

Class **Exception** < Object

Descendents of class `Exception` are used to communicate between `raise` methods and `rescue` statements in `begin/end` blocks. `Exception` objects carry information about the exception—its type (the exception’s class name), an optional descriptive string, and optional traceback information.

1.9 The standard library defines the exceptions shown in Figure 27.1 on the following page. Note that Ruby 1.9 has changed the hierarchy slightly: `SecurityError` is no longer a subclass of `StandardError` and so will not be rescued implicitly. See also the description of `Errno` on the previous page.

Class methods

exception `Exception.exception(< message >) → exc`

Creates and returns a new exception object, optionally setting the message to *message*.

new `Exception.new(< message >) → exc`

Creates and returns a new exception object, optionally setting the message to *message*.

Instance methods

backtrace `exc.backtrace → array`

Returns any backtrace associated with the exception. The backtrace is an array of strings, each containing either *filename:line: in 'method'* or *filename:line*.

```
def a
  raise "boom"
end
def b
  a()
end
begin
  b()
rescue => detail
  print detail.backtrace.join("\n")
end
```

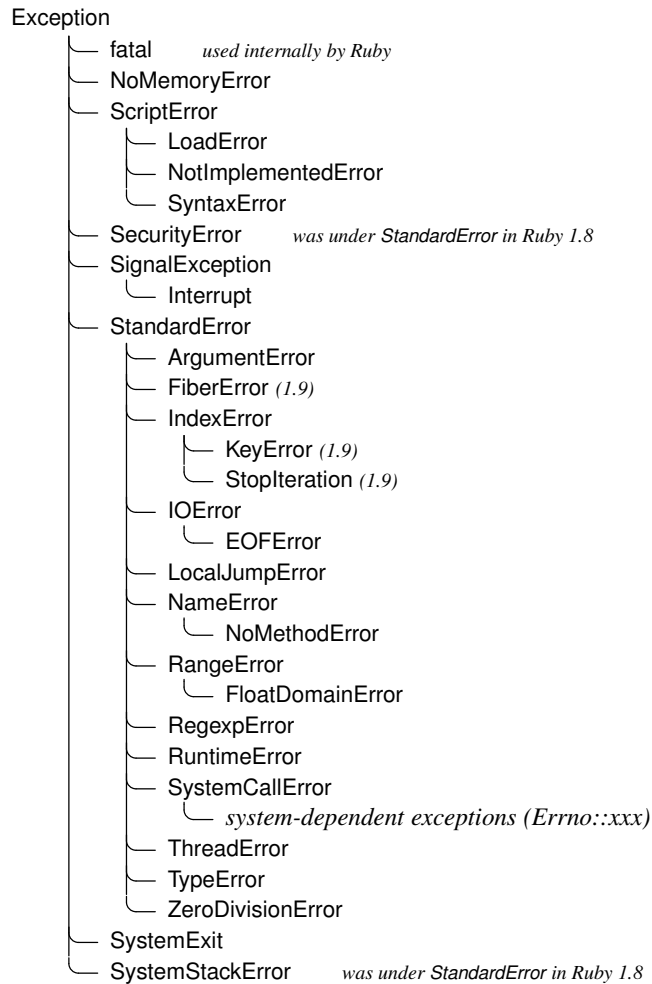
produces:

```
/tmp/prog.rb:2:in `a'
/tmp/prog.rb:6:in `b'
/tmp/prog.rb:10:in `'
```

exception `exc.exception(< message >) → exc or exception`

With no argument, returns the receiver. Otherwise, creates a new exception object of the same class as the receiver but with a different message.

Figure 27.1. Standard Exception Hierarchy



message *exc.message* → *msg*

Returns the message associated with this exception.

set_backtrace *exc.set_backtrace(array)* → *array*

Sets the backtrace information associated with *exc*. The argument must be an array of String objects in the format described in `Exception#backtrace`.

status *exc.status* → *status*

1.9 / (SystemExit only) Returns the exit status associated with this SystemExit exception. Normally this status is set using the `Kernel#exit`.

```
begin
  exit(99)
rescue SystemExit => e
  puts "Exit status is: #{e.status}"
end
```

produces:

Exit status is: 99

success? *exc.success?* → true or false

1.9 / (SystemExit only) Returns true is the exit status if nil or zero.

```
begin
  exit(99)
rescue SystemExit => e
  print "This program "
  if e.success?
    print "did"
  else
    print "did not"
  end
  puts " succeed"
end
```

produces:

This program did not succeed

to_s *exc.to_s* → *msg*

Returns the message associated with this exception (or the name of the exception if no message is set).

```
begin
  raise "The message"
rescue Exception => e
  puts e.to_s
end
```

produces:

The message