Problems manipulating the signal mask with Ruby

The futex timer thread inherits the (nearly empty) signal mask from the main thread when it is created. I've written a little Ruby library that executes sigprocmask/pthread_sigmask using FFI. However, my Ruby program isn't able to block signals since they are instead delivered to the timer thread.

Also, calls to rb_enable_interrupt() unmask all signal types. Instead, I'd like Ruby to remember the signal mask I have set.

The attached patch, against 1.9.1-p376, makes rb_disable_interrupt() save the old signal mask for rb_enable_interrupt() to use as well as masking all signals as soon as the timer thread is created. Finally, it removes the call to rb_trap_restore_mask() from rb_longjmp() - this appeared to be unnecessary. Is it?

My testing hasn't shown any problems with this, but I'd love to hear any thoughts about this.

I'm so sorry for absence from this ticket.

From Ruby 1.9.3, we unmask all mask of all threads (main thread, timer thread and all other threads) to accept signals. Generally speaking, Kernel.trap() should be used for signal manipulation in Ruby world.

However, I assume you have special reason why you need to specify signal mask in manual. Could we discuss about it and specify desirable specification?

No feedback.