# Ruby における 2038年問題の解決

田中 哲
独立行政法人
産業技術総合研究所
情報技術研究部門

札幌Ruby会議02
2009-12-05

# 概要

- Ruby 1.9.2 でTime クラスの制約が緩和
- 2038年以降も表現可能
  (OS の time_t の制約に縛られない)
- 1901年以前も表現可能
- マイクロ秒よりも細かい精度も表現可能
  (有理数を表現できる)
- 任意の時差を表現可能
- 不合理な制約をなくすことにより容易なプログラミングを実現

# ありそうなストーリー

- Ruby では時刻を表現するのに Time クラスを使う
- アプリケーションで Time を使うことはよくある
- だいたい、それで動作する
- あるとき、メールやデータベースから、範囲外の時刻が届く
- アプリケーションが動かない
- そもそも Time を使うのが間違いだったということが判明して大改修
- Time がもっと広い範囲を表現できていれば...

# Unix 伝統の制約の緩和

- 伝統的に time_t は 32bit 符号つき整数
  -2147483648〜2147483647
- 1970-01-01 00:00:00 UTC を起点とする秒数
- 表現可能範囲: 1901-12-14 〜 2038-01-19
- 2038年問題

# 2038年問題の解決
# 32bit 制約のクリア

- p Time.new(2112,9,3)
  #=> 2112-09-03 00:00:00 +0900
- p Time.new(1868,9,3)
  #=> 1868-09-03 00:00:00 +0900

# 2922億年問題の解決
# 64bit 制約のクリア

- 最近の OS には time_t が 64bit のものもある
- -292277022657年〜292277026596年
- Time.new(10**18)
  #=> 10000000000000000000-01-01 00:00:00 +0900

# 1秒以下の表現
# 秒に有理数を使える

- Ruby 1.8 : マイクロ秒単位
- Ruby 1.9.1 : ナノ秒単位
- Ruby 1.9.2 : 有理数

- t = Time.new(2009,12,5,0,0,Rational(4,3))
  p t　　#=> 2009-12-05 00:00:01 +0900
  p t.subsec　#=> (1/3)

# 任意の時差の表現

- (固定した) 時差を指定可能
- Time オブジェクト毎に変えられる
- p Time.new(2009,12,5,12,0,0,2*60*60)
  #=> 2009-12-05 12:00:00 +0200
- p Time.new(2009,12,5,12,0,0,-5*60*60)
  #=> 2009-12-05 12:00:00 -0500

# まとめ

- いろいろ表現可能になった
- おしまい

# バックグラウンド

- 自然の話
- 人間の話
- 技術の話

# 自然 (科学) の話

- 地球の公転周期は自転周期の整数倍でない
- 自転周期は (原子時計による) 1秒の整数倍でない
- そもそも自転周期は一定でない
  - ゆらぐ
    (マントルの影響?)
  - 長期的には長くなっていく
    (潮汐力により自転にブレーキがかかる)
- 相対論：時間の流れは一定でない

# 人間 (暦) の話

- 12月は (北半球で) 冬であってほしい
  春分の日は 3月20日か 3月21日であってほしい
  公転周期は自転周期の非整数倍
  閏年(閏日:2月29日)で 1年の日数を調整
- 12時は昼であってほしい
  - 地球は丸いので地域差がある
    時差で調整 (各国で決定)
  - 自転周期は秒の非整数倍
    閏秒で 1日の秒数を調整 (国際機関で決定)
- 夏時間 : 夏のあいだ時差を変える (各国で決定)
- グレゴリオ歴以外の暦 : ユリウス歴、天保歴など

# 技術の話

- 32bit time_t の制約 (1901年 〜 2038年)
- struct tm の tm_year の制約 (±21億年)
- 64bit time_t の制約 (±2922億年)
- タイムゾーン情報 (夏時間、閏秒) の配布
- 単一プロセス内での複数のタイムゾーン
- POSIX のナノ秒タイムスタンプ: 10**(-9)
- FreeBSD の bintime: 2**(-64)
- 64bit time_t をサポートする OS の普及状況

# Design Decision

扱う:

- Ruby 1.8 でできること
  - time_t の範囲内
  - システムのタイムゾーン
  - UTC
  - (OSが扱うなら) 閏秒
- 有理数
  10**(-9) と 2**(-64) を
  両方とも正確に扱える
  ad hoc でない仕様

扱わない:

- 相対論：扱わない
  位置や速度は嫌
- グレゴリオ歴以外の暦
  「閏三月」とかいわれても困る

# 難しいがなんとか扱いたい

- OS のサポート範囲外の時刻 (2038年問題)
  問題: 時差・閏秒の情報が得られない
- 複数のタイムゾーン
  問題: 時差の情報が得られない

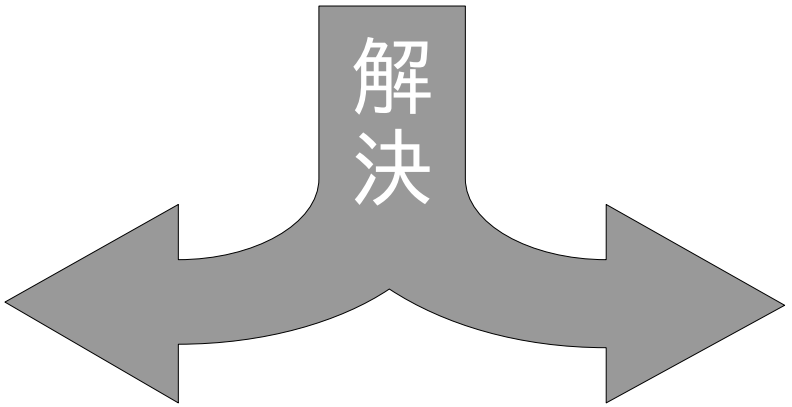
OS が時差・閏秒の情報を
提供してくれないのが問題

# 時差・閏秒の情報: zoneinfo

- 例: /usr/share/zoneinfo/Asia/Tokyo など
- ある地域における、時差の時系列変化のデータ
  各国の気まぐれで変化する
- 閏秒のデータ
  国際機関が決定するので世界中で統一されている。ただし、zoneinfo のファイル内に情報が入っている
  地球の回り具合に対応して追加されていく
- どちらも継続的な更新が必要
- 一般に OS ベンダが更新を用意する

# zoneinfo の問題

- time_t の範囲内しか情報が得られない
  - 32bit 符号つき time_t なら 2038年問題
- アクセス API の機能が足りない
  - 基本的に単一のタイムゾーンしか扱えない
    (ファイルとしてデータはあるのに)
- API は標準化されているが格納形式は OS に依存




zoneinfo を
自前で持つ
(Java, PHP)

zoneinfo を
持たない
(Perl, Ruby)

# zoneinfo を自前で持つ?

利点

- 標準APIを無視して直接アクセスできる
- OS に依存しないですべてのタイムゾーンの情報を安定して得られる
- time_t よりも広い範囲の情報を用意できる
- 複数のタイムゾーンもサポートできる

欠点

- OSの更新に頼らず自前で更新しなければならない
- OS が使っている zoneinfo と一貫しないかも?

# OSのzoneinfoだけでどこまでやれる?

- 2038年問題
  - Perl のアイデア:
    2038年以前の時差情報からそれ以降を推測
- 複数のタイムゾーン
  - 時差が固定の (夏時間がない) 仮想的なタイムゾーンは扱える
  - 環境変数 TZ を変えるのは難しい問題がある
    (OSによるタイムゾーン名の差異、レースコンディション)

# Ruby 1.9.2 の選択

- zoneinfo は自前では持たない
- 2038年問題は Perl 同様に解決する
- 時差が固定のタイムゾーンをサポートする

- zoneinfo 依存のタイムゾーンは、Ruby 1.8 同様システムのタイムゾーンだけサポートする

# 時差の推測

- y年m月d日w曜日の時差は
  y'年m月d日w曜日と等しいと推測する
- y' は 2038年以前のなるべく未来の年を選ぶ
- ただし 2月なら、y年と y'年は閏年かどうかも一致させる

- 64bit time_t の普及に伴って不要になっていくので一時的なもの
- 未来の時差はもともと変化する可能性がある

# 夏時間

- 夏のあいだ時差を変える
- 例: アメリカ (2007年から)
  - 3月第2日曜日午前2時 (標準時) から
  - 11月第1日曜日午前2時 (夏時間) まで
  - 1時間進める
- 例: ヨーロッパ
  - 3月最終日曜日午前1時 (UTC) から
  - 10月最終日曜日午前1時 (UTC) まで
  - 1時間進める
- 曜日が同じなら年を変えても同じになる

# 時差が固定のタイムゾーン

- 過去も未来も時差が一定な仮想的なタイムゾーン
- 例: UTC は時差が 0 で一定なタイムゾーン
- 特定の時刻の時差を表現するには十分
  t1 = Time.new(2009,12,5,12)
  p t1 #=> 2009-12-05 12:00:00 +0900
  t2 = t1.getlocal(-8*60*60)
  p t2 #=> 2009-12-04 19:00:00 -0800
- 複数のタイムゾーンをサポートするアプリケーションでデータ構造として利用できる

# 夏時間の厄介さ

- 存在しない時刻がある
  カリフォルニア:
  2009年3月8日2時以降
  2009年3月8日3時未満
  02:30 という時刻は存在しない
- 年月日時分秒で時刻が同定できない
  カリフォルニア:
  2009年11月1日1時以降
  2009年11月1日2時未満
  01:30 は夏時間と標準時の両方に存在する

# まとめ

- zoneinfo を持たない範囲でがんばっている
- zoneinfo の更新に巻き込まれない
- 2038年問題を解決
- 任意の固定時差を表現可能
- 1秒未満は有理数を表現可能
- 理不尽な制約をなるべくプログラマに見せない
- 容易なプログラミングの実現