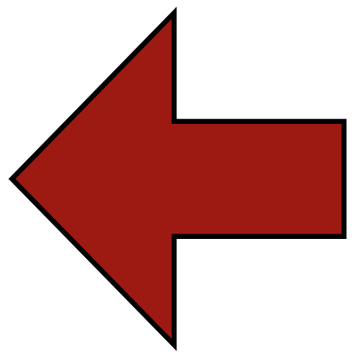
nonblocking I/O without exceptions

Feature #5138

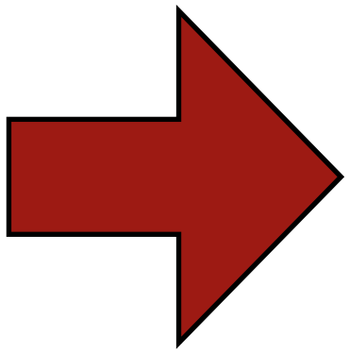
Common I/O cases should not require exceptions for flow control

```
def rbuf_fill
  begin
    @rbuf << @io.read_nonblock(BUFSIZE)
  rescue IO::WaitReadable
    return retry if IO.select([@io], nil, nil, @read_timeout)
    raise Timeout::Error
  rescue IO::WaitWritable
    return retry if IO.select(nil, [@io], nil, @read_timeout)
    raise Timeout::Error
  end
end
```



Today

What I want



```
def rbuf_fill
  case value = @io.try_read_nonblock(BUFSIZE)
  when :read_would_block
    return rbuf_fill if IO.select([@io], nil, nil, @read_timeout)
    raise Timeout::Error
  when :write_would_block
    return rbuf_fill if IO.select(nil, [@io], nil, @read_timeout)
    raise Timeout::Error
  when String
    @rbuf << value
  end
end
```

Why?

Avoid the creation hundreds (if not thousands) of exception (including their backtraces) that are immediately rescued in the course of a typical non-blocking socket read or write loop.

- ogmomg