Ruby における外部ライブラリに依存しない文字コード変換

富永 晃司，伊藤 一成，Martin J. DÜRST
青山学院大学理工学部

1 はじめに

インターネットの発達によりいつでも膨大のデータが行き交うようになった．しかしながら世界各地にはいくつもの文字コードが存在し，円滑な情報流通を妨げることがある．これから様々な場面で活躍が期待される各プログラミング言語やそのアプリケーションは，より本格的に文字コードに対応することが不可避となる．

つまりアプリケーションユーザビリティの一環として多くの文字コードを利用できる環境をプログラム言語自身が提供する必要がある．更に作成者にとって使い勝手が良く，文字コードを意識せずに対応させなければならない．

プログラミング言語 Ruby [1] においても多言語化が進んでいる．我々は昨年度 Ruby において既存の外部ライブラリを統合した [2]．本稿では新規に Ruby に組み込むことで外部に依存しない文字コード変換を行う方法を提案し実装したので，報告する．

2 Ruby の多言語化

Ruby は Java にとって代わるプログラミング言語と言われている [3]．そのためいかなる場所でも使用されることを想定し，文字コードの壁を早い段階から取り払わなければならない．

2.1 Ruby 1.8 の多言語化

2008 年 1 月現在 Ruby は安定版の Ruby 1.8 と開発版の Ruby 1.9 がある．Ruby 1.8 では，`String` オブジェクトをバイト列として扱い，文字単位の処理は正規表現のみを用いて行う．また文字コード変換に Iconv, Kconv や nkf といったライブラリが用いられている [1]．しかし，これらは変換による差異を考慮して使い分ける必要があった．そのため我々は昨年度，変換する文字コードによって Iconv や Kconv を使い分ける統合ライブラリを作成し [2]，発表した [4]．だが Iconv はプラットフォームによって実装が異なるため対応する文字コードに違いがある．また GNU 系の iconv ライブラリを使用している場合 GNU と Ruby ではライセンスが異なるため，Ruby 1.8 で多言語対応のソフトウェアを作成することが困難であった．

2.2 Ruby 1.9 の多言語化

Ruby 1.9 は 2007 年 12 月末に開発版としてリリースされた．多言語化への対応が強化され，`String` オブジェクト毎に文字コード情報がある．特定の文字コードの物理表現を参照する基本的処理 (プリミティブ) を定義し，文字列の処理をプリミティブを介して行う方式がとられている．そして `String` オブジェクト毎に文字コードに基づいてプリミティブを選択する [5]．そのため多くの場合 1 つだけの文字コードを使う際には他の文字コードに変換することなく処理を行うことができる．逆に複数の文字コードを扱う際に文字コードの変換をしない限り，例外が発生することがある．

3 変換手法

変換前文字コードと変換後文字コードに基づいて，1 つの特定の文字コードを介して変換を行う．変換の中核となる文字コードは様々な文字コードが入力されることを想定し，世界の殆どの文字を収録している Unicode を用いる．またエンコードは Ruby で中心となっている UTF-8 を採用した．主軸となる文字コードを定めることで，必要なテーブルの数を削減することが可能になった．

3.1 変換テーブル

3.1.1 変換テーブルの作成

変換テーブルは UTF-8 と，Shift_JIS のような UTF-8 とは異なる他文字コードを 1 組とし，文字コードの変換対応数の分用意する．これらの変換テーブルは全て Ruby で作成した．要素の構成順序は他文字コードの昇順に並べているため，UTF-8 は順不同になっている．また変換を行うライブラリは C 言語のレベルで開発したため，変換テーブルは C 言語の文字列の 2 次元配列データ構造である．

3.1.2 変換テーブルの問題点

各日本語文字コードにある 0x5C バイトはバックスラッシュと半角 ¥ 記号の 2 つの役割で扱われている．それらは Unicode においてそれぞれ U+005C 文字と U+00A5 文字の 2 種類を使い分けているため，1 対 1 の変換では対応しきれない箇所が存在する．

そのため UTF-8 で 5C バイト (バックスラッシュ) だったものを Shift_JIS にする変換と，UTF-8 で C2 A5 バイト (半角 ¥) だったものを Shift_JIS に変換する場合において同じバイト列を出力する．以下に Shift_JIS, UTF-8 において半角 ¥ 記号の変換テーブル要素を示す．

```
/* 左要素: Shift_JIS, 右要素: UTF-8 */
{"\x5C","\x5C"},
{"\x5C","\xC2\xA5"},
```

3.1.3 複数文字の変換

変換テーブルに文字列を用いるため複数文字の変換も可能になる．それが必要な例としてベトナム語用の文字コードである windows-1258 が挙げられる．ベト

Character-Encoding Conversion for Ruby without External Library Dependence

Koji TOMINAGA, Kazunari ITO and Martin J. DÜRST
Department of Integrated Information Technology, College of Science and Engineering, Aoyama Gakuin University
5–10–1 Fuchinobe, Sagamihara, Kanagawa 229-8558, Japan
tominaga@sw.it.aoyama.ac.jp, {kaz, duerst}@it.aoyama.ac.jp

ナム文字はアルファベットに補助記号や声調記号，その両方が付与されていることがある [6]．windows-1258 では声調記号が付与されている時，アルファベットと声調記号のそれぞれを 1 文字とみなしている．しかし Unicode では補助記号や声調記号がついているものも全て 1 文字として登録しているので，異なる文字数の間で変換を行うことがある．図 1 に windows-1258 と UTF-8 における複数文字の変換例を挙げる．このように変換時に文字の正規化を行う実装例は数少ない．

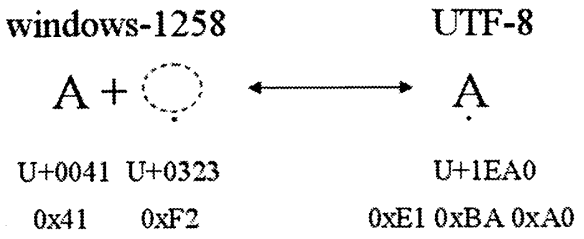

図 1: 複数文字の変換例

3.1.4 変換テーブルの探索方法

変換テーブルを探索する際，線形探索法を採用している．変換テーブルの探索方法はより早いアルゴリズムを活用したほうが良い．しかし，テーブルそのものの数を半減させるため，UTF-8 の昇降順のテーブルを用意しなかった．そのため対数関数的にデータ探索が済む二分探索を UTF-8 から変換する際に利用できないことから，どの変換も線形探索で行った．

3.2 組込メソッド

3.2.1 組込メソッドの作成

Ruby は C 言語で書かれているため C での拡張がスムーズに行える．また探索の高速化のため組込みメソッドは C 言語レベルで作成した．Ruby ではオブジェクトの実体を構造体で表現し，取り扱うときは VALUE 型のポインタを経由する．各クラス毎に構造体へのポインタにキャストを行い差別化させ，文字列は `RString` という構造体で扱う．`RString` では文字列へのポインタや文字列の長さをメンバとして有している [7]．

ファイルの構成は短い変換プログラムを 2 つ，文字コード分の変換テーブルを 8 種，1 つのヘッダファイルとなっている．また対応している文字コードは 2008 年 1 月現在 ASCII, UTF-8, Shift_JIS, CP932, EUC-JP, ISO-2022-JP, ISO-8859-1,2, windows-1258 の 9 種類である．対応可能な文字コードは容易に追加できる．変換テーブルの容量は表 1 のようになった．これらは実際のデータ量を計測するために Unix 系コマンドの strip を適用した後のデータ容量である．

表 1: 変換テーブルの容量

文字コードデータ	容量 (kbyte)
Shift_JIS	135
EUC-JP	233
CP932	143
ISO-2022-JP	106
ISO-8859-1	4
ISO-8859-2	4
WINDOWS-1258	8
ASCII	4

3.2.2 String クラスに追加されるメソッド

Ruby 1.9 は文字コード変換の際には変換後文字コードだけを指定すれば変換ができる．本稿では String クラスに直接メソッドとして transcode を追加する．transcode メソッドは引数を 1 つ又は 2 つ持つ．1 つの場合は引数を変換後コードであり，2 つの場合は変換後コードと変換前コードである．メソッドの書き方の例を次に示す．

```
# 現在の文字コードから Shift_JIS に変換
str.transcode('Shift_JIS')

# EUC-JP から Shift_JIS に変換
str.transcode('Shift_JIS', 'EUC-JP')
```

4 まとめとこれからの課題

Ruby において外部ライブラリに依存しない文字コード変換を実現するために，UTF-8 を主軸とした変換テーブルを利用した組込ライブラリを実装した．問題点は線形探索法を用いると，変換にかかる時間の増加が激しいことである．今後の課題として，テーブルの構成を変更することでテーブル全体のサイズを縮小し，探索方法もより効率的なものを使うことが挙げられる．

5 謝辞

本研究にあたり，Ruby 主開発者のまつもとゆきひろ氏をはじめ，多くの方々から多大なる御指導及び御助言を賜りました．ここに記して謝意を表します．

参考文献

[1] Dave Thomas, Chad Fowler and Andy Hunt : *Programming Ruby, Second Edition*, The Pragmatic Programmers (2006).

[2] 島田拓也, 伊藤一成, Martin J. Dürst ：Ruby における文字コード変換ライブラリの統合, 情報処理学会全国大会 (2006).

[3] Bruce Tate : *From Java to Ruby*, Pragmatic Bookshelf (2006).

[4] Sconf: Easy `charset` Conversion, `http://rubyforge.org/projects/sconf`.

[5] 松本行弘, 縄手雅彦：スクリプト言語 Ruby の拡張可能な多言語テキスト処理の実装, 情報処理学会論文誌, Vol. 46, No. 11, pp. 2633–2642 (2005).

[6] 三上喜貴：文字符号の歴史 アジア編, 共立出版 (2006).

[7] 青木峰郎：Ruby ソースコード完全解説, インプレス (2002).