` - Ignore whitespace and comments in the pattern
- `/pat/o` - Perform `#{} interpolation` only once

`i`, `m`, and `x` can also be applied on the subexpression level with the `(?on-off)` construct, which enables options *on*, and disables options *off* for the expression enclosed by the parentheses.

```
/a(?i:b)c/.match('aBc') #=> #<MatchData "aBc">  
/a(?i:b)c/.match('abc') #=> #<MatchData "abc">
```

Options may also be used with `Regexp.new`:

```
Regexp.new("abc", Regexp::IGNORECASE)  
Regexp.new("abc", Regexp::MULTILINE)  
Regexp.new("abc # Comment", Regexp::EXTENDED)  
Regexp.new("abc", Regexp::IGNORECASE | Regexp::MULTIL
```

Free-Spacing Mode and Comments

As mentioned above, the `x` option enables *free-spacing* mode. Literal white space inside the pattern is ignored, and the octothorpe (`#`) character introduces a comment until the end of the line. This allows the components of the pattern to be organized in a potentially more readable fashion.

A contrived pattern to match a number with optional decimal places:

```
float_pat = /\A
    [[:digit:]]+ # 1 or more digits before the decima
    (\.         # Decimal point
    [[:digit:]]+ # 1 or more digits after the dec
    )? # The decimal point and following digits are o
    \Z/
float_pat.match('3.14') #=> #<MatchData "3.14" 1:".14
```

There are a number of strategies for matching whitespace:

- Use a pattern such as `\s` or `\p{Space}`.
- Use escaped whitespace such as `\` , i.e. a space preceded by a backslash.
- Use a character class such as `[]`.

Comments can be included in a non-`x` pattern with the `(? #comment)` construct, where *comment* is arbitrary text ignored by the regexp engine.

Encoding

Regular expressions are assumed to use the source encoding. This can be overridden with one of the following modifiers.

- `/pat/u` - UTF-8
- `/pat/e` - EUC-JP
- `/pat/s` - Windows-31J
- `/pat/n` - ASCII-8BIT

A regexp can be matched against a string when they either share an encoding, or the regexp's encoding is *US-ASCII* and the string's encoding is ASCII-compatible.

If a match between incompatible encodings is attempted an `Encoding::CompatibilityError` exception is raised.

The `Regexp#fixed_encoding?` predicate indicates whether the regexp has a *fixed* encoding, that is one incompatible with ASCII. A regexp's encoding can be explicitly fixed by supplying `Regexp::FIXEDENCODING` as the second argument of `Regexp.new`:

```
r = Regexp.new("a".force_encoding("iso-8859-1"), Regexp::FIXEDENCODING)
r =~ "a\u3042"
#=> Encoding::CompatibilityError: incompatible encoding (ISO-8859-1 regexp with UTF-8 string)
```


Special global variables

Pattern matching sets some global variables :

- `$~` is equivalent to `Regexp.last_match`;
- `$&` contains the complete matched text;
- `$`` contains string before match;
- `$'` contains string after match;
- `$1`, `$2` and so on contain text matching first, second, etc capture group;
- `$+` contains last capture group.

Example:

```
m = /s(\w{2}).*(c)/.match('haystack') ==> #<MatchData
$~                                     ==> #<MatchData
Regexp.last_match                    ==> #<MatchData

$&      ==> "stac"
        # same as m[0]
$`      ==> "hay"
        # same as m.pre_match
$'      ==> "k"
        # same as m.post_match
$1      ==> "ta"
        # same as m[1]
$2      ==> "c"
        # same as m[2]
$3      ==> nil
        # no third group in pattern
$+      ==> "c"
        # same as m[-1]
```

These global variables are thread-local and method-local variables.

Performance

Certain pathological combinations of constructs can lead to abysmally bad performance.

Consider a string of 25 *as*, a *d*, 4 *as*, and a *c*.

```
s = 'a' * 25 + 'd' + 'a' * 4 + 'c'  
#=> "aaaaaaaaaaaaaaaaaaaaaaaaadaaac"
```

The following patterns match instantly as you would expect:

```
/(b|a)/ =~ s #=> 0  
/(b|a+)/ =~ s #=> 0  
/(b|a+)* / =~ s #=> 0
```

However, the following pattern takes appreciably longer:

```
/(b|a+)*c/ =~ s #=> 26
```

This happens because an atom in the regexp is quantified by both an immediate `+` and an enclosing `*` with nothing to differentiate which is in control of any particular character. The nondeterminism that results produces super-linear performance. (Consult *Mastering Regular Expressions* (3rd ed.), pp 222, by *Jeffery Friedl*, for an in-depth analysis). This particular case can be fixed by use of atomic grouping, which prevents the unnecessary backtracking:

```
(start = Time.now) && /(b|a+)*c/ =~ s && (Time.now -  
#=> 24.702736882  
(start = Time.now) && /(?:b|a+)*c/ =~ s && (Time.now  
#=> 0.000166571
```

A similar case is typified by the following example, which takes approximately 60 seconds to execute for me:

Match a string of 29 `as` against a pattern of 29 optional `as` followed by 29 mandatory `as`:

```
Regexp.new('a?' * 29 + 'a' * 29) =~ 'a' * 29
```

The 29 optional `as` match the string, but this prevents the 29 mandatory `as` that follow from matching. Ruby must then backtrack repeatedly so as to satisfy as many of the optional matches as it can while still matching the mandatory 29. It is plain to us that none of the optional matches can succeed, but this fact unfortunately eludes Ruby.

The best way to improve performance is to significantly reduce the amount of backtracking needed. For this case, instead of individually matching 29 optional `as`, a range of optional `as` can be matched all at once with `a{0,29}`:

```
Regexp.new('a{0,29}' + 'a' * 29) =~ 'a' * 29
```

Ruby Security

The Ruby programming language is large and complex and there are many security pitfalls often encountered by newcomers and experienced Rubyists alike.

This document aims to discuss many of these pitfalls and provide more secure alternatives where applicable.

Please check the full list of publicly known CVEs and how to correctly report a security vulnerability, at: www.ruby-lang.org/en/security/ Japanese version is here: www.ruby-lang.org/ja/security/

Security vulnerabilities should be reported via an email to security@ruby-lang.org (the [PGP public key](#)), which is a private mailing list. Reported problems will be published after fixes.

`$$SAFE`

Ruby provides a mechanism to restrict what operations can be performed by Ruby code in the form of the `$$SAFE` variable.

However, `$$SAFE` does not provide a secure environment for executing untrusted code.

If you need to execute untrusted code, you should use an operating system level sandboxing mechanism. On Linux, `ptrace` or `LXC` can be used to sandbox potentially malicious code. Other similar mechanisms exist on every major operating system.

Marshal.load

Ruby's `Marshal` module provides methods for serializing and deserializing Ruby object trees to and from a binary data format.

Never use `Marshal.load` to deserialize untrusted or user supplied data. Because `Marshal` can deserialize to almost any Ruby object and has full control over instance variables, it is possible to craft a malicious payload that executes code shortly after deserialization.

If you need to deserialize untrusted data, you should use JSON as it is only capable of returning 'primitive' types such as strings, arrays, hashes, numbers and nil. If you need to deserialize other classes, you should handle this manually. Never deserialize to a user specified class.

YAML

YAML is a popular human readable data serialization format used by many Ruby programs for configuration and database persistence of Ruby object trees.

Similar to `Marshal`, it is able to deserialize into arbitrary Ruby classes. For example, the following YAML data will create an `ERB` object when deserialized:

```
!ruby/object:ERB
src: puts `uname`
```

Because of this, many of the security considerations applying to `Marshal` are also applicable to `YAML`. Do not use `YAML` to deserialize untrusted data.

Symbols

Symbols are often seen as syntax sugar for simple strings, but they play a much more crucial role. The MRI Ruby implementation uses Symbols internally for method, variable and constant names. The reason for this is that symbols are simply integers with names attached to them, so they are faster to look up in hashtables.

Once a symbol is created, the memory used by it is never freed. If you convert user input to symbols with `to_sym` or `intern`, it is possible for an attacker to mount a denial of service attack against your application by flooding it with unique strings. Because each string is kept in memory until the Ruby process exits, this will cause memory consumption to grow and grow until Ruby runs out of memory and crashes.

Be careful with passing user input to methods such as `send`, `instance_variable_get` or `_set`, `const_get` or `_set`, etc. as these methods will convert string parameters to symbols internally and pose the same DoS potential as direct conversion through `to_sym/intern`.

The workaround to this is simple - don't convert user input to symbols. You should attempt to leave user input in string form instead.

Regular expressions

Ruby's regular expression syntax has some minor differences when compared to other languages. In Ruby, the `^` and `$` anchors do not refer to the beginning and end of the string, rather the beginning and end of a **line**.

This means that if you're using a regular expression like `/^[a-z]+$ /` to restrict a string to only letters, an attacker can bypass this check by passing a string containing a letter, then a newline, then any string of their choosing.

If you want to match the beginning and end of the entire string in Ruby, use the anchors `\A` and `\z`.

eval

Never pass untrusted or user controlled input to `eval`.

Unless you are implementing a REPL like `irb` or `pry`, `eval` is almost certainly not what you want. Do not attempt to filter user input before passing it to `eval` - this approach is fraught with danger and will most likely open your application up to a serious remote code execution vulnerability.

send

'Global functions' in Ruby (`puts`, `exit`, etc.) are actually private instance methods on `Object`. This means it is possible to invoke these methods with `send`, even if the call to `send` has an explicit receiver.

For example, the following code snippet writes "Hello world" to the terminal:

```
1.send(:puts, "Hello world")
```

You should never call `send` with user supplied input as the first parameter. Doing so can introduce a denial of service vulnerability:

```
foo.send(params[:bar]) # params[:bar] is "exit!"
```

If an attacker can control the first two arguments to `send`, remote code execution is possible:

```
# params is { :a => "eval", :b => "...ruby code to be  
foo.send(params[:a], params[:b])
```

When dispatching a method call based on user input, carefully verify that the method name. If possible, check it against a whitelist of safe method names.

Note that the use of `public_send` is also dangerous, as `send` itself is public:

```
1.public_send("send", "eval", "...ruby code to be exe
```

DRb

As DRb allows remote clients to invoke arbitrary methods, it is not suitable to expose to untrusted clients.

When using DRb, try to avoid exposing it over the network if possible. If this isn't possible and you need to expose DRb to the world, you **must** configure an appropriate security policy with `DRb : :ACL`.

Generated by [RDoc](#) 3.12.2.

Generated with the [Darkfish Rdoc Generator](#) 3.

Ruby Standard Library

The Ruby Standard Library is a vast collection of classes and modules that you can require in your code for additional features.

Below is an overview of libraries and extensions followed by a brief description.

Libraries

Abbrev

Calculates a set of unique abbreviations for a given set of strings

Base64

Support for encoding and decoding binary data using a Base64 representation

Benchmark

Provides methods to measure and report the time used to execute code

CGI

Support for the Common Gateway Interface protocol

CMath

Provides Trigonometric and Transcendental functions for complex numbers

complex.rb

Deprecated library replaced by C implementation in core

ConditionVariable

Augments the Mutex class, provided by thread.rb

CSV

Provides an interface to read and write CSV files and

data

DEBUGGER__

Debugging functionality for Ruby

Delegator

Provides three abilities to delegate method calls to an object

DRb

Distributed object system for Ruby

E2MM

Module for defining custom exceptions with specific messages

English.rb

Require 'English.rb' to reference global variables with less cryptic names

ERB

An easy to use but powerful templating system for Ruby

FileUtils

Several file utility methods for copying, moving, removing, etc

Find

This module supports top-down traversal of a set of file paths

Forwardable

Provides delegation of specified methods to a designated object

GetoptLong

Parse command line options similar to the GNU C `getopt_long()`

GServer

HTTP server with logging, thread pooling and multi-server management

IPAddr

Provides methods to manipulate IPv4 and IPv6 IP addresses

IRB

Interactive Ruby command-line tool for REPL (Read Eval Print Loop)

Logger

Provides a simple logging utility for outputting messages

mathn.rb

Deprecated library that extends math operations

MakeMakefile

Module used to generate a Makefile for C extensions

Matrix

Represents a mathematical matrix.

MiniTest

A test suite with TDD, BDD, mocking and benchmarking

Monitor

Provides an object or module to use safely by more than one thread

Mutex_m

Mixin to extend objects to be handled like a Mutex

Net::FTP

Support for the File Transfer Protocol

Net::HTTP

HTTP client api for Ruby

Net::IMAP

Ruby client api for Internet Message Access Protocol

Net::POP3

Ruby client library for POP3

Net::SMTP

Simple Mail Transfer Protocol client library for Ruby

Net::Telnet

Telnet client library for Ruby

Observable

Provides a mechanism for publish/subscribe pattern in Ruby

OpenURI

An easy-to-use wrapper for Net::HTTP, Net::HTTPS and Net::FTP

Open3

Provides access to stdin, stdout and stderr when running other programs

OptionParser

Ruby-oriented class for command-line option analysis

OpenStruct

Class to build custom data structures, similar to a Hash

PP

Provides a PrettyPrinter for Ruby objects

PrettyPrinter

Implements a pretty printing algorithm for readable structure

Prime

Prime numbers and factorization library

profile.rb

Runs the Ruby Profiler__

Profiler__

Provides a way to profile your Ruby application

PStore

Implements a file based persistence mechanism based on a Hash

Queue

Synchronized communication between threads, provided by thread.rb

Racc

A LALR(1) parser generator written in Ruby.

Rake

Ruby build program with capabilities similar to make

rational.rb

Deprecated library replaced by C implementation in core

RbConfig

Information of your configure and build of Ruby

RDoc

Produces HTML and command-line documentation for Ruby

resolv-replace.rb

Replace Socket DNS with Resolv

Resolv

Thread-aware DNS resolver library in Ruby

REXML

An XML toolkit for Ruby

Rinda

The Linda distributed computing paradigm in Ruby

RSS

Family of libraries that support various formats of XML “feeds”

Gem

Package management framework for Ruby

Scanf

A Ruby implementation of the C function scanf(3)

SecureRandom

Interface for secure random number generator

Set

Provides a class to deal with collections of unordered, unique values

Shell

An idiomatic Ruby interface for common UNIX shell commands

Shellwords

Manipulates strings with word parsing rules of UNIX Bourne shell

Singleton

Implementation of the Singleton pattern for Ruby

Synchronizer

A module that provides a two-phase lock with a counter

Tempfile

A utility class for managing temporary files

Test::Unit

A compatibility layer for MiniTest

Thread

Provides support classes for threaded programs

ThreadsWait

Watches for termination of multiple threads

Time

Extends the Time class with methods for parsing and conversion

Timeout

Auto-terminate potentially long-running operations in Ruby

tmpdir.rb

Extends the Dir class to manage the OS temporary file

path

Tracer

Outputs a source level execution trace of a Ruby program

TSort

Topological sorting using Tarjan's algorithm

un.rb

Utilities to replace common UNIX commands

URI

A Ruby module providing support for Uniform Resource Identifiers

WeakRef

Allows a referenced object to be garbage-collected

WEBrick

An HTTP server toolkit for Ruby

XMLRPC

Remote Procedure Call over HTTP support for Ruby

YAML

Ruby client library for the Psych YAML implementation

Extensions

BigDecimal

Provides arbitrary-precision floating point decimal arithmetic

Coverage

Provides coverage measurement for Ruby

Date

A subclass of Object includes Comparable module for handling dates

DateTime

Subclass of Date to handling dates, hours, minutes, seconds, offsets

DBM

Provides a wrapper for the UNIX-style Database Manager Library

Digest

Provides a framework for message digest libraries

Etc

Provides access to information typically stored in UNIX /etc directory

Fcntl

Loads constants defined in the OS fcntl.h C header file

Fiddle

A libffi wrapper for Ruby

GDBM

Ruby extension for the GNU dbm (gdbm) library

IO

Extensions for Ruby IO class, including wait and ::console

JSON

Implements Javascript Object Notation for Ruby

NKF

Ruby extension for Network Kanji Filter

objspace

Extends ObjectSpace module to add methods for internal statistics

OpenSSL

Provides SSL, TSL and general purpose cryptography for Ruby

Pathname

Representation of the name of a file or directory on the filesystem

Psych

A YAML parser and emitter for Ruby

PTY

Creates and manages pseudo terminals

Readline

Provides an interface for GNU Readline and Edit Line (libedit)

Ripper

Provides an interface for parsing Ruby programs into S-expressions

SBDM

Provides a simple file-based key-value store with String keys and values

Socket

Access underlying OS socket implementations

StringIO

Pseudo I/O on String objects

StringScanner

Provides lexical scanning operations on a String

Syslog

Ruby interface for the POSIX system logging facility

Tk

Provides a framework for building a Graphical User Interface (GUI)

WIN32OLE

Provides an interface for OLE Automation in Ruby

Zlib

Ruby interface for the zlib compression/decompression library

Generated by [RDoc 3.12.2](#).

Generated with the [Darkfish Rdoc Generator 3](#).

Ruby Syntax

The Ruby syntax is large and is split up into the following sections:

Literals

Numbers, Strings, Arrays, Hashes, etc.

Assignment

Assignment and variables

Control Expressions

if, unless, while, until, for, break, next, redo

Methods

Method and method argument syntax

Calling Methods

How to call a method (or send a message to a method)

Modules and Classes

Creating modules and classes including inheritance

Exceptions

Exception handling syntax

Precedence

Precedence of ruby operators

Refinements

Use and behavior of the experimental refinements feature

Miscellaneous

alias, undef, BEGIN, END

Generated by [RDoc 3.12.2](#).

Generated with the [Darkfish Rdoc Generator 3](#).

Assignment

In Ruby assignment uses the = (equals sign) character. This example assigns the number five to the local variable v:

```
v = 5
```

Assignment creates a local variable if the variable was not previously referenced.

Local Variable Names

A local variable name must start with a lowercase US-ASCII letter or a character with the eight bit set. Typically local variables are US-ASCII compatible since the keys to type them exist on all keyboards.

(Ruby programs must be written in a US-ASCII-compatible character set. In such character sets if the eight bit is set it indicates an extended character. Ruby allows local variables to contain such characters.)

A local variable name may contain letters, numbers, an _ (underscore or low line) or a character with the eighth bit set.

Local Variable Scope

Once a local variable name has been assigned to all uses of the name for the rest of the scope are considered local variables.

Here is an example:

```
1.times do
  a = 1
  puts "local variables in the block: #{local_variables}"
end

puts "no local variables outside the block" if local_variables.empty?
```

This prints:

```
local variables in the block: a
no local variables outside the block
```

Since the block creates a new scope, any local variables created inside it do not leak to the surrounding scope.

Variables defined in an outer scope appear in inner scope:

```
a = 0

1.times do
  puts "local variables: #{local_variables.join ", "}"
end
```

This prints:

```
local variables: a
```

You may isolate variables in a block from the outer scope by listing them following a `;` in the block's arguments. See the documentation for block local variables in the calling methods documentation for an example.

See also `Kernel#local_variables`, but note that a `for` loop does not create a new scope like a block does.

Local Variables and Methods

In Ruby local variable names and method names are nearly identical. If you have not assigned to one of these ambiguous names ruby will assume you wish to call a method. Once you have assigned to the name ruby will assume you wish to reference a local variable.

The local variable is created when the parser encounters the assignment, not when the assignment occurs:

```
a = 0 if false # does not assign to a
p local_variables # prints [:a]
p a # prints nil
```

The similarity between method and local variable names can lead to confusing code, for example:

```
def big_calculation
  42 # pretend this takes a long time
end

big_calculation = big_calculation()
```

Now any reference to `big_calculation` is considered a local variable and will be cached. To call the method, use `self.big_calculation`.

You can force a method call by using empty argument parentheses as shown above or by using an explicit receiver like `self..` Using an explicit receiver may raise a `NameError` if the method's visibility is not public.

Another commonly confusing case is when using a modifier `if`:

```
p a if a = 0.zero?
```

Rather than printing “true” you receive a `NameError`, “undefined local variable or method `a`”. Since ruby parses the bare `a` left of the `if` first and has not yet seen an assignment to `a` it assumes you wish to call a method. Ruby then sees the assignment to `a` and will assume you are referencing a local method.

The confusion comes from the out-of-order execution of the expression. First the local variable is assigned-to then you attempt to call a nonexistent method.

Instance Variables

Instance variables are shared across all methods for the same object.

An instance variable must start with a @ (“at” sign or commercial at). Otherwise instance variable names follow the rules as local variable names. Since the instance variable starts with an @ the second character may be an upper-case letter.

Here is an example of instance variable usage:

```
class C
  def initialize(value)
    @instance_variable = value
  end

  def value
    @instance_variable
  end
end

object1 = C.new "some value"
object2 = C.new "other value"

p object1.value # prints "some value"
p object2.value # prints "other value"
```

An uninitialized instance variable has a value of `nil`. If you run Ruby with warnings enabled you will get a warning when accessing an uninitialized instance variable.

The `value` method has access to the value set by the `initialize` method, but only for the same object.

Class Variables

Class variables are shared between a class, its subclasses and its instances.

A class variable must start with a @@ (two “at” signs). The rest of the name follows the same rules as instance variables.

Here is an example:

```
class A
  @@class_variable = 0

  def value
    @@class_variable
  end

  def update
    @@class_variable = @@class_variable + 1
  end
end

class B < A
  def update
    @@class_variable = @@class_variable + 2
  end
end

a = A.new
b = B.new

puts "A value: #{a.value}"
puts "B value: #{b.value}"
```

This prints:

```
A value: 0
B value: 0
```

Continuing with the same example, we can update using objects from either class and the value is shared:

```
puts "update A"
a.update

puts "A value: #{a.value}"
puts "B value: #{b.value}"

puts "update B"
b.update

puts "A value: #{a.value}"
puts "B value: #{b.value}"

puts "update A"
a.update

puts "A value: #{a.value}"
puts "B value: #{b.value}"
```

This prints:

```
update A
A value: 1
B value: 1
update B
A value: 3
B value: 3
update A
A value: 4
B value: 4
```

Accessing an uninitialized class variable will raise a `NameError` exception.

Note that classes have instance variables because classes are objects, so try not to confuse class and instance variables.

Global Variables

Global variables are accessible everywhere.

Global variables start with a \$ (dollar sign). The rest of the name follows the same rules as instance variables.

Here is an example:

```
$global = 0

class C
  puts "in a class: #{ $global }"

  def my_method
    puts "in a method: #{ $global }"

    $global = $global + 1
    $other_global = 3
  end
end

C.new.my_method

puts "at top-level, $global: #{ $global }, $other_global: #{ $other_global }"
```

This prints:

```
in a class: 0
in a method: 0
at top-level, $global: 1, $other_global: 3
```

An uninitialized global variable has a value of `nil`.

Ruby has some special globals that behave differently depending on context such as the regular expression match variables or that have a side-effect when assigned to. See the global variables documentation for details.

Assignment Methods

You can define methods that will behave like assignment, for example:

```
class C
  def value=(value)
    @value = value
  end
end

c = C.new
c.value = 42
```

Using assignment methods allows your programs to look nicer. When assigning to an instance variable most people use `Module#attr_accessor`:

```
class C
  attr_accessor :value
end
```

When using method assignment you must always have a receiver. If you do not have a receiver Ruby assumes you are assigning to a local variable:

```
class C
  attr_accessor :value

  def my_method
    value = 42

    puts "local_variables: #{local_variables.join " "},
    puts "@value: #{@value.inspect}"
  end
end

C.new.my_method
```

This prints:

```
local_variables: value
@value: nil
```

To use the assignment method you must set the receiver:

```
class C
  attr_accessor :value

  def my_method
    self.value = 42

    puts "local_variables: #{local_variables.join " ",
    puts "@value: #{@value.inspect}"
  end
end

C.new.my_method
```

This prints:

```
local_variables:
@value: 42
```


Abbreviated Assignment

You can mix several of the operators and assignment. To add 1 to an object you can write:

```
a = 1
a += 2
p a # prints 3
```

This is equivalent to:

```
a = 1
a = a + 2
p a # prints 3
```

You can use the following operators this way: +, -, *, /, %, **, &, |, ^, <<, >>

There are also ||= and &&=. The former makes an assignment if the value was nil or false while the latter makes an assignment if the value was not nil or false.

Here is an example:

```
a ||= 0
a &&= 1
p a # prints 1
```

Note that these two operators behave more like a ||= a = 0 than a = a ||= 0.

Implicit Array Assignment

You can implicitly create an array by listing multiple values when assigning:

```
a = 1, 2, 3  
p a # prints [1, 2, 3]
```

This implicitly creates an Array.

You can use `*` or the “splat” operator or unpack an Array when assigning. This is similar to multiple assignment:

```
a = *[1, 2, 3]  
p a # prints [1, 2, 3]
```

You can splat anywhere in the right-hand side of the assignment:

```
a = 1, *[2, 3]  
p a # prints [1, 2, 3]
```

Multiple Assignment

You can assign multiple values on the right-hand side to multiple variables:

```
a, b = 1, 2  
  
p a: a, b: b # prints {:a=>1, :b=>2}
```

In the following sections any place “variable” is used an assignment method, instance, class or global will also work:

```
def value=(value)  
  p assigned: value  
end  
  
self.value, $global = 1, 2 # prints {:assigned=>1}  
  
p $global # prints 2
```

You can use multiple assignment to swap two values in-place:

```
old_value = 1  
  
new_value, old_value = old_value, 2  
  
p new_value: new_value, old_value: old_value  
# prints {:new_value=>1, :old_value=>2}
```

If you have more values on the right hand side of the assignment than variables on the left hand side the extra values are ignored:

```
a, b = 1, 2, 3  
  
p a: a, b: b # prints {:a=>1, :b=>2}
```

You can use `*` to gather extra values on the right-hand side of the assignment.

```
a, *b = 1, 2, 3  
p a: a, b: b # prints {:a=>1, :b=>[2, 3]}
```

The `*` can appear anywhere on the left-hand side:

```
*a, b = 1, 2, 3  
p a: a, b: b # prints {:a=>[1, 2], :b=>3}
```

But you may only use one `*` in an assignment.

Array Decomposition

Like Array decomposition in method arguments you can decompose an Array during assignment using parenthesis:

```
(a, b) = [1, 2]
p a: a, b: b # prints {:a=>1, :b=>2}
```

You can decompose an Array as part of a larger multiple assignment:

```
a, (b, c) = 1, [2, 3]
p a: a, b: b, c: c # prints {:a=>1, :b=>2, :c=>3}
```

Since each decomposition is considered its own multiple assignment you can use * to gather arguments in the decomposition:

```
a, (b, *c), *d = 1, [2, 3, 4], 5, 6
p a: a, b: b, c: c, d: d
# prints {:a=>1, :b=>2, :c=>[3, 4], :d=>[5, 6]}
```

Calling Methods

Calling a method sends a message to an object so it can perform some work.

In ruby you send a message to an object like this:

```
my_method()
```

Note that the parenthesis are optional:

```
my_method
```

Except when there is difference between using and omitting parentheses, this document uses parenthesis when arguments are present to avoid confusion.

This section only covers calling methods. See also the [syntax documentation on defining methods](#)

Receiver

`self` is the default receiver. If you don't specify any receiver `self` will be used. To specify a receiver use `.`:

```
my_object.my_method
```

This sends the `my_method` message to `my_object`. Any object can be a receiver but depending on the method's visibility sending a message may raise a `NoMethodError`.

You may also use `::` to designate a receiver, but this is rarely used due to the potential for confusion with `::` for namespaces.

Arguments

There are three types of arguments when sending a message, the positional arguments, keyword (or named) arguments and the block argument. Each message sent may use one, two or all types of arguments, but the arguments must be supplied in this order.

All arguments in ruby are passed by reference and are not lazily evaluated.

Each argument is separated by a , :

```
my_method(1, '2', :three)
```

Arguments may be an expression, a hash argument:

```
'key' => value
```

or a keyword argument:

```
key: value
```

Hash and keyword arguments must be contiguous and must appear after all positional arguments, but may be mixed:

```
my_method('a' => 1, b: 2, 'c' => 3)
```

Positional Arguments

The positional arguments for the message follow the method name:


```
my_method(argument1, argument2)
```

In many cases parenthesis are not necessary when sending a message:

```
my_method argument1, argument2
```

However, parenthesis are necessary to avoid ambiguity. This will raise a `SyntaxError` because ruby does not know which method argument3 should be sent to:

```
method_one argument1, method_two argument2, argument3
```

If the method definition has a `*argument` extra positional arguments will be assigned to `argument` in the method as an `Array`.

If the method definition doesn't include keyword arguments the keyword or hash-type arguments are assigned as a single hash to the last argument:

```
def my_method(options)
  p options
end

my_method('a' => 1, b: 2) # prints: {'a'=>1, :b=>2}
```

If too many positional arguments are given an `ArgumentError` is raised.

Default Positional Arguments

When the method defines default arguments you do not need to supply all the arguments to the method. Ruby will fill in the missing arguments in-order.

First we'll cover the simple case where the default arguments appear on the right. Consider this method:

```
def my_method(a, b, c = 3, d = 4)
  p [a, b, c, d]
end
```

Here `c` and `d` have default values which ruby will apply for you. If you send only two arguments to this method:

```
my_method(1, 2)
```

You will see ruby print `[1, 2, 3, 4]`.

If you send three arguments:

```
my_method(1, 2, 5)
```

You will see ruby print `[1, 2, 5, 4]`

Ruby fills in the missing arguments from left to right.

Ruby allows default values to appear in the middle of positional arguments. Consider this more complicated method:

```
def my_method(a, b = 2, c = 3, d)
  p [a, b, c, d]
end
```

Here `b` and `c` have default values. If you send only two arguments to this method:

```
my_method(1, 4)
```

You will see ruby print `[1, 2, 3, 4]`.

If you send three arguments:

```
my_method(1, 5, 6)
```

You will see ruby print [1, 5, 3, 6].

Describing this in words gets complicated and confusing. I'll describe it in variables and values instead.

First 1 is assigned to a, then 6 is assigned to d. This leaves only the arguments with default values. Since 5 has not been assigned to a value yet, it is given to b and c uses its default value of 3.

Keyword Arguments

Keyword arguments follow any positional arguments and are separated by commas like positional arguments:

```
my_method(positional1, keyword1: value1, keyword2: va
```

Any keyword arguments not given will use the default value from the method definition. If a keyword argument is given that the method did not list an `ArgumentError` will be raised.

Block Argument

The block argument sends a closure from the calling scope to the method.

The block argument is always last when sending a message to a method. A block is sent to a method using `do ... end` or `{ ... }`:

```
my_method do
```

```
# ...  
end
```

or:

```
my_method {  
  # ...  
}
```

`do end` has lower precedence than `{ }` so:

```
method_1 method_2 {  
  # ...  
}
```

Sends the block to `method_2` while:

```
method_1 method_2 do  
  # ...  
end
```

Sends the block to `method_1`. Note that in the first case if parentheses are used the block is sent to `method_1`.

A block will accept arguments from the method it was sent to. Arguments are defined similar to the way a method defines arguments. The block's arguments go in `| ... |` following the opening `do` or `{`:

```
my_method do |argument1, argument2|  
  # ...  
end
```

Block Local Arguments

You may also declare block-local arguments to a block using `;` in the block arguments list. Assigning to a block-local argument will not override local arguments outside

the block in the caller's scope:

```
def my_method
  yield self
end

place = "world"

my_method do |obj; place|
  place = "block"
  puts "hello #{obj} this is #{place}"
end

puts "place is: #{place}"
```

This prints:

```
hello main this is block
place is world
```

So the `place` variable in the block is not the same `place` variable as outside the block. Removing `; place` from the block arguments gives this result:

```
hello main this is block
place is block
```

Array to Arguments Conversion

Given the following method:

```
def my_method(argument1, argument2, argument3)
end
```

You can turn an Array into an argument list with `*` (or `splat`) operator:

```
arguments = [1, 2, 3]
my_method(*arguments)
```

or:

```
arguments = [2, 3]
my_method(1, *arguments)
```

Both are equivalent to:

```
my_method(1, 2, 3)
```

If the method accepts keyword arguments the splat operator will convert a hash at the end of the array into keyword arguments:

```
def my_method(a, b, c: 3)
end

arguments = [1, 2, { c: 4 }]
my_method(*arguments)
```

You may also use the ** (described next) to convert a Hash into keyword arguments.

If the number of objects in the Array do not match the number of arguments for the method an ArgumentError will be raised.

If the splat operator comes first in the call, parentheses must be used to avoid a warning.

Hash to Keyword Arguments Conversion

Given the following method:

```
def my_method(first: 1, second: 2, third: 3)
end
```

You can turn a Hash into keyword arguments with the `**` operator:

```
arguments = { first: 3, second: 4, third: 5 }  
my_method(**arguments)
```

or:

```
arguments = { first: 3, second: 4 }  
my_method(third: 5, **arguments)
```

Both are equivalent to:

```
my_method(first: 3, second: 4, third: 5)
```

If the method definition uses `**` to gather arbitrary keyword arguments they will not be gathered by `*`:

```
def my_method(*a, **kw)  
  p arguments: a, keywords: kw  
end  
  
my_method(1, 2, '3' => 4, five: 6)
```

Prints:

```
{:arguments=>[1, 2], :keywords=>{"3"=>4, :five=>6}}
```

Unlike the splat operator described above the `**` operator has no commonly recognized name.

Proc to Block Conversion

Given a method that use a block:

```
def my_method  
  yield self
```

```
end
```

You can convert a proc or lambda to a block argument with the & operator:

```
argument = proc { |a| puts "#{a.inspect} was yielded" }  
my_method(&argument)
```

If the splat operator comes first in the call, parenthesis must be used to avoid a warning.

Unlike the splat operator described above the & operator has no commonly recognized name.

Method Lookup

When you send a message Ruby looks up the method that matches the name of the message for the receiver. Methods are stored in classes and modules so method lookup walks these, not the objects themselves.

Here is the order of method lookup for the receiver's class or module R :

- The prepended modules of R in reverse order
- For a matching method in R
- The included modules of R in reverse order

If R is a class with a superclass, this is repeated with R 's superclass until a method is found.

Once a match is found method lookup stops.

If no match is found this repeats from the beginning, but looking for `method_missing`. The default `method_missing` is `BasicObject#method_missing` which raises a `NameError` when invoked.

If refinements (an experimental feature) are active the method lookup changes. See the refinements documentation for details.

Control Expressions

Ruby has a variety of ways to control execution. All the expressions described here return a value.

For the tests in these control expressions, `nil` and `false` are false-values and `true` and any other object are true-values. In this document “true” will mean “true-value” and “false” will mean “false-value”.

if Expression

The simplest `if` expression has two parts, a “test” expression and a “then” expression. If the “test” expression evaluates to a true then the “then” expression is evaluated.

Here is a simple if statement:

```
if true then
  puts "the test resulted in a true-value"
end
```

This will print “the test resulted in a true-value”.

The `then` is optional:

```
if true
  puts "the test resulted in a true-value"
end
```

This document will omit the optional `then` for all expressions as that is the most common usage of `if`.

You may also add an `else` expression. If the test does not evaluate to true the `else` expression will be executed:

```
if false
  puts "the test resulted in a true-value"
else
  puts "the test resulted in a false-value"
end
```

This will print “the test resulted in a false-value”.

You may add an arbitrary number of extra tests to an `if` expression using `elsif`. An `elsif` executes when all tests

above the `elsif` are false.

```
a = 1

if a == 0
  puts "a is zero"
elsif a == 1
  puts "a is one"
else
  puts "a is some other value"
end
```

This will print “a is one” as `1` is not equal to `0`. Since `else` is only executed when there are no matching conditions.

Once a condition matches, either the `if` condition or any `elsif` condition, the `if` expression is complete and no further tests will be performed.

Like an `if`, an `elsif` condition may be followed by a `then`.

In this example only “a is one” is printed:

```
a = 1

if a == 0
  puts "a is zero"
elsif a == 1
  puts "a is one"
elsif a >= 1
  puts "a is greater than or equal to one"
else
  puts "a is some other value"
end
```

The tests for `if` and `elsif` may have side-effects. The most common use of side-effect is to cache a value into a local variable:

```
if a = object.some_value
  # do something to a
```

end

The result value of an `if` expression is the last value executed in the expression.

Ternary if

You may also write a if-then-else expression using ? and :. This ternary if:

```
input_type = gets =~ /hello/i ? "greeting" : "other"
```

Is the same as this if expression:

```
input_type =  
  if gets =~ /hello/i  
    "greeting"  
  else  
    "other"  
  end
```

While the ternary if is much shorter to write than the more verbose form, for readability it is recommended that the ternary if is only used for simple conditionals. Also, avoid using multiple ternary conditions in the same expression as this can be confusing.

unless Expression

The `unless` expression is the opposite of the `if` expression. If the value is false the “then” expression is executed:

```
unless true
  puts "the value is a false-value"
end
```

This prints nothing as `true` is not a false-value.

You may use an optional `then` with `unless` just like `if`.

Note that the above `unless` expression is the same as:

```
if not true
  puts "the value is a false-value"
end
```

Like an `if` expression you may use an `else` condition with `unless`:

```
unless true
  puts "the value is false"
else
  puts "the value is true"
end
```

This prints “the value is true” from the `else` condition.

You may not use `elsif` with an `unless` expression.

The result value of an `unless` expression is the last value executed in the expression.

Modifier `if` and `unless`

`if` and `unless` can also be used to modify an expression. When used as a modifier the left-hand side is the “then” expression and the right-hand side is the “test” expression:

```
a = 0
a += 1 if a.zero?
p a
```

This will print 1.

```
a = 0
a += 1 unless a.zero?
p a
```

This will print 0.

While the modifier and standard versions have both a “test” expression and a “then” expression, they are not exact transformations of each other due to parse order. Here is an example that shows the difference:

```
p a if a = 0.zero?
```

This raises the `NameError` “undefined local variable or method ``a``”.

When ruby parses this expression it first encounters `a` as a method call in the “then” expression, then later it sees the assignment to `a` in the “test” expression and marks `a`

as a local variable.

When running this line it first executes the “test” expression, `a = 0.zero?`.

Since the test is true it executes the “then” expression, `p a`. Since the `a` in the body was recorded as a method which does not exist the `NameError` is raised.

The same is true for `unless`.

case Expression

The `case` expression can be used in two ways.

The most common way is to compare an object against multiple patterns. The patterns are matched using the `===` method which is aliased to `==` on `Object`. Other classes must override it to give meaningful behavior. See `Module#===` and `Regexp#===` for examples.

Here is an example of using `case` to compare a `String` against a pattern:

```
case "12345"  
when /^1/  
  puts "the string starts with one"  
else  
  puts "I don't know what the string starts with"  
end
```

Here the string `"12345"` is compared with `/^1/` by calling `/^1/ === "12345"` which returns `true`. Like the `if` expression the first `when` that matches is executed and all other matches are ignored.

If no matches are found the `else` is executed.

The `else` and `then` are optional, this `case` expression gives the same result as the one above:

```
case "12345"  
when /^1/  
  puts "the string starts with one"  
end
```

You may place multiple conditions on the same `when`:

```
case "2"  
when /^1/, "2"  
  puts "the string starts with one or is '2'"  
end
```

Ruby will try each condition in turn, so first `/^1/ === "2"` returns `false`, then `"2" === "2"` returns `true`, so “the string starts with one or is '2'” is printed.

You may use `then` after the `when` condition. This is most frequently used to place the body of the `when` on a single line.

```
case a  
when 1, 2 then puts "a is one or two"  
when 3   then puts "a is three"  
else     puts "I don't know what a is"  
end
```

The other way to use a `case` expression is like an `if-elsif` expression:

```
a = 2  
  
case  
when a == 1, a == 2  
  puts "a is one or two"  
when a == 3  
  puts "a is three"  
else  
  puts "I don't know what a is"  
end
```

Again, the `then` and `else` are optional.

The result value of a `case` expression is the last value executed in the expression.

while Loop

The `while` loop executes while a condition is true:

```
a = 0
while a < 10 do
  p a
  a += 1
end

p a
```

Prints the numbers 0 through 10. The condition `a < 10` is checked before the loop is entered, then the body executes, then the condition is checked again. When the condition results in false the loop is terminated.

The `do` keyword is optional. The following loop is equivalent to the loop above:

```
while a < 10
  p a
  a += 1
end
```

The result of a `while` loop is `nil` unless `break` is used to supply a value.

until Loop

The `until` loop executes while a condition is false:

```
a = 0
until a > 10 do
  p a
  a += 1
end

p a
```

This prints the numbers 0 through 11. Like a `while` loop the condition `a > 10` is checked when entering the loop and each time the loop body executes. If the condition is false the loop will continue to execute.

Like a `while` loop the `do` is optional.

Like a `while` loop the result of an `until` loop is `nil` unless `break` is used.

for Loop

The `for` loop consists of `for` followed by a variable to contain the iteration argument followed by `in` and the value to iterate over using `each`. The `do` is optional:

```
for value in [1, 2, 3] do
  puts value
end
```

Prints 1, 2 and 3.

Like `while` and `until`, the `do` is optional.

The `for` loop is similar to using `each`, but does not create a new variable scope.

The result value of a `for` loop is the value iterated over unless `break` is used.

The `for` loop is rarely used in modern ruby programs.

Modifier `while` and `until`

Like `if` and `unless`, `while` and `until` can be used as modifiers:

```
a = 0
a += 1 while a < 10
p a # prints 10
```

`until` used as a modifier:

```
a = 0
a += 1 until a > 10
p a # prints 11
```

You can use `begin` and `end` to create a `while` loop that runs the body once before the condition:

```
a = 0
begin
  a += 1
end while a < 10
p a # prints 10
```

If you don't use `rescue` or `ensure` Ruby optimizes away any exception handling overhead.

break Statement

Use `break` to leave a block early. This will stop iterating over the items in `values` if one of them is even:

```
values.each do |value|
  break if value.even?

  # ...
end
```

You can also terminate from a `while` loop using `break`:

```
a = 0

while true do
  p a
  a += 1

  break if a < 10
end

p a
```

This prints the numbers 0 and 1.

`break` accepts a value that supplies the result of the expression it is “breaking” out of:

```
result = [1, 2, 3].each do |value|
  break value * 2 if value.even?
end

p result # prints 4
```


next Statement

Use `next` to skip the rest of the current iteration:

```
result = [1, 2, 3].map do |value|
  next if value.odd?

  value * 2
end

p result # prints [2, nil, 6]
```

`next` accepts an argument that can be used the result of the current block iteration:

```
result = [1, 2, 3].map do |value|
  next value if value.odd?

  value * 2
end

p result # prints [2, 2, 6]
```

redo Statement

Use `redo` to redo the current iteration:

```
result = []  
  
while result.length < 10 do  
  result << result.length  
  
  redo if result.last.even?  
  
  result << result.length + 1  
end  
  
p result
```

This prints `[0, 1, 3, 3, 5, 5, 7, 7, 9, 9, 11]`

In Ruby 1.8 you could also use `retry` where you used `redo`. This is no longer true, now you will receive a `SyntaxError` when you use `retry` outside of a `rescue` block. See [Exceptions](#) for proper usage of `retry`.

Flip-Flop

The flip-flop is rarely seen conditional expression. It's primary use is for processing text from ruby one-line programs used with `ruby -n` or `ruby -p`.

The form of the flip-flop is an expression that indicates when the flip-flop turns on, `..` (or `...`), then an expression that indicates when the flip-flop will turn off. While the flip-flop is on it will continue to evaluate to `true`, and `false` when off.

Here is an example:

```
selected = []

0.upto 10 do |value|
  selected << value if value==2..value==8
end

p selected # prints [2, 3, 4, 5, 6, 7, 8]
```

In the above example the on condition is `n==2`. The flip-flop is initially off (false) for 0 and 1, but becomes on (true) for 2 and remains on through 8. After 8 it turns off and remains off for 9 and 10.

The flip-flop must be used inside a conditional such as `if`, `while`, `unless`, `until` etc. including the modifier forms.

When you use an inclusive range (`..`) the off condition is evaluated when the on condition changes:

```
selected = []

0.upto 5 do |value|
  selected << value if value==2..value==2
end
```

```
p selected # prints [2]
```

Here both sides of the flip-flop are evaluated so the flip-flop turns on and off only when `value` equals 2. Since the flip-flop turned on in the iteration it returns true.

When you use an exclusive range (`..`) the off condition is evaluated on the following iteration:

```
selected = []  
  
0.upto 5 do |value|  
  selected << value if value==2...value==2  
end  
  
p selected # prints [2, 3, 4, 5]
```

Here the flip-flop turns on when `value` equals 2 but doesn't turn off on the same iteration. The off condition isn't evaluated until the following iteration and `value` will never be two again.

Generated by [RDoc](#) 3.12.2.

Generated with the [Darkfish Rdoc Generator](#) 3.

Exception Handling

Exceptions are rescued in a `begin/end` block:

```
begin
  # code that might raise
rescue
  # handle exception
end
```

If you are inside a method you do not need to use `begin` or `end` unless you wish to limit the scope of rescued exceptions:

```
def my_method
  # ...
rescue
  # ...
end
```

The same is true for a `class` or `module`.

You can assign the exception to a local variable by using `=>` `variable_name` at the end of the `rescue` line:

```
begin
  # ...
rescue => exception
  warn exception.message
  raise # re-raise the current exception
end
```

By default `StandardError` and its subclasses are rescued. You can rescue a specific set of exception classes (and their subclasses) by listing them after `rescue`:

```
begin
```

```
# ...
rescue ArgumentError, NameError
  # handle ArgumentError or NameError
end
```

You may rescue different types of exceptions in different ways:

```
begin
  # ...
rescue ArgumentError
  # handle ArgumentError
rescue NameError
  # handle NameError
rescue
  # handle any StandardError
end
```

The exception is matched to the rescue section starting at the top, and matches only once. If an `ArgumentError` is raised in the `begin` section it will not be handled in the `StandardError` section.

You may retry rescued exceptions:

```
begin
  # ...
rescue
  # do something that may change the result of the be
  retry
end
```

Execution will resume at the start of the `begin` block, so be careful not to create an infinite loop.

Inside a rescue block is the only valid location for `retry`, all other uses will raise a `SyntaxError`. If you wish to retry a block iteration use `redo`. See [Control Expressions](#) for details.

To always run some code whether an exception was raised or not, use `ensure`:

```
begin
  # ...
rescue
  # ...
ensure
  # this always runs
end
```

You may also run some code when an exception is not raised:

```
begin
  # ...
rescue
  # ...
else
  # this runs only when no exception was raised
ensure
  # ...
end
```

Generated by [RDoc 3.12.2](#).

Generated with the [Darkfish Rdoc Generator 3](#).

Literals

Literals create objects you can use in your program.
Literals include:

- Booleans and nil
- Numbers
- Strings
- Symbols
- Arrays
- Hashes
- Ranges
- Regular Expressions
- Procs

Booleans and nil

`nil` and `false` are both false values. `nil` is sometimes used to indicate “no value” or “unknown” but evaluates to `false` in conditional expressions.

`true` is a true value. All objects except `nil` and `false` evaluate to a true value in conditional expressions.

(There are also the constants `TRUE`, `FALSE` and `NIL`, but the lowercase literal forms are preferred.)

Numbers

You can write integers of any size as follows:

```
1234  
1_234
```

These numbers have the same value, 1,234. The underscore may be used to enhance readability for humans. You may place an underscore anywhere in the number.

Floating point numbers may be written as follows:

```
12.34  
1234e-2  
1.234E1
```

These numbers have the same value, 12.34. You may use underscores in floating point numbers as well.

You can use a special prefix to write numbers in decimal, hexadecimal, octal or binary formats. For decimal numbers use a prefix of `0d`, for hexadecimal numbers use a prefix of `0x`, for octal numbers use a prefix of `0` or `0o`, for binary numbers use a prefix of `0b`. The alphabetic component of the number is not case-sensitive.

Examples:

```
0d170  
0D170  
  
0xaa  
0xAa  
0xAA  
0Xaa
```

```
0xAa  
0xA
```

```
0252  
0o252  
00252
```

```
0b10101010  
0B10101010
```

All these numbers have the same decimal value, 170.
Like integers and floats you may use an underscore for readability.

Strings

The most common way of writing strings is using ":

```
"This is a string."
```

The string may be many lines long.

Any internal " must be escaped:

```
"This string has a quote: \". As you can see, it is
```

Double-quote strings allow escaped characters such as `\n` for newline, `\t` for tab, etc.

Double-quote strings allow interpolation of other values using `#{...}`:

```
"One plus one is two: #{1 + 1}"
```

Any expression may be placed inside the interpolated section, but it's best to keep the expression small for readability.

Interpolation may be disabled by escaping the “#” character or using single-quote strings:

```
'#{1 + 1}' #=> "\#{1 + 1}"
```

In addition to disabling interpolation, single-quoted strings also disable all escape sequences except for the single-quote (`\'`) and backslash (`<tt>\</tt>`).

You may also create strings using %:

```
%Q(1 + 1 is #{1 + 1}) #=> "1 + 1 is 2"
```

There are two different types of % strings %q(...) behaves like a single-quote string (no interpolation or character escaping) while %Q behaves as a double-quote string. See Percent Strings below for more discussion of the syntax of percent strings.

Adjacent string literals are automatically concatenated by the interpreter:

```
"con" "cat" "en" "at" "ion" #=> "concatenation"  
"This string contains " "no newlines." #
```

Any combination of adjacent single-quote, double-quote, percent strings will be concatenated as long as a percent-string is not last.

```
%q{a} 'b' "c" #=> "abc"  
"a" 'b' %q{c} #=> NameError: uninitialized constant q
```

Here Documents

If you are writing a large block of text you may use a “here document” or “heredoc”:

```
expected_result = <<HEREDOC  
This would contain specially formatted text.  
  
That might span many lines  
HEREDOC
```

The heredoc starts on the line following <<HEREDOC and ends with the next line that starts with HEREDOC. The result includes the ending newline.

You may use any identifier with a heredoc, but all-uppercase identifiers are typically used.

You may indent the ending identifier if you place a "-" after <<:

```
expected_result = <<-INDENTED_HEREDOC
This would contain specially formatted text.

That might span many lines
INDENTED_HEREDOC
```

Note that while the closing identifier may be indented, the content is always treated as if it is flush left. If you indent the content those spaces will appear in the output.

A heredoc allows interpolation and escaped characters. You may disable interpolation and escaping by surrounding the opening identifier with single quotes:

```
expected_result = <<- 'EXPECTED'
One plus one is #{1 + 1}
EXPECTED

p expected_result # prints: "One plus one is \#{1 + 1}
```

The identifier may also be surrounded with double quotes (which is the same as no quotes) or with backticks. When surrounded by backticks the HEREDOC behaves like Kernel#`:

```
puts <<- `HEREDOC`
cat #{__FILE__}
HEREDOC
```

To call a method on a heredoc place it after the opening identifier:

```
expected_result = <<-EXPECTED.chomp  
One plus one is #{1 + 1}  
EXPECTED
```

You may open multiple heredocs on the same line, but this can be difficult to read:

```
puts(<<-ONE, <<-TWO)  
content for heredoc one  
ONE  
content for heredoc two  
TWO
```

Symbols

A Symbol represents a name inside the ruby interpreter. See Symbol for more details on what symbols are and when ruby creates them internally.

You may reference a symbol using a colon: `:my_symbol`.

You may also create symbols by interpolation:

```
:"my_symbol1"  
:"my_symbol#{1 + 1}"
```

Like strings, a single-quote may be used to disable interpolation:

```
:'my_symbol#{1 + 1}' #=> :"my_symbol\#{1 + 1}"
```

When creating a Hash there is a special syntax for referencing a Symbol as well.

Arrays

An array is created using the objects between [and]:

```
[1, 2, 3]
```

You may place expressions inside the array:

```
[1, 1 + 1, 1 + 2]  
[1, [1 + 1, [1 + 2]]]
```

See [Array](#) for the methods you may use with an array.

Hashes

A hash is created using key-value pairs between { and }:

```
{ "a" => 1, "b" => 2 }
```

Both the key and value may be any object.

You can create a hash using symbol keys with the following syntax:

```
{ a: 1, b: 2 }
```

This same syntax is used for keyword arguments for a method.

See Hash for the methods you may use with a hash.

Ranges

A range represents an interval of values. The range may include or exclude its ending value.

```
(1..2) # includes its ending value  
(1...2) # excludes its ending value
```

You may create a range of any object. See the Range documentation for details on the methods you need to implement.

Regular Expressions

A regular expression is created using “/”:

```
/my regular expression/
```

The regular expression may be followed by flags which adjust the matching behavior of the regular expression. The “i” flag makes the regular expression case-insensitive:

```
/my regular expression/i
```

Interpolation may be used inside regular expressions along with escaped characters. Note that a regular expression may require additional escaped characters than a string.

See Regexp for a description of the syntax of regular expressions.

Procs

A proc can be created with `->`:

```
-> { 1 + 1 }
```

Calling the above proc will give a result of 2.

You can require arguments for the proc as follows:

```
->(v) { 1 + v }
```

This proc will add one to its argument.

Percent Strings

Besides `%(. . .)` which creates a String, The `%` may create other types of object. As with strings, an uppercase letter allows interpolation and escaped characters while a lowercase letter disables them.

These are the types of percent strings in ruby:

`%i`

Array of Symbols

`%q`

String

`%r`

Regular Expression

`%s`

Symbol

`%w`

Array of Strings

`%x`

Backtick (capture subshell result)

For the two array forms of percent string, if you wish to include a space in one of the array entries you must escape it with a `"\"` character:

```
%w[one one-hundred\ one]
#=> ["one", "one-hundred one"]
```

If you are using “(”, “[”, “{”, “<” you must close it with “)”, “]”, “}”, “>” respectively. You may use most other non-alphanumeric characters for percent string delimiters such as “%”, “|”, “^”, etc.

Generated by [RDoc](#) 3.12.2.

Generated with the [Darkfish Rdoc Generator](#) 3.

Methods

Methods implement the functionality of your program. Here is a simple method definition:

```
def one_plus_one
  1 + 1
end
```

A method definition consists of the `def` keyword, a method name, the body of the method, `return` value and the `end` keyword. When called the method will execute the body of the method. This method returns 2.

This section only covers defining methods. See also the [syntax documentation on calling methods](#)

Method Names

Method names may be one of the operators or must start a letter or a character with the eight bit set. It may contain letters, numbers, an `_` (underscore or low line) or a character with the eight bit set. The convention is to use underscores to separate words in a multiword method name:

```
def method_name
  puts "use underscores to separate words"
end
```

Ruby programs must be written in a US-ASCII-compatible character set such as UTF-8, ISO-8859-1 etc. In such character sets if the eight bit is set it indicates an extended character. Ruby allows method names and other identifiers to contain such characters. Ruby programs cannot contain some characters like ASCII NUL (`<code>x00</code>`).

The following are the examples of valid ruby methods:

```
def hello
  "hello"
end

def こんにちは
  puts "means hello in Japanese"
end
```

Typically method names are US-ASCII compatible since the keys to type them exist on all keyboards.

Method names may end with a `!` (bang or exclamation mark), a `?` (question mark) or `=` equals sign.

The bang methods(! at the end of method name) are called and executed just like any other method. However, by convention, a method with an exclamation point or bang is considered dangerous. In ruby core library the dangerous method implies that when a method ends with a bang(!), it indicates that unlike its non-bang equivalent, permanently modifies its receiver. Almost always, Ruby core library will have a non-bang counterpart(method name which does NOT end with !) of every bang method (method name which does end with !) that has does not modify the receiver. This convention is typically true for ruby core library but may/may not hold true for other ruby libraries.

Methods that end with a question mark by convention return boolean. But they may not always return just `true` or `false`. Often they will may return an object to indicate a true value (or “truthy” value).

Methods that end with an equals sign indicate an assignment method. For assignment methods the return value is ignored, the arguments are returned instead.

These are method names for the various ruby operators. Each of these operators accept only one argument. Following the operator is the typical use or name of the operator. Creating an alternate meaning for the operator may lead to confusion as the user expects plus to add things, minus to subtract things, etc. Additionally, you cannot alter the precedence of the operators.

+

add

-

subtract

*

multiply

**

power

/

divide

%

modulus division, String#%

&

AND

^

XOR (exclusive OR)

>>

right-shift

<<

left-shift, append

==

equal

!=

not equal

===

case equality. See Object#===

=~

pattern match. (Not just for regular expressions)

!~

does not match

<=>

comparison aka spaceship operator. See Comparable

<

less-than

<=

less-than or equal

>

greater-than

>=

greater-than or equal

To define unary methods minus, plus, tilde and not (!)
follow the operator with an @ as in +@ or !@:

```
class C
  def -@
    puts "you inverted this object"
  end
end

obj = C.new

-obj # prints "you inverted this object"
```

Unary methods accept zero arguments.

Additionally, methods for element reference and assignment may be defined: `[]` and `[]=` respectively. Both can take one or more arguments, and element reference can take none.

```
class C
  def [](a, b)
    puts a + b
  end

  def []=(a, b, c)
    puts a * b + c
  end
end

obj = C.new

obj[2, 3] # prints "5"
obj[2, 3] = 4 # prints "10"
```

Return Values

By default, a method returns the last expression that was evaluated in the body of the method. In the example above, the last (and only) expression evaluated was the simple sum `1 + 1`. The `return` keyword can be used to make it explicit that a method returns a value.

```
def one_plus_one
  return 1 + 1
end
```

It can also be used to make a method return before the last expression is evaluated.

```
def two_plus_two
  return 2 + 2
  1 + 1 # this expression is never evaluated
end
```

Note that for assignment methods the return value will always be ignored. Instead the argument will be returned:

```
def a=(value)
  return 1 + value
end

p(a = 5) # prints 5
```

Scope

The standard syntax to define a method:

```
def my_method
  # ...
end
```

adds the method to a class. You can define an instance method on a specific class with the `class` keyword:

```
class C
  def my_method
    # ...
  end
end
```

A method may be defined on another object. You may define a “class method” (a method that is defined on the class, not an instance of the class) like this:

```
class C
  def self.my_method
    # ...
  end
end
```

However, this is simply a special case of a greater syntactical power in Ruby, the ability to add methods to any object. Classes are objects, so adding class methods is simply adding methods to the Class object.

The syntax for adding a method to an object is as follows:

```
greeting = "Hello"
def greeting.broaden
```

```
    self + ", world!"  
end  
  
greeting.broaden # returns "Hello, world!"
```

`self` is a keyword referring to the current object under consideration by the compiler, which might make the use of `self` in defining a class method above a little clearer. Indeed, the example of adding a `hello` method to the class `String` can be rewritten thus:

```
def String.hello  
  "Hello, world!"  
end
```

A method defined like this is called a “singleton method”. `broaden` will only exist on the string instance `greeting`. Other strings will not have `broaden`.

Overriding

When Ruby encounters the `def` keyword, it doesn't consider it an error if the method already exists: it simply redefines it. This is called *overriding*. Rather like extending core classes, this is a potentially dangerous ability, and should be used sparingly because it can cause unexpected results. For example, consider this irb session:

```
>> "43".to_i
=> 43
>> class String
>>   def to_i
>>     42
>>   end
>> end
=> nil
>> "43".to_i
=> 42
```

This will effectively sabotage any code which makes use of the method `String#to_i` to parse numbers from strings.

Arguments

A method may accept arguments. The argument list follows the method name:

```
def add_one(value)
  value + 1
end
```

When called, the user of the `add_one` method must provide an argument. The argument is a local variable in the method body. The method will then add one to this argument and return the value. If given `1` this method will return `2`.

The parentheses around the arguments are optional:

```
def add_one value
  value + 1
end
```

Multiple arguments are separated by a comma:

```
def add_values(a, b)
  a + b
end
```

When called, the arguments must be provided in the exact order. In other words, the arguments are positional.

Default Values

Arguments may have default values:

```
def add_values(a, b = 1)
  a + b
end
```

```
end
```

The default value does not need to appear first, but arguments with defaults must be grouped together. This is ok:

```
def add_values(a = 1, b = 2, c)
  a + b + c
end
```

This will raise a `SyntaxError`:

```
def add_values(a = 1, b, c = 1)
  a + b + c
end
```

Array Decomposition

You can decompose (unpack or extract values from) an Array using extra parentheses in the arguments:

```
def my_method((a, b))
  p a: a, b: b
end

my_method([1, 2])
```

This prints:

```
{:a=>1, :b=>2}
```

If the argument has extra elements in the Array they will be ignored:

```
def my_method((a, b))
  p a: a, b: b
end
```

```
my_method([1, 2, 3])
```

This has the same output as above.

You can use a `*` to collect the remaining arguments. This splits an Array into a first element and the rest:

```
def my_method((a, *b))
  p a: a, b: b
end

my_method([1, 2, 3])
```

This prints:

```
{:a=>1, :b=>[2, 3]}
```

The argument will be decomposed if it responds to `to_ary`. You should only define `to_ary` if you can use your object in place of an Array.

Use of the inner parentheses only uses one of the sent arguments. If the argument is not an Array it will be assigned to the first argument in the decomposition and the remaining arguments in the decomposition will be `nil`:

```
def my_method(a, (b, c), d)
  p a: a, b: b, c: c, d: d
end

my_method(1, 2, 3)
```

This prints:

```
{:a=>1, :b=>2, :c=>nil, :d=>3}
```

You can nest decomposition arbitrarily:

```
def my_method((a, b), c)
  # ...
end
```

Array/Hash Argument

Prefixing an argument with * causes any remaining arguments to be converted to an Array:

```
def gather_arguments(*arguments)
  p arguments
end

gather_arguments 1, 2, 3 # prints [1, 2, 3]
```

The array argument must be the last positional argument, it must appear before any keyword arguments.

The array argument will capture a Hash as the last entry if a hash was sent by the caller after all positional arguments.

```
gather_arguments 1, a: 2 # prints [1, {:a=>2}]
```

However, this only occurs if the method does not declare any keyword arguments.

```
def gather_arguments_keyword(*positional, keyword: nil)
  p positional: positional, keyword: keyword
end

gather_arguments_keyword 1, 2, three: 3
#=> raises: unknown keyword: three (ArgumentError)
```

Also, note that a bare * can be used to ignore arguments:

```
def ignore_arguments(*)
```

```
end
```

Keyword Arguments

Keyword arguments are similar to positional arguments with default values:

```
def add_values(first: 1, second: 2)
  first + second
end
```

Arbitrary keyword arguments will be accepted with **:

```
def gather_arguments(first: nil, **rest)
  p first, rest
end

gather_arguments first: 1, second: 2, third: 3
# prints 1 then {:second=>2, :third=>3}
```

When calling a method with keyword arguments the arguments may appear in any order. If an unknown keyword argument is sent by the caller an `ArgumentError` is raised.

When mixing keyword arguments and positional arguments, all positional arguments must appear before any keyword arguments.

Block Argument

The block argument is indicated by `&` and must come last:

```
def my_method(&my_block)
  my_block.call(self)
end
```

Most frequently the block argument is used to pass a block to another method:

```
def each_item(&block)
  @items.each(&block)
end
```

If you are only going to call the block and will not otherwise manipulate it or send it to another method using `yield` without an explicit block parameter is preferred. This method is equivalent to the first method in this section:

```
def my_method
  yield self
end
```

There is also a performance benefit to using `yield` over a calling a block parameter. When a block argument is assigned to a variable a Proc object is created which holds the block. When using `yield` this Proc object is not created.

If you only need to use the block sometimes you can use `Proc.new` to create a proc from the block that was passed to your method. See `Proc.new` for further details.

Exception Handling

Methods have an implied exception handling block so you do not need to use `begin` or `end` to handle exceptions.

This:

```
def my_method
  begin
    # code that may raise an exception
  rescue
    # handle exception
  end
end
```

May be written as:

```
def my_method
  # code that may raise an exception
rescue
  # handle exception
end
```

If you wish to rescue an exception for only part of your method use `begin` and `end`. For more details see the page on exception handling

Miscellaneous Syntax

Ending an Expression

Ruby uses a newline as the end of an expression. When ending a line with an operator, open parentheses, comma, etc. the expression will continue.

You can end an expression with a ; (semicolon). Semicolons are most frequently used with `ruby -e`.

Indentation

Ruby does not require any indentation. Typically ruby programs are indented two spaces.

If you run ruby with warnings enabled and have an indentation mis-match you will receive a warning.

alias

The `alias` keyword is most frequently used to alias methods. When aliasing a method you can use either its name or a symbol:

```
alias new_name old_name  
alias :new_name :old_name
```

For methods, `Module#alias_method` can often be used instead of `alias`.

You can also use `alias` to alias global variables:

```
$old = 0  
  
alias $new $old  
  
p $new # prints 0
```

You may use `alias` in any scope.

undef

The `undef` keyword prevents the current class from responding to calls to the named methods.

```
undef my_method
```

You may use symbols instead of method names:

```
undef :my_method
```

You may undef multiple methods:

```
undef method1, method2
```

You may use `undef` in any scope. See also `Module#undef_method`

defined?

`defined?` is a keyword that returns a string describing its argument:

```
p defined?(UNDEFINED_CONSTANT) # prints nil
p defined?(RUBY_VERSION)       # prints "constant"
p defined?(1 + 1)               # prints "method"
```

You don't need to use parenthesis with `defined?` but they are recommended due to the low precedence of `defined?`.

For example, if you wish to check if an instance variable exists and that the instance variable is zero:

```
defined? @instance_variable && @instance_variable.zero?
```

This returns "expression" which is not what you want if the instance variable is not defined.

```
@instance_variable = 1
defined?(@instance_variable) && @instance_variable.zero?
```

Adding parentheses when checking if the instance variable is defined is a better check. This correctly returns `nil` when the instance variable is not defined and `false` when the instance variable is not zero.

Using the specific reflection methods such as `instance_variable_defined?` for instance variables or `const_defined?` for constants is less error prone than using `defined?`.

BEGIN and END

`BEGIN` defines a block that is run before any other code in the current file. It is typically used in one-liners with `ruby -e`. Similarly `END` defines a block that is run after any other code.

`BEGIN` must appear at top-level and `END` will issue a warning when you use it inside a method.

Here is an example:

```
BEGIN {  
  count = 0  
}
```

You must use `{` and `}` you may not use `do` and `end`.

Here is an example one-liner that adds numbers from standard input or any files in the argument list:

```
ruby -ne 'BEGIN { count = 0 }; END { puts count }; co
```

Modules

Modules serve two purposes in Ruby, namespacing and mix-in functionality.

A namespace can be used to organize code by package or functionality that separates common names from interference by other packages. For example, the IRB namespace provides functionality for irb that prevents a collision for the common name “Context”.

Mix-in functionality allows sharing common methods across multiple classes or modules. Ruby comes with the Enumerable mix-in module which provides many enumeration methods based on the `each` method and Comparable allows comparison of objects based on the `<=>` comparison method.

Note that there are many similarities between modules and classes. Besides the ability to mix-in a module, the description of modules below also applies to classes.

Module Definition

A module is created using the `module` keyword:

```
module MyModule
  # ...
end
```

A module may be reopened any number of times to add, change or remove functionality:

```
module MyModule
  def my_method
  end
end

module MyModule
  alias my_alias my_method
end

module MyModule
  remove_method :my_method
end
```

Reopening classes is a very powerful feature of Ruby, but it is best to only reopen classes you own. Reopening classes you do not own may lead to naming conflicts or difficult to diagnose bugs.

Nesting

Modules may be nested:

```
module Outer
  module Inner
  end
end
```

Many packages create a single outermost module (or class) to provide a namespace for their functionality.

You may also define inner modules using `::` provided the outer modules (or classes) are already defined:

```
module Outer::Inner::GrandChild
end
```

Note that this will raise a `NameError` if `Outer` and `Outer::Inner` are not already defined.

This style has the benefit of allowing the author to reduce the amount of indentation. Instead of 3 levels of indentation only one is necessary. However, the scope of constant lookup is different for creating a namespace using this syntax instead of the more verbose syntax.

Scope

self

`self` refers to the object that defines the current scope. `self` will change when entering a different method or when defining a new module.

Constants

Accessible constants are different depending on the module nesting (which syntax was used to define the module). In the following example the constant `A::z` is accessible from `B` as `A` is part of the nesting:

```
module A
  Z = 1

  module B
    p Module.nesting #=> [A::B, A]
    p Z #=> 1
  end
end
```

However, if you use `::` to define `A::B` without nesting it inside `A` a `NameError` exception will be raised because the nesting does not include `A`:

```
module A
  Z = 1
end

module A::B
  p Module.nesting #=> [A::B]
  p Z #=> raises NameError
end
```

If a constant is defined at the top-level you may precede it with `::` to reference it:

```
Z = 0

module A
  Z = 1

  module B
    p ::Z ==> 0
  end
end
```

Methods

For method definition documentation see the syntax documentation for methods

Class methods may be called directly. (This is slightly confusing, but a method on a module is often called a “class method” instead of a “module method”. See also `Module#module_function` which can convert an instance method into a class method.)

When a class method references a constant it uses the same rules as referencing it outside the method as the scope is the same.

Instance methods defined in a module are only callable when included. These methods have access to the constants defined when they were included through the ancestors list:

```
module A
  Z = 1

  def z
    Z
  end
end
```

```
include A

p self.class.ancestors #=> [Object, A, Kernel, BasicO
p z #=> 1
```

Visibility

Ruby has three types of visibility. The default is `public`. A public method may be called from any other object.

The second visibility is `protected`. When calling a protected method the sender must be a subclass of the receiver or the receiver must be a subclass of the sender. Otherwise a `NoMethodError` will be raised.

Protected visibility is most frequently used to define `==` and other comparison methods where the author does not wish to expose an object's state to any caller and would like to restrict it only to inherited classes.

Here is an example:

```
class A
  def n(other)
    other.m
  end
end

class B < A
  def m
    1
  end

  protected :m
end

class C < B
end
```

```
a = A.new
b = B.new
c = C.new

c.n b #=> 1 -- C is a subclass of B
b.n b #=> 1 -- m called on defining class
a.n b # raises NoMethodError A is not a subclass of B
```

The third visibility is `private`. A private method may not be called with a receiver, not even `self`. If a private method is called with a receiver a `NoMethodError` will be raised.

alias and undef

You may also alias or undefine methods, but these operations are not restricted to modules or classes. See the miscellaneous syntax section for documentation.

Classes

Every class is also a module, but unlike modules a class may not be mixed-in to another module (or class). Like a module, a class can be used as a namespace. A class also inherits methods and constants from its superclass.

Defining a class

Use the `class` keyword to create a class:

```
class MyClass
  # ...
end
```

If you do not supply a superclass your new class will inherit from `Object`. You may inherit from a different class using `<` followed by a class name:

```
class MySubclass < MyClass
  # ...
end
```

There is a special class `BasicObject` which is designed as a blank class and includes a minimum of built-in methods. You can use `BasicObject` to create an independent inheritance structure. See the `BasicObject` documentation for further details.

Inheritance

Any method defined on a class is callable from its subclass:

```
class A
  Z = 1

  def z
    Z
  end
end

class B < A
end

p B.new.z #=> 1
```

The same is true for constants:

```
class A
  Z = 1
end

class B < A
  def z
    Z
  end
end

p B.new.z #=> 1
```

You can override the functionality of a superclass method by redefining the method:

```
class A
  def m
    1
  end
end
```



```
class B < A
  def m
    2
  end
end

p B.new.m #=> 2
```

If you wish to invoke the superclass functionality from a method use `super`:

```
class A
  def m
    1
  end
end

class B < A
  def m
    2 + super
  end
end

p B.new.m #=> 3
```

When used without any arguments `super` uses the arguments given to the subclass method. To send no arguments to the superclass method use `super()`. To send specific arguments to the superclass method provide them manually like `super(2)`.

`super` may be called as many times as you like in the subclass method.

Singleton Classes

The singleton class (also known as the metaclass or eigenclass) of an object is a class that holds methods for only that instance. You can access the singleton class of an object using `class << object` like this:

```
class C
end

class << C
  # self is the singleton class here
end
```

Most frequently you'll see the singleton class accessed like this:

```
class C
  class << self
    # ...
  end
end
```

This allows definition of methods and attributes on a class (or module) without needing to write `def self.my_method`.

Since you can open the singleton class of any object this means that this code block:

```
o = Object.new

def o.my_method
  1 + 1
end
```

is equivalent to this code block:

```
o = Object.new

class << o
  def my_method
    1 + 1
  end
end
```

Both objects will have a `my_method` that returns 2.

Generated by [RDoc 3.12.2](#).

Generated with the [Darkfish Rdoc Generator 3](#).

Precedence

From highest to lowest, this is the precedence table for ruby. High precedence operations happen before low precedence operations.

```
!, ~, unary +  
  
**  
  
unary -  
  
*, /, %  
  
+, -  
  
<<, >>  
  
&  
  
|, ^  
  
>, >=, <, <=  
  
<=>, ==, ===, !=, =~, !~  
  
&&  
  
||  
  
.., ...  
  
?, :  
  
modifier-rescue  
  
=, +=, -=, etc.  
  
defined?  
  
not
```

```
or, and  
modifier-if, modifier-unless, modifier-while, modifie  
{ } blocks
```

Unary + and unary - are for +1, -1 or -(a + b).

Modifier-if, modifier-unless, etc. are for the modifier versions of those keywords. For example, this is a modifier-unless expression:

```
a += 1 unless a.zero?
```

{ ... } blocks have priority below all listed operations, but do ... end blocks have lower priority.

All other words in the precedence table above are keywords.

Generated by [RDoc](#) 3.12.2.

Generated with the [Darkfish Rdoc Generator](#) 3.

Refinements

Due to Ruby's open classes you can redefine or add functionality to existing classes. This is called a “monkey patch”. Unfortunately the scope of such changes is global. All users of the monkey-patched class see the same changes. This can cause unintended side-effects or breakage of programs.

Refinements are designed to reduce the impact of monkey patching on other users of the monkey-patched class. Refinements provide a way to extend a class locally.

Here is a basic refinement:

```
class C
  def foo
    puts "C#foo"
  end
end

module M
  refine C do
    def foo
      puts "C#foo in M"
    end
  end
end
```

First, a class `c` is defined. Next a refinement for `c` is created using `Module#refine`. Refinements only modify classes, not modules so the argument must be a class.

`Module#refine` creates an anonymous module that contains the changes or refinements to the class (`c` in the example). `self` in the refine block is this anonymous

module similar to `Module#module_eval`.

Activate the refinement with using:

```
using M  
c = C.new  
c.foo # prints "C#foo in M"
```

Scope

You may only activate refinements at top-level, not inside any class, module or method scope. You may activate refinements in a string passed to `Kernel#eval` that is evaluated at top-level. Refinements are active until the end of the file or the end of the eval string, respectively.

Refinements are lexical in scope. When control is transferred outside the scope the refinement is deactivated. This means that if you require or load a file or call a method that is defined outside the current scope the refinement will be deactivated:

```
class C
end

module M
  refine C do
    def foo
      puts "C#foo in M"
    end
  end
end

def call_foo(x)
  x.foo
end

using M

x = C.new
x.foo # prints "C#foo in M"
call_foo(x) #=> raises NoMethodError
```

If a method is defined in a scope where a refinement is active the refinement will be active when the method is called. This example spans multiple files:

c.rb:

```
class C
end
```

m.rb:

```
require "c"

module M
  refine C do
    def foo
      puts "C#foo in M"
    end
  end
end
```

m_user.rb:

```
require "m"

using M

class MUser
  def call_foo(x)
    x.foo
  end
end
```

main.rb:

```
require "m_user"

x = C.new
m_user = MUser.new
m_user.call_foo(x) # prints "C#foo in M"
x.foo              #=> raises NoMethodError
```

Since the refinement `M` is active in `m_user.rb` where `MUser#call_foo` is defined it is also active when `main.rb` calls `call_foo`.

Since using is a method, refinements are only active when it is called. Here are examples of where a refinement `M` is and is not active.

In a file:

```
# not activated here
using M
# activated here
class Foo
  # activated here
  def foo
    # activated here
  end
  # activated here
end
# activated here
```

In eval:

```
# not activated here
eval <<EOF
  # not activated here
  using M
  # activated here
EOF
# not activated here
```

When not evaluated:

```
# not activated here
if false
  using M
end
# not activated here
```

When defining multiple refinements in the same module, inside a refine block all refinements from the same module are active when a refined method is called:

```
module ToJSON
```

```

refine Integer do
  def to_json
    to_s
  end
end

refine Array do
  def to_json
    "[" + map { |i| i.to_json }.join(",") + "]"
  end
end

refine Hash do
  def to_json
    "{" + map { |k, v| k.to_s.dump + ":" + v.to_json }.join(",") + "}"
  end
end

using ToJSON

p [{1=>2}, {3=>4}].to_json # prints "[{"1":2},{3:4}]"

```

You may also activate refinements in a class or module definition, in which case the refinements are activated from the point where using is called to the end of the class or module definition:

```

# not activated here
class Foo
  # not activated here
  using M
  # activated here
  def foo
    # activated here
  end
  # activated here
end
# not activated here

```

Note that the refinements in M are not activated automatically even if the class Foo is reopened later.

Method Lookup

When looking up a method for an instance of class `c`
Ruby checks:

- If refinements are active for `c`, in the reverse order they were activated:
 - The prepended modules from the refinement for `c`
 - The refinement for `c`
 - The included modules from the refinement for `c`
- The prepended modules of `c`
- `c`
- The included modules of `c`

If no method was found at any point this repeats with the superclass of `c`.

Note that methods in a subclass have priority over refinements in a superclass. For example, if the method `/` is defined in a refinement for `Integer` `1 / 2` invokes the original `Fixnum#` because `Fixnum` is a subclass of `Integer` and is searched before the refinements for the superclass `Integer`.

If a method `f00` is defined on `Integer` in a refinement, `1.f00` invokes that method since `f00` does not exist on `Fixnum`.

super

When `super` is invoked method lookup checks:

- The included modules of the current class. Note that the current class may be a refinement.
- If the current class is a refinement, the method lookup proceeds as in the Method Lookup section above.
- If the current class has a direct superclass, the method proceeds as in the Method Lookup section above using the superclass.

Note that `super` in a method of a refinement invokes the method in the refined class even if there is another refinement which has been activated in the same context.

Indirect Method Calls

When using indirect method access such as `Kernel#send`, `Kernel#method` or `Kernel#respond_to?` refinements are not honored for the caller context during method lookup.

This behavior may be changed in the future.

Refinements and module inclusion

Refinements are inherited by module inclusion. That is, using activates all refinements in the ancestors of the specified module. Refinements in a descendant have priority over refinements in an ancestor.

Further Reading

See bugs.ruby-lang.org/projects/ruby-trunk/wiki/RefinementsSpec for the current specification for implementing refinements. The specification also contains more details.

Generated by [RDoc](#) 3.12.2.

Generated with the [Darkfish Rdoc Generator](#) 3.