Ruby - Bug #5957

pthreads not working on ulibc (linuxthreads)

02/01/2012 10:58 PM - stevegoobermanhill (stephen gooberman-hill)

Status:	Rejected		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 1.9.2p290 (2011-07-09 revision 32553) [arm-linux]	**Backport:**	

Description

Hi
I have an arm-linux crosscompile of ruby 1.9.2p290 and 1.9.3p0

Trying to implement a thread (via Thrad.new) causes an error to be thrown
"ruby engine can initialize only in the main thread"

and a stack trace is thrown.

Tracing this issue (in 1.9.2p290) leads to thread_pthread.c lines 350-388

```
static int
native_thread_init_stack(rb_thread_t *th)
{
rb_thread_id_t curr = pthread_self();

if (pthread_equal(curr, native_main_thread.id)) {
th->machine_stack_start = native_main_thread.stack_start;
th->machine_stack_maxsize = native_main_thread.stack_maxsize;
}
else {

#ifdef STACKADDR_AVAILABLE
void *start;
size_t size;

if (get_stack(&start, &size) == 0) {
    th->machine_stack_start = start;
    th->machine_stack_maxsize = size;
}

#else
rb_raise(rb_eNotImpError, "ruby engine can initialize only in the main thread");
#endif
}
#ifdef __ia64
th->machine_register_stack_start = native_main_thread.register_stack_start;
th->machine_stack_maxsize /= 2;
th->machine_register_stack_maxsize = th->machine_stack_maxsize;
#endif
return 0;
}
```

I have done some digging, comparing by build machine on which I run ruby1.9.2p290 built from source, and my arm-linux target machine

STACKADDR_AVAILABLE is defined based on a whole set of possible conditions higher in the code (lines 221-229)
#if defined HAVE_PTHREAD_GETATTR_NP || defined HAVE_PTHREAD_ATTR_GET_NP
#define STACKADDR_AVAILABLE 1
#elif defined HAVE_PTHREAD_GET_STACKADDR_NP && defined HAVE_PTHREAD_GET_STACKSIZE_NP
#define STACKADDR_AVAILABLE 1
#elif defined HAVE_THR_STKSEGMENT || defined HAVE_PTHREAD_STACKSEG_NP

#define STACKADDR_AVAILABLE 1
#elif defined HAVE_PTHREAD_GETTHRDS_NP
#define STACKADDR_AVAILABLE 1
#endif

I have done some greping around /usr/includes, and I can't find any of these definitions.

So I assume that on my i686 build system ruby is running on the native_main_thread, but on the arm system it is not.

Is this the case. Any ideas on how to persuade ruby to run on the native main thread. I can probably rewrite a few critical sections of my app to run of fibers if necessary....but I'd prefer to have thread support

Kind regards

Steve G-H

Related issues:

Related to Ruby - Bug #6358: arm-linux : sleep() time dependent threading bug	**Closed**	**04/25/2012**

History

#1 - 02/02/2012 05:30 AM - kosaki (Motohiro KOSAKI)

- Status changed from Open to Rejected

I'm sorry. We have no plan to support linuxthreads any more. Core developers can't access linuxthreads platform easily and I don't think we can get great contributor in near futur in this area. So, I don't think the supporting is practical option. Please use fiber. Again, I'm very sorry.

#2 - 02/03/2012 06:04 AM - stevegoobermanhill (stephen gooberman-hill)

Hi Motohiro,
thanks for taking a look :-)

I've been doing some more digging and I think my initial thoughts were wrong. I am now pretty sure that this is actually an issue around the uClibc build on my ARM board - basically it has been built with pretty much every possible pre-processor directive switched off, so it is not supporting sufficient functionality to enable pthreads to run sucessfully (pthread_getattr_np is defined in a #ifdef block but not built).

I'm in touch with the board manufacturers (Techbase - it is an NPE series industrial computer / gsm/gprs/edge modem). If I can get ruby running (with thread support - as I need TCPServer) then I am looking at bulk orders - so they are (at the moment) being very helpful.

Kind regards

Steve