1J－1

標準Cライブラリと同様のインタフェースを持つ日本語入出力ライブラリの開発および評価

牛嶋 哲
東京工業大学大学院理工学研究科情報科学専攻

1 はじめに

ソフトウェアを日本語に対応させるに当たって最も基本的な要求事項の一つはファイル入出力を日本語に対応させることである。それには日本語を含むテキストファイルをバイト列としてではなく文字の列として扱うための仕組みが必要である。この仕組みをライブラリという形式で用意することによってさまざまなソフトウェアの日本語化に役立てることができると考え、C言語用の日本語入出力ライブラリを作成した。まずライブラリの概要を紹介し、実装上の留意点について述べ、いくつかの点について評価を試みる。

2 日本語入出力ライブラリの概要

日本語入出力ライブラリはテキストファイル（端末装置等を含む）を日本語テキストストリームに写像する。日本語テキストストリームは文字の並びである。ただし各文字はASCIIあるいはJIS X 0208のいずれかに含まれるものとする。すべての文字は2バイトで表現される。ASCII文字の表現は第1バイトが0、第2バイトがその文字に対するASCIIコードである。JIS X 0208に含まれる文字はその文字に対する2バイト符号であらわす。

日本語入出力ライブラリは標準Cライブラリ[1]と同様のインタフェースを提供している。たとえば関数ja_printfおよびja_fputsのインタフェースは以下の通りである。

```
int ja_printf(ja_char *format, ...)
int ja_fputs(ja_char *s, ja_FILE *stream)
```

Development and Evaluation of a Japanese I/O Library with an Interface Similar to that of the Standard C Library
Tetsu Ushijima
Department of Information Science, Tokyo Institute of Technology

```
void copy(ja_FILE *in, *out)
{ int c;
  while ((c = ja_getc(in)) != EOF)
    ja_putc(c, out); }
```

図1: 日本語入出力ライブラリを用いたプログラム例

これらはそれぞれ標準Cライブラリの関数printfおよびfputsに対応する。このように日本語入出力ライブラリが定義する外部名は、標準Cライブラリにおいて対応する名前に「ja_」を前置したものになっている。

プログラム例として、入力ストリームから出力ストリームに文字単位で複写する関数を図1に示す。

3 実装上の留意点

3.1 端末入出力

シフト状態を持つような符号系（たとえばISO-2022-JP[2]）を端末装置に接続されたストリームにおいて用いる場合、出力文字および入力文字に対するエコーバックが混在しても文字化けが起こらないようにするためには注意が必要である。このような場合にはストリームが期待するシフト状態は端末装置のシフト状態と必ずしも一致しないからである。

そこで各端末装置のシフト状態を記録しておくための表を用意し、必要であれば入出力に先だって端末装置のシフト状態をそのストリームが期待するシフト状態にあわせることができるようにした。

3.2 ファイル位置の指定

入力ストリームにおいてファイル位置を求めるためには、バッファに読み込んだ文字の並びがファイル上では何バイトのバイト列であったかを知る必要がある。ところがエスケープシーケンスを用いて文字集合を切り替える符号系の場合にはバッファの内

容だけからそれを知ることは一般的にはできない。エスケープシーケンスが必要最小限であるとは限らないからである。そこでそのような符号系を用いる場合にはバッファに読み込んだ各文字について、その文字を得るまでに走査したバイト数を記録しておくことにした。

シフト状態を持つような符号系を用いるストリームについてはファイル位置を取得する時点でのシフト状態も記録しておく必要がある。ja_fgetpos および ja_fsetpos についてはファイル位置とともにシフト状態も保存・復帰するが、ja_ftell および ja_fseek については初期シフト状態においてのみ正しく動作する、ということにした。

4 評価

4.1 日本語化に要する手間

既存のソフトウェアを日本語に対応させるにはどの程度の手間が必要なのかを調べるために、C 言語の原プログラムから文字列リテラルを抜き出すプログラムである xstr を日本語に対応させ、文字列リテラルを ja_char の配列を参照する式に置き換えるように変更した。変更前の原プログラムは 471 行であり、具体的な変更点は以下の通りであった。

- 日本語入出力ライブラリ用ヘッダの取り込み。
- 名前の置き換え (ja_を前置、53 箇所)。
- 日本語リテラルを配列変数で代用 (13 箇所)。
- 日本語文字列操作関数の定義を追加。
- 文字列データベースファイルを 2 バイト単位で処理するように変更。

日本語化の本質的な部分は使用する関数を取り替えることによって達成されるが、細かな部分についてはプログラムの内容を理解した上で変更する必要があった。

4.2 基本性能

日本語入出力ライブラリの基本性能を測定するために、図 1 の関数を基にして符号系変換フィルタ nkf のクローン (nkfclone) を作成した。このフィルタおよび各種符号系変換フィルタが 863943 文字 (ASCII536600 文字、日本語 327343 文字) で構成されるファイルの符号系を変換するために要する時間を計測した (表 1)。なお使用した計算機 (および OS) は Sony NWS-5000 (NEWS-OS 4.2.1R) である。各フィルタは細かな機能に違いがあるので一概には言えないが、この結果を見る限り日本語入出力ライブラリの性能がひどく悪いということはない。

表 1: 符号系変換フィルタの実行時間

フィルタ	実行時間 (秒)
qkc 1.0	0.84
nkfclone	1.27
nkf 1.4	1.42
kc 1.7	1.76
ack 1.39	1.97

4.3 日本語化に伴う効率の変化

実際のソフトウェアにおいて日本語化に伴う効率の変化を調べるために、CLU 言語 [3] の処理系 CLU2C[4] に含まれる入出力用のライブラリモジュール stream (C で記述されている) が標準 C ライブラリの入出力関数を呼び出す場合と日本語入出力ライブラリの関数を呼び出す場合とで CLU コンパイラ (stream を用いてファイルを読み書きする) の実行時間がどう変化するのかを調べた。ただし使用した計算機は 4.2 節と同じである。両者のプロファイル結果を検討した結果、入出力処理の占める割合が 3.0% から 8.1% に変化し、全体として実行時間が 6.5% 増加したことがわかった。

5 まとめ

日本語入出力ライブラリの概要を紹介し、実装上の留意点について述べた。また、いくつかの点について評価を試みた。

参考文献

[1] Plauger, P. J.: *The Standard C Library*, Prentice-Hall, Inc. (1992).

[2] Murai, J., Crispin, M. and van der Poel, E.: Japanese Character Encoding for Internet Messages, RFC 1468 (1993).

[3] Liskov, B. et al.: *CLU Reference Manual*, Springer-Verlag (1981).

[4] 江原善: 開発効率と移植性を重視した CLU 言語処理系の作成, 修士論文, 東京工業大学 (1992).