Ruby trunk - Feature #11558

Time related C APIs

09/30/2015 05:59 AM - naruse (Yui NARUSE)

Status:	Closed
Priority:	Normal
Assignee:	
Target version:	

Description

Time関連のC APIを追加して欲しいです。
具体的には以下のようなものが欲しいです。

struct timespecとoffsetを取って、Timeを返してください。
VALUE rb_time_timespec_new(const struct timespec *ts, int offset);
趣旨としては、[秒, ナノ秒, offset]からTimeを作って欲しいと言うことです。
（rb_time_num_new(Rational, offset)とかだと手間かかる上に遅い）

現在時刻をstruct timespecで欲しいです。
void timespec_now(struct timespec *ts);

既存の非公開APIを公開してください。
struct tm * localtime_with_gmtoff_zone(const time_t *t, struct tm *result, long *gmtoff, const char **zone);
time_t timegm_noleapsecond(struct tm *tm);
void tm_add_offset(struct tm *tm, long diff);

https://github.com/nurse/strptime/blob/v0.1.1/ext/strptime/ruby_time.c

History

#1 - 09/30/2015 08:55 AM - akr (Akira Tanaka)

Yui NARUSE wrote:

> Time関連のC APIを追加して欲しいです。

要望は理解できます。

> struct timespecとoffsetを取って、Timeを返してください。
> VALUE rb_time_timespec_new(const struct timespec *ts, int offset);
> 趣旨としては、[秒, ナノ秒, offset]からTimeを作って欲しいと言うことです。
> （rb_time_num_new(Rational, offset)とかだと手間かかる上に遅い）

いいんじゃないでしょうか。

> 現在時刻をstruct timespecで欲しいです。
> void timespec_now(struct timespec *ts);

Ruby の定義であることがわかる prefix が名前に必要だと思います。

> 既存の非公開APIを公開してください。
> struct tm * localtime_with_gmtoff_zone(const time_t *t, struct tm *result, long *gmtoff, const char **zone);

time_t を使うと 2038年問題を考慮しないといけないので、よくないと思います。
struct tm を使うと (tm_year が int なので) 2**31年問題を考慮しないといけないので、よくないと思います。
zone を char * で返すのはメモリ管理に責任を持てなくなるので、よくないと思います。

> time_t timegm_noleapsecond(struct tm *tm);

time_t と struct tm は上記のようによくないと思います。

> void tm_add_offset(struct tm *tm, long diff);

struct tm は上記のようによくないと思います。
tm_year が overflow したらどうするんでしょうか。

まぁ、time_t と struct tm が癌なので、Ruby の公開API用の構造体を定義すれば解決できるとは思います。
(あと zone は Ruby 内部で VALUE で扱うことに変えないといけないと思う。)

#2 - 09/30/2015 10:12 AM - naruse (Yui NARUSE)

Akira Tanaka wrote:

> Yui NARUSE wrote:
>
>> 既存の非公開APIを公開してください。
>> struct tm * localtime_with_gmtoff_zone(const time_t *t, struct tm *result, long *gmtoff, const char **zone);
>
> time_t を使うと 2038年問題を考慮しないといけないので、よくないと思います。
> struct tm を使うと (tm_year が int なので) 2**31年問題を考慮しないといけないので、よくないと思います。

time_tは入力なので、このAPIが考える必要は必ずしもないのではありませんか。
内部処理はyearはint64_tでやって、struct tmにつっこむときにエラーを返せばいいかな。

> zone を char * で返すのはメモリ管理に責任を持てなくなるので、よくないと思います。

これは確かにNULLを渡すことしか考えていなかったので、zoneを削除したものにした方がよいですね。

>> time_t timegm_noleapsecond(struct tm *tm);
>
> time_t と struct tm は上記のようによくないと思います。

こちらはtimet_t (32bit) が危ないですね。
入力はstruct tmのままで、戻り値は64bitですかねぇ。

>> void tm_add_offset(struct tm *tm, long diff);
>
> struct tm は上記のようによくないと思います。
> tm_year が overflow したらどうするんでしょうか。

tm_yearがint64_tでもギリギリ狙ったらあふれますよね。
さておき、戻り値intにしてエラー返すのがよいかな。

> まぁ、time_t と struct tm が癌なので、Ruby の公開API用の構造体を定義すれば解決できるとは思います。

VALUEにしないかぎり、入力を納められる出力のサイズ問題は解決不可能ではありませんか。
VALUEにはしたくありませんが……。

そういえば、struct timespecもtv_sevはtime_tですね。

#3 - 09/30/2015 12:07 PM - akr (Akira Tanaka)

Yui NARUSE wrote:

>>> struct tm * localtime_with_gmtoff_zone(const time_t *t, struct tm *result, long *gmtoff, const char **zone);
>>
>> time_t を使うと 2038年問題を考慮しないといけないので、よくないと思います。
>
> time_tは入力なので、このAPIが考える必要は必ずしもないのではありませんか。
> 内部処理はyearはint64_tでやって、struct tmにつっこむときにエラーを返せばいいかな。

アプリケーションに time_t の値が与えられるのであればそうでしょうが、
アプリケーションが time_t の値を生成するのであれば、表現できなかったらどうするかというのは問題になるでしょう。
それはアプリケーションの責任であり、Ruby の責任ではない、という趣旨なのだとは思いますが、
実際のところアプリケーションが対処するのは難しいでしょうし、
2038年問題が現実に問題になることはそれなりにあるのではないでしょうか。

>>> time_t timegm_noleapsecond(struct tm *tm);
>>
>> time_t と struct tm は上記のようによくないと思います。
>
> こちらはtimet_t (32bit) が危ないですね。
> 入力はstruct tmのままで、戻り値は64bitですかねぇ。

エラーをどう表現するのか、というのも問題ですね。
-1 にすると 1969-12-31 23:59:59 UTC と区別がつかないんだよなぁ。

>>> void tm_add_offset(struct tm *tm, long diff);
>>
>> tm_year が overflow したらどうするんでしょうか。
>
> tm_yearがint64_tでもギリギリ狙ったらあふれますよね。
> さておき、戻り値intにしてエラー返すのがよいかな。

えぇ、void だとエラーを返せませんよね。

> VALUEにしないかぎり、入力を納められる出力のサイズ問題は解決不可能ではありませんか。
> VALUEにはしたくありませんが……。

VALUE にするか、64bit 整数型にするか、あるいは time.c の wideint_t みたいなことをするか、
というのが選択肢ですね。

VALUE にすると 32bit 環境で 1970年からの秒数が Bignum になるので遅く、
64bit 整数型にすると値の範囲の制限は残るためちゃんとエラーを扱わなければならず、
wideint_t みたいなのは話が大がかりなものになる、
というのが主な得失でしょうか。

> そういえば、struct timespecもtv_sevはtime_tですね。

えぇ、time_t というよりは struct timespec の代わりの型をつくったほうがいいでしょうね。
struct timespec を受け付ける API 自体は、扱えないときに fallback する先があれば問題ないのですが。
(あと現在時刻を struct timespec で返すのは問題ないと思います。)

#4 - 11/08/2015 06:44 AM - naruse (Yui NARUSE)

とりあえず特に異論の無かった下記を入れようかと思います。

```
diff --git a/include/ruby/intern.h b/include/ruby/intern.h
index af6b75d..3fb1637 100644
--- a/include/ruby/intern.h
+++ b/include/ruby/intern.h
@@ -919,8 +919,10 @@ VALUE rb_mutex_unlock(VALUE mutex);
 VALUE rb_mutex_sleep(VALUE self, VALUE timeout);
 VALUE rb_mutex_synchronize(VALUE mutex, VALUE (*func)(VALUE arg), VALUE arg);
 /* time.c */
+void rb_timespec_now(struct timespec *);
 VALUE rb_time_new(time_t, long);
 VALUE rb_time_nano_new(time_t, long);
+VALUE rb_time_timespec_new(const struct timespec *, int);
 VALUE rb_time_num_new(VALUE, VALUE);
 struct timeval rb_time_interval(VALUE num);
 struct timeval rb_time_timeval(VALUE time);
diff --git a/time.c b/time.c
index 11c76a5..da8cf25 100644
--- a/time.c
+++ b/time.c
@@ -1892,6 +1892,25 @@ timew2timespec_exact(wideval_t timew, struct timespec *ts)
     return ts;
 }

+void
+rb_timespec_now(struct timespec *ts)
+{
+#ifdef HAVE_CLOCK_GETTIME
```

```
+    if (clock_gettime(CLOCK_REALTIME, ts) == -1) {
+   rb_sys_fail("clock_gettime");
+    }
+#else
+    {
+   struct timeval tv;
+   if (gettimeofday(&tv, 0) < 0) {
+       rb_sys_fail("gettimeofday");
+   }
+   ts->tv_sec = tv.tv_sec;
+   ts->tv_nsec = tv.tv_usec * 1000;
+    }
+#endif
+}
+
 static VALUE
 time_init_0(VALUE time)
 {
@@ -1903,20 +1922,7 @@ time_init_0(VALUE time)
     tobj->gmt = 0;
     tobj->tm_got=0;
     tobj->timew = WINT2FIXWV(0);
-#ifdef HAVE_CLOCK_GETTIME
-    if (clock_gettime(CLOCK_REALTIME, &ts) == -1) {
-   rb_sys_fail("clock_gettime");
-    }
-#else
-    {
-        struct timeval tv;
-        if (gettimeofday(&tv, 0) < 0) {
-            rb_sys_fail("gettimeofday");
-        }
-        ts.tv_sec = tv.tv_sec;
-        ts.tv_nsec = tv.tv_usec * 1000;
-    }
-#endif
+    rb_timespec_now(&ts);
     tobj->timew = timespec2timew(&ts);

     return time;
@@ -2306,6 +2312,23 @@ rb_time_nano_new(time_t sec, long nsec)
 }

 VALUE
+rb_time_timespec_new(const struct timespec *ts, int offset)
+{
+    VALUE time = time_new_timew(rb_cTime, nsec2timew(ts->tv_sec, ts->tv_nsec));
+    if (offset) {
+   struct time_object *tobj;
+   if (offset < -86400 || 86400 < offset)
+       rb_raise(rb_eArgError, "utc_offset out of range");
+   GetTimeval(time, tobj);
+   tobj->tm_got = 0;
+   tobj->gmt = 2;
+   tobj->vtm.utc_offset = INT2FIX(offset);
+   tobj->vtm.zone = NULL;
+    }
+    return time;
+}
+
+VALUE
 rb_time_num_new(VALUE timev, VALUE off)
 {
     VALUE time = time_new_timew(rb_cTime, rb_time_magnify(v2w(timev)));
```

#5 - 11/12/2015 03:40 AM - naruse (Yui NARUSE)

- Status changed from Open to Closed