Ruby master - Bug #5239

bootstraptest/runner.rb: assert_normal_exit logic broken on Debian/GNU kFreeBSD

08/28/2011 02:02 AM - lucas (Lucas Nussbaum)

Status:	Rejected		
Priority:	Normal		
Assignee:	kosaki (Motohiro KOSAKI)		
Target version:	1.9.3		
ruby -v:	-	**Backport:**	

Description

Hi,

assert_normal_exit() breaks on some platforms, such as Debian GNU/kFreeBSD.

It does:

```
begin
  $stderr.reopen("assert_normal_exit.log", "w")
  io = IO.popen("#{@ruby} -W0 #{filename}")
  pid = io.pid
  th = Thread.new {
    io.read
    io.close
    $?
  }
  if !th.join(timeout)
    Process.kill :KILL, pid
    timeout_signaled = true
  end
  status = th.value
ensure
  $stderr.reopen(old_stderr)
  old_stderr.close
end
if status.signaled?
```

The problem is that the call to $? in the thread cannot retrieve the exit value of the process started by popen. "$?" is transformed into a wait4() syscall, but wait4 is not allowed to inquire the state of non-child processes. And the popen process is not a child of the sub-thread, it is a child of the main thread.

http://pubs.opengroup.org/onlinepubs/009695399/functions/wait.html confirms that waitpid() is not supposed to work on non-child processes.

It works on Linux because wait4() is friendlier there and accepts to give the state of non-child processes.

This was introduced in svn revision 26700, when a timeout was added.

History

#1 - 08/28/2011 02:13 AM - lucas (Lucas Nussbaum)

- File assert_normal_exit.patch added

The attached patch fixes this. Could you consider applying to 1.9.3 too?

Thanks

#2 - 08/28/2011 02:30 AM - kosaki (Motohiro KOSAKI)

- Status changed from Open to Assigned

- Assignee set to kosaki (Motohiro KOSAKI)

- Target version set to 1.9.3

> The problem is that the call to $? in the thread cannot retrieve the exit value of the process started by popen. "$?" is transformed into a wait4() syscall, but wait4 is not allowed to inquire the state of non-child processes. And the popen process is not a child of the sub-thread, it is a child of the main thread.
>
> http://pubs.opengroup.org/onlinepubs/009695399/functions/wait.html confirms that waitpid() is not supposed to work on non-child processes.

In fact, it's a child process. see more below paragraph.

> RATIONALE
> A call to the wait() or waitpid() function only returns status on an immediate child process of the *calling process*

The pthread rule is, pthread_create() doen't introduce any process tree change. In the other words, kFreeBSD is buggy.

But I'm planning to commit your patch because 1) it's only test case change and no impact to ruby code base 2) your patch doesn't introduce any ugliness.

#3 - 08/28/2011 06:29 AM - lucas (Lucas Nussbaum)

On 28/08/11 at 02:30 +0900, Motohiro KOSAKI wrote:

> Issue #5239 has been updated by Motohiro KOSAKI.
>
> Status changed from Open to Assigned
> Assignee set to Motohiro KOSAKI
> Target version set to 1.9.3
>
> > The problem is that the call to $? in the thread cannot retrieve the exit value of the process started by popen. "$?" is transformed into a wait4() syscall, but wait4 is not allowed to inquire the state of non-child processes. And the popen process is not a child of the sub-thread, it is a child of the main thread.
> >
> > http://pubs.opengroup.org/onlinepubs/009695399/functions/wait.html confirms that waitpid() is not supposed to work on non-child processes.
>
> In fact, it's a child process. see more below paragraph.
>
> > RATIONALE
> > A call to the wait() or waitpid() function only returns status on an immediate child process of the *calling process*
>
> The pthread rule is, pthread_create() doen't introduce any process tree change. In the other words, kFreeBSD is buggy.

One could argue that a different PID makes a different process, but it's
correct that it's not a different process.

> But I'm planning to commit your patch because 1) it's only test case change and no impact to ruby code base 2) your patch doesn't introduce any ugliness.

Thanks

Lucas

#4 - 08/28/2011 09:53 AM - kosaki (Motohiro KOSAKI)

- ruby -v changed from 1.9.3 to -

> > RATIONALE
> > A call to the wait() or waitpid() function only returns status on an immediate child process of the *calling process*
>
> The pthread rule is, pthread_create() doen't introduce any process tree change. In the other words, kFreeBSD is buggy.

> One could argue that a different PID makes a different process, but it's
> correct that it's not a different process.

popen makes different process. and Thread.new don't make a process. isn't it?
Therefore a subthread clealy can use waitpid() for waiting child
process. Moreover
real FreeBSD works this test completely. It's only kFreeBSD issue.

I wonder why kFreeBSD don't handle threading correctly. Long time ago,
linuxthreads made the same mistake and application users sufferd from it.
I didn't expected we observed the same issue at 21th century.

#5 - 08/29/2011 01:18 AM - lucas (Lucas Nussbaum)

Hi,

I've just checked, and FreeBSD 8.2 is also affected by this issue.
Test script:
system("false")
sleep 0.5
th = Thread::new { $? }
p th
th.join
p th
p th.value

#6 - 08/29/2011 01:59 PM - lucas (Lucas Nussbaum)

On 29/08/11 at 12:43 +0900, KOSAKI Motohiro wrote:

> > I've just checked, and FreeBSD 8.2 is also affected by this issue.
> > Test script:
> > system("false")
> > sleep 0.5
> > th = Thread::new { $? }
> > p th
> > th.join
> > p th
> > p th.value
>
> This is buggy. $? is thread local storage variable. last exit value
> was set to main thread.
> because system() was called from it. Therefore it fail even if run on linux.

Right, sorry.

> In the other hand, on assert_normal_exit(), io.close is called from
> sub thread. it's a source
> of difference.

Right, it changes the order of termination.

> Please try following code. it work on linux.
>
> io = IO.popen("false")
> th = Thread.new {
> io.read
> io.close
> v = $?
> p v
> v
> }
> sleep 0.5
>
> p th
> th.join
> p th
> p th.value

Indeed, that works on FreeBSD 8.2:
#
#
#

#

- Lucas

#7 - 08/29/2011 03:23 PM - lucas (Lucas Nussbaum)

On 29/08/11 at 13:56 +0900, Lucas Nussbaum wrote:

> On 29/08/11 at 12:43 +0900, KOSAKI Motohiro wrote:
>
> > I've just checked, and FreeBSD 8.2 is also affected by this issue.
> > Test script:
> > system("false")
> > sleep 0.5
> > th = Thread::new { $? }
> > p th
> > th.join
> > p th
> > p th.value
> >
> > This is buggy. $? is thread local storage variable. last exit value
> > was set to main thread.
> > because system() was called from it. Therefore it fail even if run on linux.
>
> Right, sorry.
>
> > In the other hand, on assert_normal_exit(), io.close is called from
> > sub thread. it's a source
> > of difference.
>
> Right, it changes the order of termination.

Err, no, it doesn't. the popen process still ends before the thread. I
don't see why io.close would be a source of difference?

In any case, it's a bug in Debian GNU/kfreebsd. With the following test
case:
<----------
#include
#include
#include
#include
#include
#include

int retval;
int pid;

void * thread_get_retval(void *arg) {
if (waitpid(pid, &retval, 0) == -1) {
perror("waitpid");
exit(1);
}
printf("hello\n");
return(0);
}

int main() {
pthread_t tid;
if ((pid = fork()) == 0) {
return 42;
}
printf("PID: %d\n", pid);

```
if (pthread_create(&tid, NULL, thread_get_retval, NULL)) {
    perror("pthread_create");
    exit(1);
}

if (pthread_join( tid, NULL )) {
    perror("pthread_join");
```

```
    exit(1);
}
printf("Retval: %d\n", WEXITSTATUS(retval));
return 0;

}
```
---------->

Debian GNU/Linux:
PID: 27610
hello
Retval: 42

FreeBSD:
PID: 1172
hello
Retval: 42

Debian GNU/kFreeBSD:
./debian-kfreebsd-amd64:~# ./forkthread
PID: 719
waitpid: No child processes

Do you still want to apply my patch? I could apply it only in the Debian
package if you prefer.

- Lucas

#8 - 08/29/2011 03:43 PM - kosaki (Motohiro KOSAKI)

- Status changed from Assigned to Rejected

#9 - 08/29/2011 03:53 PM - kosaki (Motohiro KOSAKI)

> > In the other hand, on assert_normal_exit(), io.close is called from
> > sub thread. it's a source
> > of difference.
>
> Right, it changes the order of termination.

Err, no, it doesn't. the popen process still ends before the thread. I
don't see why io.close would be a source of difference?

In ruby internal, system(), io.close do 1) wait to finish child process *and*
2) set a last exit code of a child process to caller thread's thread
local variable.

In the other word, $? is a more magical variable than you did expect. ;-)

btw, please imazine what's happen if $? is true global variable, it's
completely useless.

Debian GNU/Linux:
PID: 27610
hello
Retval: 42

FreeBSD:
PID: 1172
hello
Retval: 42

Debian GNU/kFreeBSD:
./debian-kfreebsd-amd64:~# ./forkthread
PID: 719
waitpid: No child processes

Do you still want to apply my patch? I could apply it only in the Debian
package if you prefer.

It would be better. Thanks.

Files

assert_normal_exit.patch	684 Bytes	08/28/2011	lucas (Lucas Nussbaum)