Ruby master - Feature #8809

Process.clock_getres

08/22/2013 11:33 PM - akr (Akira Tanaka)

Status:	Closed	
Priority:	Normal	
Assignee:		
Target version:		

Description

How about Process.clock_getres method?

POSIX defines clock_getres function to provide resolution information
of clocks.

I made a pacth to invoke clock_getres function.

Process.clock_getres(Process::CLOCK_MONOTONIC) #=> 1.0e-09
Process.clock_getres(Process::CLOCK_MONOTONIC_COARSE) #=> 0.00400025

The result means that the resolution of CLOCK_MONOTONIC is 1ns and
the resolution of CLOCK_MONOTONIC_COARSE is 4.00025ms.

Process.clock_getres has optional unit argument as Process.clock_gettime.

Process.clock_getres(Process::CLOCK_MONOTONIC, :nanosecond) #=> 1
Process.clock_getres(Process::CLOCK_MONOTONIC_COARSE, :nanosecond) #=> 4000250

It supports emulated clocks as well.

Process.clock_getres(:SUS_GETTIMEOFDAY_BASED_CLOCK_REALTIME) #=> 1.0000000000000002e-06
Process.clock_getres(:SUS_GETRUSAGE_BASED_CLOCK_PROCESS_CPUTIME_ID) #=> 1.0000000000000002e-06

The unit argument can be :hertz, which means the reciprocal of the second.

Process.clock_getres(:SUS_GETRUSAGE_BASED_CLOCK_PROCESS_CPUTIME_ID, :hertz) #=> 1000000.0

Note that
Process.clock_getres(:POSIX_TIMES_BASED_CLOCK_PROCESS_CPUTIME_ID, :hertz) is the clock ticks per second (CLK_TCK) and
Process.clock_getres(:ISO_C_CLOCK_BASED_CLOCK_PROCESS_CPUTIME_ID, :hertz) is CLOCK_PER_SEC.
I wanted to access them easily to investigate emulated clock behaviors on
various OSes.

Any comments?

Associated revisions

Revision 23da5a78 - 08/31/2013 01:21 PM - akr (Akira Tanaka)

- process.c (rb_clock_getres): New method.
  (timetick2dblnum_reciprocal): New function.
- configure.in: Check clock_getres.

[ruby-core:56780] [Feature #8809] accepted at
DevelopersMeeting20130831Japan
https://bugs.ruby-lang.org/projects/ruby/wiki/DevelopersMeeting20130831Japan

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@42744 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 42744 - 08/31/2013 01:21 PM - akr (Akira Tanaka)

- process.c (rb_clock_getres): New method.
  (timetick2dblnum_reciprocal): New function.

- configure.in: Check clock_getres.

[ruby-core:56780] [Feature #8809] accepted at
DevelopersMeeting20130831Japan
https://bugs.ruby-lang.org/projects/ruby/wiki/DevelopersMeeting20130831Japan

Revision 42744 - 08/31/2013 01:21 PM - akr (Akira Tanaka)

- process.c (rb_clock_getres): New method.
  (timetick2dblnum_reciprocal): New function.

- configure.in: Check clock_getres.

[ruby-core:56780] [Feature #8809] accepted at
DevelopersMeeting20130831Japan
https://bugs.ruby-lang.org/projects/ruby/wiki/DevelopersMeeting20130831Japan

Revision 42744 - 08/31/2013 01:21 PM - akr (Akira Tanaka)

- process.c (rb_clock_getres): New method.
  (timetick2dblnum_reciprocal): New function.

- configure.in: Check clock_getres.

[ruby-core:56780] [Feature #8809] accepted at
DevelopersMeeting20130831Japan
https://bugs.ruby-lang.org/projects/ruby/wiki/DevelopersMeeting20130831Japan

Revision 42744 - 08/31/2013 01:21 PM - akr (Akira Tanaka)

- process.c (rb_clock_getres): New method.
  (timetick2dblnum_reciprocal): New function.

- configure.in: Check clock_getres.

[ruby-core:56780] [Feature #8809] accepted at
DevelopersMeeting20130831Japan
https://bugs.ruby-lang.org/projects/ruby/wiki/DevelopersMeeting20130831Japan

Revision 42744 - 08/31/2013 01:21 PM - akr (Akira Tanaka)

- process.c (rb_clock_getres): New method.
  (timetick2dblnum_reciprocal): New function.

- configure.in: Check clock_getres.

[ruby-core:56780] [Feature #8809] accepted at
DevelopersMeeting20130831Japan
https://bugs.ruby-lang.org/projects/ruby/wiki/DevelopersMeeting20130831Japan

Revision 42744 - 08/31/2013 01:21 PM - akr (Akira Tanaka)

- process.c (rb_clock_getres): New method.
  (timetick2dblnum_reciprocal): New function.

- configure.in: Check clock_getres.

[ruby-core:56780] [Feature #8809] accepted at
DevelopersMeeting20130831Japan
https://bugs.ruby-lang.org/projects/ruby/wiki/DevelopersMeeting20130831Japan

History

#1 - 08/23/2013 01:23 AM - david_macmahon (David MacMahon)

On Aug 22, 2013, at 7:33 AM, akr (Akira Tanaka) wrote:

> I made a pacth to invoke clock_getres function.

Thanks for making a patch! It makes the discussion much less abstract (more real?). I think I will try to follow your example in the future.

> Process.clock_getres(Process::CLOCK_MONOTONIC) #=> 1.0e-09
> Process.clock_getres(Process::CLOCK_MONOTONIC_COARSE) #=> 0.00400025
>
> The result means that the resolution of CLOCK_MONOTONIC is 1ns and
> the resolution of CLOCK_MONOTONIC_COARSE is 4.00025ms.

Did you consider having these methods return Rational rather than Float?

> Process.clock_getres has optional unit argument as Process.clock_gettime.
>
> Process.clock_getres(Process::CLOCK_MONOTONIC, :nanosecond) #=> 1
> Process.clock_getres(Process::CLOCK_MONOTONIC_COARSE, :nanosecond) #=> 4000250
>
> It supports emulated clocks as well.
>
> Process.clock_getres(:SUS_GETTIMEOFDAY_BASED_CLOCK_REALTIME) #=> 1.0000000000000002e-06
> Process.clock_getres(:SUS_GETRUSAGE_BASED_CLOCK_PROCESS_CPUTIME_ID) #=> 1.0000000000000002e-06
>
> The unit argument can be :hertz, which means the reciprocal of the second.
>
> Process.clock_getres(:SUS_GETRUSAGE_BASED_CLOCK_PROCESS_CPUTIME_ID, :hertz) #=> 1000000.0

How would you feel about supporting :ns and :hz as equivalents for :nanosecond and :hertz?

> Note that
> Process.clock_getres(:POSIX_TIMES_BASED_CLOCK_PROCESS_CPUTIME_ID, :hertz) is the clock ticks per second (CLK_TCK) and
> Process.clock_getres(:ISO_C_CLOCK_BASED_CLOCK_PROCESS_CPUTIME_ID, :hertz) is CLOCK_PER_SEC.
> I wanted to access them easily to investigate emulated clock behaviors on
> various OSes.

Those are some long symbols! Are these intended only for experimental/investigative use?

> Any comments?

I appreciate having access to POSIX functionality, so I'm all for this idea!

Thanks,
Dave

#2 - 08/23/2013 07:53 AM - akr (Akira Tanaka)

2013/8/23 David MacMahon davidm@astro.berkeley.edu:

> > Process.clock_getres(Process::CLOCK_MONOTONIC) #=> 1.0e-09
> > Process.clock_getres(Process::CLOCK_MONOTONIC_COARSE) #=> 0.00400025

>> The result means that the resolution of CLOCK_MONOTONIC is 1ns and
>> the resolution of CLOCK_MONOTONIC_COARSE is 4.00025ms.

> Did you consider having these methods return Rational rather than Float?

Process.clock_getres can return rational if it supports
:rational_second as a unit.

The current default of unit is :float_second and
I think float is good enough.

>> Process.clock_getres(:SUS_GETRUSAGE_BASED_CLOCK_PROCESS_CPUTIME_ID, :hertz) #=> 1000000.0

> How would you feel about supporting :ns and :hz as equivalents for :nanosecond and :hertz?

It is difficult to support :microsecond in that style
because the SI prefix, Greek m, is not representable in ASCII.

Someone may argue :hz should be :Hz.

I feel :float_s is bit curious.

So it is difficult to adopt :ns style as canonical style of unit.

I think several aliases are possible but
I'd like to concentrate to main feature.
The discussion for what aliases should be added or not can be diverge.

>> Note that
>> Process.clock_getres(:POSIX_TIMES_BASED_CLOCK_PROCESS_CPUTIME_ID, :hertz) is the clock ticks per second (CLK_TCK) and
>> Process.clock_getres(:ISO_C_CLOCK_BASED_CLOCK_PROCESS_CPUTIME_ID, :hertz) is CLOCK_PER_SEC.
>> I wanted to access them easily to investigate emulated clock behaviors on
>> various OSes.

> Those are some long symbols! Are these intended only for experimental/investigative use?

I choose the long symbols that is longer than Process::CLOCK_PROCESS_CPUTIME_ID.
Basically users should use Process::CLOCK_PROCESS_CPUTIME_ID if no reason.
--
Tanaka Akira

#3 - 08/23/2013 08:23 AM - david_macmahon (David MacMahon)

On Aug 22, 2013, at 3:37 PM, Tanaka Akira wrote:

> Process.clock_getres can return rational if it supports
> :rational_second as a unit.
>
> The current default of unit is :float_second and
> I think float is good enough.

Agreed. Plus, if someone really wants, they can request nanosecond precision, which is all that clock_getres supports (at least on Linux).

>>> Process.clock_getres(:SUS_GETRUSAGE_BASED_CLOCK_PROCESS_CPUTIME_ID, :hertz) #=> 1000000.0

>> How would you feel about supporting :ns and :hz as equivalents for :nanosecond and :hertz?

> It is difficult to support :microsecond in that style
> because the SI prefix, Greek m, is not representable in ASCII.

I know it's not SI, but I often use ASCII "u" for Greek m ("µ"), so :microsecond would be aliased by :us.

> Someone may argue :hz should be :Hz.

No doubt! :-)

> I feel :float_s is bit curious.

How about separating the type and the resolution into two different parameters?

```
Process.clock_getres(Process::CLOCK_MONOTONIC, :float, :second)
```

...or...

```
Process.clock_getres(Process::CLOCK_MONOTONIC, Float, :second)
```

> So it is difficult to adopt :ns style as canonical style of unit.
>
> I think several aliases are possible but
> I'd like to concentrate to main feature.
> The discussion for what aliases should be added or not can be diverge.

Agreed. I think the main feature is great!

> > > Note that
> > > Process.clock_getres(:POSIX_TIMES_BASED_CLOCK_PROCESS_CPUTIME_ID, :hertz) is the clock ticks per second (CLK_TCK)
> > > and
> > > Process.clock_getres(:ISO_C_CLOCK_BASED_CLOCK_PROCESS_CPUTIME_ID, :hertz) is CLOCK_PER_SEC.
> > > I wanted to access them easily to investigate emulated clock behaviors on
> > > various OSes.
> >
> > Those are some long symbols! Are these intended only for experimental/investigative use?
>
> I choose the long symbols that is longer than Process::CLOCK_PROCESS_CPUTIME_ID.
> Basically users should use Process::CLOCK_PROCESS_CPUTIME_ID if no reason.

Sounds good.

Dave

#4 - 08/24/2013 09:23 AM - akr (Akira Tanaka)

david_macmahon (David MacMahon) wrote:

> I know it's not SI, but I often use ASCII "u" for Greek m ("µ"), so :microsecond would be aliased by :us.

It may be possible.

I found ISO 2955.
ISO 2955: Information processing - Representation units in Systems with limited Character sets

> > I feel :float_s is bit curious.
>
> How about separating the type and the resolution into two different parameters?
>
> ```
> Process.clock_getres(Process::CLOCK_MONOTONIC, :float, :second)
> ```
>
> ...or...
>
> ```
> Process.clock_getres(Process::CLOCK_MONOTONIC, Float, :second)
> ```

I think most useful combinations are follows.

- float second
- integer nanosecond (clock_gettime/clock_getres native format)

The current design makes us possible to specify
former as no unit argument and
later as :nanosecond.

Your design force us longer description for integer nanosecond.

#5 - 08/24/2013 09:24 AM - akr (Akira Tanaka)

- File clock_getres-2.patch added

I updated the patch.

#6 - 08/24/2013 10:41 AM - akr (Akira Tanaka)

- File clock_getres-3.patch added

I updated the patch again.

#7 - 08/31/2013 10:21 PM - akr (Akira Tanaka)

- Status changed from Open to Closed

- % Done changed from 0 to 100

This issue was solved with changeset r42744.
Akira, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

- process.c (rb_clock_getres): New method.
  (timetick2dblnum_reciprocal): New function.

- configure.in: Check clock_getres.

[ruby-core:56780] [Feature #8809] accepted at
DevelopersMeeting20130831Japan
https://bugs.ruby-lang.org/projects/ruby/wiki/DevelopersMeeting20130831Japan

Files

clock_getres.patch	4.39 KB	08/22/2013	akr (Akira Tanaka)
clock_getres-2.patch	6.06 KB	08/24/2013	akr (Akira Tanaka)
clock_getres-3.patch	6 KB	08/24/2013	akr (Akira Tanaka)