

Linux マイグレーションガイド

Linux の Shift JIS サポート
— 現状とその対応策 —

2009 年 7 月 10 日

日本 OSS 推進フォーラム
プラットフォーム部会
マイグレーションタスクフォース

目次

はじめに.....	3
本資料のねらい.....	3
Linux の台頭と過去の資産の継承.....	3
Linux の日本語環境に対する誤解.....	3
日本における Linux/OSS の国際化.....	4
開発コミュニティ.....	4
Unicode 採用の利点.....	5
なぜ Linux で Shift JIS ロケールがサポートされない.....	6
ユーザの Shift JIS 資産.....	6
(1)データ移行の汎用論.....	6
(2)データ交換.....	7
(3)Linux の現状 (Shell スクリプト).....	7
(4)オープン系のプログラム資産 (C/C++・Java・PHP・Perl など).....	8
(5)レガシー系プログラム資産 (COBOL、帳票、など).....	9
(6)DBMS.....	9
(7)フォントの対応.....	9
(8)ターミナルエミュレーターでの Shift JIS ベンダ定義文字.....	10
まとめ.....	10
Appendix 1 ベンダ定義コードのマッピングの状況.....	10
Appendix 2 既知の問題と回避策の例.....	11
Appendix 3 参照 Web リソース一覧.....	12
本検討の貢献者.....	13

Copyright(c)日本 OSS 推進フォーラム-2009-

本資料は、GNU Free Documentation License によって公開されています。

同ライセンスの条文(英語):

<http://www.gnu.org/licenses/fdl.html>

Wikipedia による説明:

http://ja.wikipedia.org/wiki/GNU_Free_Documentation_License

はじめに

現在 Linux に限らず、多くのプラットフォームで、文字コードセットはUnicode (UTF-8 など) への移行が推奨されている。しかしながら日本では、特にエンタープライズのお客様において、大量のデータとアプリケーションプログラムで Shift JIS 系の文字コードが使用されている。Linux が本格的に企業のビジネスユースで使われ始め、Windows や UNIX 等のシステムで使われてきたアプリケーションプログラムやデータを Linux ベースのオープンソースのシステムで扱うようになってきた現在、過去の Shift JIS ベースの資産の継承をどうするかということが問題になるようになってきた。

本資料のねらい

本稿は、Shift JIS に関する議論のスタートとなるべく、Shift JIS のサポートの問題点の整理を行い、また、現在考えられる解決案の整理を試みるものである。特に、Linux 導入時、システムインテグレーターやソフトウェアベンダーの技術者がユーザから Shift JIS ユーザ資産の活用について相談された時に、ベースラインとして議論ができるような資料を提供することをねらいとしている。もちろん、システムインテグレーター、ソフトウェアベンダー、あるいは、ディストリビュータが個別に、本稿の記述を超えて、更に優れた解を提示することを制約するものではない。

Shift JIS コードとは: (Shift JIS コードの歴史) Wikipedia 参照

<http://ja.wikipedia.org/wiki/SJIS>

Linux の台頭と過去の資産の継承

現在 Linux の文字コードの扱いは、Unicode (UTF-8 など) を使用することが主流となっている。したがって、Linux を使用しようとするユーザーは、基本的にアプリケーションプログラムの処理する文字コードとして、既存のデータ資産を一括で Unicode に変換して移行するか、もしくは、プログラムを追加・変更して、データ資産の文字コードの Unicode への変換を随時行うなどしている。

Linux 黎明期のころ、Linux が専らメールサーバや Web サーバのシステムで使用されていたころは Shift JIS から Unicode への移行が大きく問題になることは少なかったが、Linux が基幹システムで使用されるようになり、UNIX 等の大型の基幹システムを引き継ぐ移行先システムとなったときに、長年蓄積された Shift JIS で書かれたプログラム資産とデータ資産をいかに効率よく Linux が引き継ぐかということが問題になるようになり、一部 Shift JIS の運用を継続することも選択肢とみなされるようになった。

Linux の日本語環境に対する誤解

Linux の Shift JIS への対応を見て、多分に Windows 環境に大きな影響を受けた方々から「そもそも、Linux の日本語環境は成熟していると言えるのか?」という発言がなされることがある。

実情は、Linux ディストリビューションで使用されている GNU C ライブラリ(glibc)のバージョン 2.2.6(2002 年)以降(2009 年現在バージョン 2.10 になっている)、国際化の実装が完成しており、特別な日本語版 Linux ディストリビューションなどを作らなくとも、Linux のアップストリームコミュニティがリリースしたカーネルそのまま、一つのアプリケーションがロケールを切り替えるだけで世界のいろいろな言語のサポートができるようになってきているし、また、JIS 第四水準の文字などでも何ら問題なくサポート可能となっている。また、ミッションクリティカルなサーバ環境で今でも使用されているコンソール環境においても、2001 年に登場した Linux カーネル 2.4 以降、多国語を使用することが出来るようになってきている。一部のデスクトップツールなどには未だ課題が少し残るものの、サーバの利用に関しては十分に成熟している。

日本における Linux/OSS の国際化

過去(2001~2002 年ころ)、Shift JIS を含めた日本語サポートのコミュニティが、OSS アプリケーション開発者を中心に立ち上がり、大きな努力を重ねる中で、日本語のサポート(国際化と地域化)を実現してきた。参加した開発者達は技術的見地、経済的見地、イノベーションの見地から、それぞれのアプリケーションを Unicode 対応にしていくことにかじを切った。このようなコミュニティの活動によって Linux の国際化ミッションが完成した。

開発コミュニティ

オープンソースである Linux は周知のとおり、開発とメンテナンスを行うコミュニティが存在し、これらコミュニティによってソースコードが開発され、改善される仕組みを有している。エンドユーザを含めたグローバルコミュニティが開発することで、世界のユーザニーズの反映や改善が高速に行われる。したがって、Shift JIS の対応を Linux で行うには、Shift JIS 対応の改良を実施し、かつ、メンテナンスを行うコミュニティの力が必要である。残念ながら、Shift JIS に関する改善の活動を阻害する次のような要素がある。

1. Unicode と違い Shift JIS が日本独自の符号化方式であるうえ、POSIX 準拠ではないために、Shift JIS 対応をしようとすると他の符号化方式で問題が起きることがあげられる。つまり、世界中の各国語をサポートしなければならない開発者からみたときに、受け入れがたい開発課題ということである。その結果、Shift JIS サポートが日本の特殊な要求であり、全体的な方向性から逸れた提案と理解されてしまう。
2. Shift JIS が、PC 登場以前の 1 バイトカナコードのデータ資産の継承を目指した過去の技術であり、新しい技術への挑戦を尊ぶ Linux の開発技術者が自発的に開発に取り組む要素が考えにくいこと。
3. アプリケーションを Shift JIS で構築するには、シェルや基本コマンドなど、アプリケーションから呼び出される多数の Linux の中核的なプログラム群によって Shift JIS がサポートされる必要がある。このような中核プログラム群の開発者、特に当該パッチの組み込みを承認するメンテナの多くが海外の開発者であり、例え、日本人開発者が Shift JIS に関わる修正パッチを投稿しても、このような開発者の理解を得ることが、コミュニケーションの問題を克服してもなお難しいということである。
4. 国際化は単に C ライブラリや Linux コマンドだけの問題ではないという事情がある。オープンシステム上のアプリケーションでは、POSIX のロケール機能を使って国際化を実現する

のが常道ではあるが、Shift JIS 対応をしようと思った場合には、それに加えて、それぞれのアプリケーションソフトが個別にその対応を行う必要がある。OSS アプリケーションでは、C ライブラリの国際化対応以前から開発されているアプリケーションが多いこともあって、歴史あるソフトウェアの場合、POSIX のロケール機能を使って国際化を実現せずに、それぞれのソフトで独自に国際化を実現している事も多くなっているが、OSS コミュニティの Shift JIS サポートのメンテナンス負荷はさらに大きくなる。

5. UNIX は文字コード非依存の目標で作られていることが多く、そのための大きな開発投資を厭うことなく Shift JIS への対応を行ってきた。一方、Linux では、glibc (GNU libc) のワイド文字を Unicode にしたため、Shift JIS 固有のベンダ定義文字やユーザ定義文字を変換できないといった問題を根本的に解決することができない。

これらの背景に基づき課題の解決にむけて本資料を提供する。

Unicode 採用の利点

Linux を導入するとユーザは特に意識しなくても Unicode(UCS や UTF8 など)を利用することになり、世界標準に準拠した Unicode のメリットにより世界各国言語の混在したアプリケーションも容易に作成できるし、また、Unicode に対応した多様なミドルウェアを利用できる。最新の日本語処理の標準 JIS X0213 も、Unicode との整合性を保っており、今後も Unicode 環境で高い品質のアプリケーションが流通することが予想される。

今後、Unicode の積極的な利用が推奨される理由は次の通りである；

1. 一つの企業の中に様々なコンピュータプラットフォームが混在する状況で、どのプラットフォームでも利用できるコード系である
2. 国内における企業間、あるいは、公機関とのデータ交換でデフォルトとして採用される可能性が高いコード系である
3. 企業のグローバル化により各拠点で使用するアプリケーションの共通化が要請される状況で、多くのアプリケーションが対応できるコード系である
4. Windows のベンダ定義文字と同時に JIS X 0213:2004 で追加になった文字も扱える (Windows の Shift JIS と Shift_JIS-2004 は互換性がないため、混ぜて使うことが出来ない)
5. 適切に国際化対応を考慮すれば、日本語対応の特殊コード(バックスラッシュの考慮など)を追加する必要がなく、容易に高い品質のソフトウェアを開発できる
6. Unicode の新しい規格 (Unicode 5.1) では、JIS 漢字で同一コードポイントに包摂されている文字を区別して扱うことが出来るようになった (異体字シーケンス)

(参考:

<http://internet.watch.impress.co.jp/cda/jouyou/2008/09/10/20794.html>)

なぜ Linux で Shift JIS ロケールがサポートされない

現在、日本で利用されている多くの Linux ディストリビューションでも、Unicode 系の UTF-8 がデフォルトとされ、Shift JIS ロケールが用意されているケースでも、利用は推奨されていない。ちなみに、ユーザーのロケール設定は、Linux ターミナル画面で locale コマンドを打てば LANG=ja_JP.UTF8 のように表示されるので確認できる。

Shift JIS 系ロケール(sjis、cp932、ibm943 など、Appendix 1 参照)は、次のような理由のために推奨されていない；

1. Linuxの文字処理ライブラリ関数は、Unicode を扱うことを基本としているため、本ライブラリ関数を使ってインプリメントされた Linux システムコマンドでは、ファイルデータの中の文字処理や、ファイル名の処理で、Unicode は正しく扱っても、Shift JIS は扱えないことがある。
2. Shift JIS データの処理は、「特別」な扱いとなり、メールクライアント Thunderbird など、個々のミドルウェアに多大な開発負担を負わせている。
3. 特に、正統 Shift JIS ロケール sjis では、0x5C=U+00A5 というマッピングのために、オープン系プログラム(C言語、Java など)の動作が保証されない。cp932 などでは問題ない。

詳しくは、次の記事を参照。

[Linux での シフト JIS サポート](#)

ユーザの Shift JIS 資産

しかし、今日の日本のユーザ環境では、ほぼ必然的に、既存のコンピュータシステムを置換する形、ないしは、並存する形で Linux が導入される。既存の Windows/UNIX からの移行ではユーザの Shift JIS 資産をどう活用するか、また、既存の Windows/UNIX と並存した環境における設置では、それらシステムとの間のデータ交換で Shift JIS データをどう扱うかが重要な課題となる。この状況は、何も Linux 固有の問題ではなく、MacOS などでも同様である。

ここでは、Linux における Shift JIS データの取り扱いについて、汎用的なケースと、特殊なケースについて対処方法を述べてみる。

(1)データ移行の汎用論

Shift JIS データを含むいろいろなコード系データを安心してデータ変換できる汎用ツールとして iconv が存在する。Linux では SJIS, EUC-JP, JIS, EBCDIC, Unicode の変換をサポートしている。iconv は、(a)glibc に含まれるライブラリ関数、(b)libiconv (外部ライブラリ)として結合する関数、(c)iconv コマンドとして利用するものの 3 通りある。

しかしながら、Shift JIS のサポートの範囲については、標準では、ベンダ定義のコードに対するサポートがなされていないことに注意が必要である。そこで、当該システムにおいて、ベンダ定義の文字コードを使用し、Unicode 等にマッピングを行うシステムを実現する場合には、それぞれの

コードに対する改造や対応を必要とする。実現方法は次の2種類がある；

1. iconv ユーティリティおよび libiconv の改造を行う。
2. glibc, libiconv の定義テーブルに独自定義のテーブルを追加する。

iconv_open(3) に指定する、codeset は、次の3つの要素を包含しているため、それぞれの codeset がどのような内容を含んでいるのかという事を知っておく必要がある。

1. CES (Character Encoding System): (文字)符号化方式
2. CCS (Coded Character Set): 符号化文字集合
3. CCS と UCS(Unicode) とのマッピング

GNU libc (glibc) の codeset として Shift JIS 符号化方式(CES) は、sjis、cp932、ibm943 というのがある。Windows/UNIX などのシステムでは、sjis はほとんど使われておらず、もっぱら cp932 や ibm943 が使われている。

Appendix 1 は、ベンダ定義コードのマッピングの状況である。このような技術情報は、各ベンダー、システムインテグレータ、ソフトウェアベンダによって共有されており、彼らの移行サービスを利用することにより、安心してデータ資産・プログラム資産の移行を行うことができる。

(2)データ交換

ネットワークやデータ媒体を通じて、日々データを交換するようなケースでも、ユーティリティ iconv が利用できる。iconv のベンダ固有文字への対応の必要性は、(1)の記述と同じである。

しかし、インターネットツールでは、個々のアプリケーションが多様なコード系に対応しているケースも多い。たとえば、ブラウザ Mozilla Firefox では、HTML に記述された、content="text/html; charset=utf-8"により、さまざまなコード系のコンテンツを正しく表示できるようになっている。

(3)Linux の現状 (Shell スクリプト)

Linux のカーネルは、コード系への依存性はない。例えば、ファイル名として Unicode や Shift JIS の文字列を使うことも許されている。

しかし、問題は Linux のコマンドやライブラリでの扱いである。Shift JIS の文字符号化方式(CES)では、英数字1バイト、日本語2バイトという単純な論理で文字を処理できたところ、Unicode では、日本語が2~4バイトと可変長となる。文字を扱うライブラリ関数は、Unicode を扱うことを基本としているため、本ライブラリ関数を使ってインプリメントされた Linux システムコマンドでは、ファイルデータの中の文字処理や、ファイル名の処理で、Unicode は正しく扱えても、Shift JIS は扱えないことがあり、shell スクリプトも Shift JIS データを含んだファイルや Shift JIS で表現したファイル名の場合に誤動作することがある。Shift JIS データの処理は、「特別」な扱いとなり、Shift JIS 固有の問題が発生しても一般のディストリビュータや Linux メインラインで障害修正が受け付けられないことがある。

Unicode/Shift JIS のデータ処理の違いの解説としては、以下；

<http://itpro.nikkeibp.co.jp/article/COLUMN/20070221/262658/>

コンソール、プリンターなど、文字を扱う周辺装置では、デバイスドライバーが適切に対応している。

Appendix 2 は、Linux の修正対応がなされず、回避策がとられた例である。

[IBM 様の参考資料]

<http://www->

[06.ibm.com/jp/domino01/mkt/cnpages7.nsf/ec7481a5abd4ed3149256f9400478d7d/4925722f004efe9249257341002bf237/\\$FILE/Linux%E6%96%87%E5%AD%97%E3%82%B3%E3%83%BC%E3%83%89v1.05.pdf](http://www-06.ibm.com/jp/domino01/mkt/cnpages7.nsf/ec7481a5abd4ed3149256f9400478d7d/4925722f004efe9249257341002bf237/$FILE/Linux%E6%96%87%E5%AD%97%E3%82%B3%E3%83%BC%E3%83%89v1.05.pdf)

(4)オープン系のプログラム資産(C/C++・Java・PHP・Perl など)

Shift JIS では、日本語文字の 2 バイト目に 0x80 未満 (ascii コード域) が現れる。このため、「ソ」、「構」などの 2 バイト目に ascii バックスラッシュ文字 (0x5c、日本語キーボードでは ¥) 出現し、C、Perl、shell などの多くのオープン系プログラム言語がこの 0x5c をエスケープ文字として処理するために、上記のような文字を含んだソースコードが正しく処理できない。いわゆる「ダメ文字」問題。この問題の解決のために個別のパッチを用意している例もある。

[HP 様の参考資料]

ロケールに影響する問題

<http://docs.hp.com/ja/5187-2245/apbs12.html>

※パッチを当てて対応。この場合、Apache サーバの mod_perl としては利用できない。

ーC 言語 (gcc) の Shift JIS 対応

gcc バージョン 3.4 以上の開発環境において、C 言語で Shift JIS を使用したプログラムを Linux に移植する場合には、次の回避策が存在する。

1. 文字列リテラルに Shift JIS 文字列を記述している場合 gcc では、次のオプションを付ける事でコンパイルが可能になる。

```
-finput-charset=cp932 -fexec-charset=cp932
```

2. Linux の C 言語ワイド文字は、Unicode となっていて UNIX の CSI (Code Set Independent) モデルとは異なるので、ワイド文字を使ったソフトウェアの UNIX からの移植には注意が必要。

ーJava アプリケーションでの対応

Shift JIS をソースコードないしは IO コードに使用した Java アプリケーションの、Linux への移植については、次のパターンが考えられる。

1. Java アプリケーションにおいて、IO コード体系として Shift JIS を使用している場合
2. Java アプリケーションにおいて、Shift JIS をソースコードの記述に使用している場合

これらについて、次のような問題点と回避策がある

1. JDK における Shift JIS と Unicode のマッピングの問題

(参考:http://www.ingrid.org/java/i18n/encoding/shift_jis.html)

(参考:

<http://www.atmarkit.co.jp/fjava/rensai3/mojibake02/mojibake02.html>)

既存のアプリケーションが、どの JDK バージョンを使用していて、文字コードがどのように指定されているかによって、対策を変える必要がある。Java 2 SDK 1.2-1.4.0 の JDK から移植する場合は、どのような OS であっても注意が必要 (Java 固有の問題)。また、特に、文字コードとして "Shift_JIS" が指定されている場合には、OS によって動きが違う事が指摘されているので、注意が必要。

2. ソースコードの記述 (コメント含む) に Shift JIS を使用した場合

```
javac -encoding Shift_JIS foo.java
```

のように、文字コードを明示的に指定する必要がある。とくに Windows 環境からの移植では、"Windows-31J" 文字コードを指定する必要がある。

(5)レガシー系プログラム資産 (COBOL、帳票、など)

文字コードを Unicode に切り替えた上で COBOL 等のプログラムを継承する方法と、Shift JIS を維持したままプログラム資産を継続使用する方法とがある。前者に基づく移行の例は、COBOL コンソーシアムにて「[富士通様の COBOL to COBOL マイグレーション事例](#)」として報告されている。最大の問題は、Unicode 採用により日本語データのバイト長が変わることがあり、COBOL などのプログラム論理への影響が出て、プログラムテストの負荷が大きくなることである。後者では、そのような負担のないことがメリットであるが、(3)や(4)のような他の運用とのデータ変換や整合性の問題が付きまとうことになる。

(6)DBMS

Oracle/MySQL/PostgreSQL などの DBMS では、すでに Unicode に対応しているため、UTF などでデータ保存が可能である。さらに、Shift JIS など多様なコード系のデータ保存にも対応している。入力するプラットフォーム、利用・表示するプラットフォームに則してコード系の選択ができる状況にある。Unicode に対応していない古いバージョンから、現在のバージョンに移行する場合に、文字コードを意識した移行が必要になる場合がある。DBMS から取り出した Shift JIS のデータを、Linux 上で利用・表示する場合には、(2)や(3)と同じ配慮が必要となる。

(7)フォントの対応

Shift JIS 対応に作製した外字フォントの移行の課題がある。まず、Unicode 文字集合に対応する文字がないか確認の上で、該当する文字が存在しない場合は Unicode 外字域に割り当てて、作製した外字フォントを登録する対処が必要となる。

(8)ターミナルエミュレーターでの Shift JIS ベンダ定義文字

xterm,gnome-terminalなどの端末エミュレータにおける Shift JIS のサポートには次の制限事項がある。通常の ja_JP.SJIS ロケールでは、gnome-terminal と konsole では、Shift JIS のベンダ定義文字を表示できない。

1. gnome-terminal では、localedef -i ja_JP -f WINDOWS-31J ja_JP.SJIS でロケール設定する事でフォントが対応していれば表示可能
2. konsole では、qt の文字コード変換ルーチンにパッチを当てる事でフォントが対応していれば表示可能

まとめ

まとめると次のような現状となっているのである。

1. Linux が企業の基幹システムとしてますます重要性を増してきて、UNIX 等のミッションクリティカルなシステムの代替となっている。
2. 過去資産の継承の点から、Shift JIS のソフト資産、データ資産の継承は重要な課題である。
3. 反面、オープンソースがもつコミュニティーベースでの開発の中で、日本のユーザの利用状況に基づき Shift JIS の実装を試みても、完全な Shift JIS のサポートがなかなか提供できない。
4. Linux においても Shift JIS 系ロケールの設定はできるが、不具合があっても改善・修正が進まないことが多い。
5. Linux 以外の他のプラットフォームも含め、JIS 標準化の方向性と合致する Unicode へのデータ移行が推奨される。

Appendix 1 ベンダ定義コードのマッピングの状況

iconv_open(3) に指定する、codeset は、次の 3 つの要素を包含しているため、それぞれの codeset がどのような内容を含んでいるのかという事を知っておく必要がある。

4. CES (Character Encoding System): (文字)符号化方式
5. CCS (Coded Character Set): 符号化文字集合
6. CCS と UCS(Unicode) とのマッピング

GNU libc (glibc) の codeset として Shift JIS 符号化方式(CES) は、sjis、cp932、ibm943 というのがある。Windows/UNIX などのシステムでは、sjis はほとんど使われておらず、もっぱら cp932 や ibm943 が使われている。

	sjis	cp932	ibm943
CES	シフト JIS	シフト JIS	シフト JIS
CCS	JIS X 0201	JIS X 0201	JIS X 0201

	JIS X 0208	JIS X 0208 NEC 特殊文字 NEC 選定 IBM 拡張文字 IBM 拡張文字 ユーザー定義文字	JIS X 0208 NEC 特殊文字 NEC 選定 IBM 拡張文字 IBM 拡張文字 ユーザー定義文字
CCS と UCS のマッピング	Unicode コンソール シリアムのマッピングを一部修正	MS 互換	IBM 互換

UCS とのマッピングの違いによって、「～」を変換できないなどの問題が発生するため、Windows の Unicode をシフト JIS 符号化方式に変換したり、シフト JIS 符号化方式を EUC-JP 符号化方式、ISO-2022-JP 符号化方式へ変換する際には注意が必要となる。

glibc の sjis, cp932, ibm943 の UCS(Unicode) とのマッピングは次のようになっている。

文字	シフト JIS のコード値	sjis のマッピング	cp932 のマッピング	ibm943 のマッピング
\	0x5C	U+00A5	U+005C	U+005C
～	0x7E	U+203E	U+007E	U+007E
—	0x815C	U+2015	U+2015	U+2014
～	0x8160	U+301C	U+FF5E	U+301C
	0x8161	U+2016	U+2225	U+2016
—	0x817C	U+2212	U+FF0D	U+2212
¢	0x8191	U+00A2	U+FFE0	U+FFE0
£	0x8192	U+00A3	U+FFE1	U+FFE1
¬	0x81CA	U+00AC	U+FFE2	U+FFE2

新たに、iconv に codeset を追加する場合は、上記のようなマッピングの違いの問題や、重複して定義されている文字の変換をどのように行うのかという事も考える必要がある。

Appendix 2 既知の問題と回避策の例

Linux で Shift JIS サポートを行うに当たっては、以下に示す問題が知られているが、互換性の維持の観点から根本修正は困難である。したがって、以下に示す回避策が推奨される。

1. ja_JP.shiftjisx0213 でロケール定義結果を確認するも、正しく設定できない。

```
LC_ALL=ja_JP.shiftjisx0213 locale
locale: Cannot set LC_CTYPE to default locale: No such file or directory
locale: Cannot set LC_MESSAGES to default locale: No such file or directory
locale: Cannot set LC_ALL to default locale: No such file or directory
LANG=POSIX
LC_CTYPE="ja_JP.shiftjisx0213"
LC_NUMERIC="ja_JP.shiftjisx0213"
LC_TIME="ja_JP.shiftjisx0213"
LC_COLLATE="ja_JP.shiftjisx0213"
LC_MONETARY="ja_JP.shiftjisx0213"
LC_MESSAGES="ja_JP.shiftjisx0213"
LC_PAPER="ja_JP.shiftjisx0213"
LC_NAME="ja_JP.shiftjisx0213"
LC_ADDRESS="ja_JP.shiftjisx0213"
LC_TELEPHONE="ja_JP.shiftjisx0213"
LC_MEASUREMENT="ja_JP.shiftjisx0213"
LC_IDENTIFICATION="ja_JP.shiftjisx0213"
LC_ALL=ja_JP.shiftjisx0213
```

2. キャラクターマップが正しく設定されない。

```
LC_ALL=ja_JP.shiftjisx0213
linux-xrsls:~ # locale charmap
locale: Cannot set LC_CTYPE to default locale: No such file or directory
locale: Cannot set LC_MESSAGES to default locale: No such file or directory
locale: Cannot set LC_ALL to default locale: No such file or directory
ANSI_X3.4-1968
```

原因: 上記の原因は、本来 locale -a で ja_JP.SHIFT_JISX0213 と表示するべき定義を ja_JP.shiftjisx0213 と表示していること(“_”の有無に注意)に起因している。これは、glibc 内の定義ファイル記述ミスによるものである。

コードレベルの回避策が 2006 年に提案されたが、適用されていない。glibc で初期の jisx0213 定義リリース時に対応されていれば、取り込まれた可能性があるが、時期的に対応不可という判断であった。

(出典:http://sourceware.org/bugzilla/show_bug.cgi?id=3140)

回避策

ja_JP.SHIFT_JISX0213 として定義することで回避可能である。しかしながら、ja_JP.SHIFT_JISX0213 を適用してもウィンドウシステムを提供する X.org では、非対応になるため、I/O 処理を行う場合のみ、ja_JP.SHIFT_JISX0213 を定義して使用することが現時点での最善策とる。

Appendix 3 参照 Web リソース一覧

Shift JIS コードとは: (Shift JIS コードの歴史) Wikipedia

<http://ja.wikipedia.org/wiki/SJIS>

異体字シーケンス インプレス INTERNET Watch

<http://internet.watch.impress.co.jp/cda/jouyou/2008/09/10/20794.html>

Linux での Shift JIS サポート slashdot

<http://slashdot.jp/~numa/journal/262697>

Unicode/Shift JIS のデータ処理の違いの 日経ITプロ

<http://itpro.nikkeibp.co.jp/article/COLUMN/20070221/262658/>

IBM 様の解説資料

<http://www->

[06.ibm.com/jp/domino01/mkt/cnpages7.nsf/ec7481a5abd4ed3149256f9400478d7d/4925722f004efe9249257341002bf237/\\$FILE/Linux%E6%96%87%E5%AD%97%E3%82%B3%E3%83%BC%E3%83%89v1.05.pdf](http://www-06.ibm.com/jp/domino01/mkt/cnpages7.nsf/ec7481a5abd4ed3149256f9400478d7d/4925722f004efe9249257341002bf237/$FILE/Linux%E6%96%87%E5%AD%97%E3%82%B3%E3%83%BC%E3%83%89v1.05.pdf)

HP 様の Perl 関連参考資料

<http://docs.hp.com/ja/5187-2245/apbs12.html>

Java のコード系について

Ingrid

http://www.ingrid.org/java/i18n/encoding/shift_jis.html

アットマーク IT

<http://www.atmarkit.co.jp/fjava/rensai3/mojibake02/mojibake02.html>

富士通様の COBOL to COBOL マイグレーション事例 COBOLコンソーシアム

http://www.cobol.gr.jp/knowledge/material/080905_report/02.pdf

本検討の貢献者

(敬称略)

有馬 浩一(ノベル)

大釜 秀作(住友電工)

小菌井 康志(The Linux Foundation)

工内 隆(The Linux Foundation)

高橋 千恵子(NEC)

田郷 明(レッドハット)

伊達 政広(富士通)

橋本 尚(日立製作所)

藤田 祐治(レッドハット)

三浦 広志(NTT データ)

森陰 政幸(ターボリナックス)

森山 将之(ミラクルリナックス)