Unicodeの最新動向

織田 哲治　　木戸 彰夫　　小田 明
日本アイ・ビー・エム株式会社

国際符号化文字集合 Unicode 1.1 及びISO 10646-1 が制定されてから、既に5年余りが経過しようとしている。現在も、ISO やUnicode コンソーシアムによって文字の追加など符号化文字集合規格としての保守作業が行われる傍ら実装に向けての整備／検討が進められている。本稿では、Unicode の内容解説から始め、その実装についての考察、他の漢字圏の符号化文字集合との関連を述べながら Unicode の最新動向をまとめる。また昨今、その浸透とともにますますあちこちで聞かれる Unicode に纏わるいくつかの批判的な議論についても技術的解説を試みる。本稿は、1章と全体のまとめを織田が、2～4章を小田が、5章を木戸が執筆した。

Current Status of Unicode

Tetsuji Orita　　Akio Kido　　Akira Oda
IBM Japan, Ltd.

Unicode 1.1 and ISO 10646-1 have been generally available on 1993. Since then, the effort to maintain these character set standards, such as new script addition, has been continuing by ISO and Unicode Consortium. In parallel with it, several activities to accelerate its implementation are also in progress. This report describes the current status of Unicode. It includes the introduction of Unicode, some approaches of its implementation, the relation with other Asian countries' coded character set movement, and some discussions on Unicode, especially in Japan.

1．Unicodeとは

Unicode[1]は、2バイト(16ビット)で世界中の言語で使われている文字をできるだけ多く含めようとする符号化文字集合である。Unicode の開発プロジェクトは、ソフトウェアの各国語実装をより効率よく行うことを目的として、1988年に、米国のいくつかのソフトウェアメーカーのエンジニアが集まって結成された Unicode コンソーシアムというグループによって始められた。このグループは1991年にUnicode Inc.となり、その年にUnicode 1.0 を発表した。

国際統一符号化文字集合の検討は、ISO(国際標準化機構)でも既に1983年頃から始まっており、1990 年には ISO 10646-1 規格 DIS(Draft International Standard)案が提出されている。このDIS 案は、漢字圏をはじめとする各国の文字集合を独立して配置した符号化文字集合であったが、その後、1991年にUnicode Inc.とISOが話し合いを持ち、Unicode とISO 10646-1 間で互換性のあるコード体系とすることが決定された。その結果、1993年にUnicode Inc.は、Unicode 1.1を発表し、ISOでは、 10646-1[2]が制定された。Unicode は、その後、1996年にUnicode 2.0 として改訂されているが、ISO では補遺(amendment)としてこの改訂に対応している。このUnicode 2.0 での改訂で、6,656文字のハングル文字の削除、及びそれを含んだ 11,172 文字のハングル文字の新たな区点位置への追加が行われた。(「4. 他の漢字圏の国々の動向」参照)

現在、Unicode Inc.は、ISO 10646-1 を保守管理するISO/IEC JTC1/SC2/WG2のリエゾンメンバーとなっており、今後も Unicode と ISO 10646-1 間で互換性を維持していくことになっている。

ISO 10646-1 の符号空間は、4オクテットの正規形式と 2 オクテットの BMP(Basic

Multilingual Plane)形式として表現され、それぞれをUCS-4(Universal Character Set four-octet), UCS-2(Universal Character Set two-octet)と呼ぶ。4 オクテットの符号化文字集合は、4 次元空間のイメージとしてとらえられる。つまり、256(row) x 256(column)の 2 次元の面(plane)が 256 面集まった群(group)、その群が 128 個集まって構成される。そのうちの最初の面を 2 オクテットの BMP 形式として表現した仕様が Unicode となる。現在は、BMP のみに文字の割り当てが行われている。

Unicode 2.0 では、約 39,000 文字が含まれている。Unicode の文字集合は、それまでの文字コードの国際規格及び各国の規格を含むように、また、それら以外で実装されている文字も参照しながら定義されている。日本の文字コード規格には、JIS X0201,JIS X0208 及び JIS X0212 があるが、それらは全て Unicode に含まれている。以下に、Unicode の主な特徴について述べる。

・Unification(文字統合)

Unicode には、20,902 文字の統合漢字が規定されているが、これらはその採用元となった各言語の漢字の原規格に収容されている漢字をある規則で統合し、配列したものである。ここで用いられたのが統合規則(unification rule)である。これは、言語間で規格中の異体字の違いが実字形(actual shape)、つまり単なる表示字形上のデザインの違いであるか、あるいは抽象字形(abstract shape)、つまりデザインをこえた字形の違いであるかを考慮し、実字形の違いと判断される場合のみ統合し、同じ区点位置に並列配置している。ISO 10646-1 Annex S (Amendment 8)には、実字形の違い、抽象字形の違いの具体例と統合規則のより詳細が示されている。基本的には、この統合規則により統合漢字が規定されているが、既存の原規格との整合性を維持するために、例外規則として、一つの規格内で既に別区点位置に割り当てられている漢字はたとえそれが統合規則により統合される漢字であっても、別々の区点位置を割り当てることになっている。(source code separation)

・合成文字列

それまでの符号化文字集合と同じように、Unicode は、その符号化空間の一つの点に一つの文字を割り当てており、それが文字データ処理の単位となるが、その中に結合文字として分類される文字集合が含まれている。この結合文字(combing character)には、欧米の文字で用いられるアクセントなどのダイアクリティカルマークや、タイ語やラオス語など東南アジアの文字が含まれる。一つ以上の結合文字は、基底文字(non-combing character)と結合されて合成列となる。合成列は、Unicode の文字集合の一部ではないが、既に Unicode 上に含まれている基底文字と同じ形で表現される場合や、複数の結合文字が異なる順序で結合された場合に二つの合成列が同じ形に表現される場合がある。

・UTF-8

ISO あるいは PC の符号系を実装している既存のシステムを考えると、00～FF オクテットを使用している UCS を使用するためには、ファイルシステムなどにおいて、全く新たなシステムの実装を行わなければならない。そこで、既存のシステム上にそのまま UCS の文字集合を実装するための UCS の代替符号化表現として考えられたのが、UTF-8 である。UTF-8(UCS Transformation Format 8)では、Unicode/UCS-2 の区点位置 0000～007F の ASCII 文字を 1 バイトの 00～7F とし、以降の文字を、各バイトが 00～7F オクテットを含まない複数バイトで符号化したものである。UTF-8 は、既存の符号系との互換性のため可変バイト長になっているが、主に既存のファイルシステムに UCS の文字を格納する際の使用を考えているので、この場合も、後述するプロセスコードとしての利用では Unicode/UCS-2 が使われる。

・UTF-16

当然ながら Unicode では、ISO 10646 の BMP 以降の面の文字は表現できない。そこで、Unicode を実装したシステムから、そのままの拡張でこれらの面の文字を使用するための符号化表現とし

て考えられたのが、UTF-16 である。UTF-16(UCS Transformation Format for 16 planes of Group 00)では、Unicode 上に予約されている区点位置 D800～DBFF 及び DC00～DFFF から各々2 オクテットを使用し結合することで表現可能な 1024 x 1024 個の符号を、BMP に続く 16 個の面の文字集合(16 x 65535)の符号化のために利用する。

現在も、ISO SC2/WG2 では、ISO 10646-1 への追加文字の検討が行われている。これには、少数民族の言語、古典・古語などの文字の他に、約 6,000 文字の漢字の追加提案が含まれている。この漢字は、ISO SC2/WG2 のもとに活動している IRG(Ideographic Rapporteur Group)が、ISO 10646-1 制定直後からずっと検討してきたものであり、ISO 10646-1 Amendment 5 で削除されたハングル文字の区点位置に割り当てられることになっている。この中には、約 600 文字の日本提案の漢字が含まれている。これにより、BMP(plane 0) / Unicode の 65,536 文字分の区点位置の中で、大きな連続区域の文字の割り当てはほぼ終わろうとしており、次に plane 0 以降の面の規定作業が始まろうとしている。現在は、plane 0 上の文字の補充として、plane 1 に非漢字、plane 2 に漢字をそれぞれ追加するという枠組みが決まっている。実際の追加文字の検討はこれから進められる。この plane 1 や plane 2 は、Unicode からは、UTF-16 によって使用する。

2．利用形態から見た実装

Unicode の実装についてはいろいろな観点から述べることができる。例えば、文字の入力方法、フォントの実装方法など Unicode をサポートするためのシステムの実装方法も一つのトピックスである。本稿では、Unicode の適用分野を考えるために、文字コードとしての利用形態の観点にしぼってその特徴と状況について述べる。

ところで、情報処理システムに新しいコード体系を持ち込む場合を考えてみると、全く新しくすべてを構築する場合は別として、通常、既存のシステムとの共存性・互換性および相互運用性、また既存システムからの移行が考慮点となる。実際問題、現在安定稼働している基幹業務系のシステムをある時点で突然すべて Unicode ベースに移行するということは考えにくい。というのも、技術的には、移行作業は膨大なデータベースをコード変換する作業および適用業務プログラム内でのコード体系に依存する部分の変換作業であるが、これらの作業に対する投資がビジネス上の何らかの利益につながらない場合は移行は行われないからである。

このことを念頭において、以下では、Unicode を(1)情報交換用文字コード(2)プロセスコード(3)データベースでの利用といった視点で考察し、最後に新しい世界での利用について述べる。

2．1 情報交換用文字コードとしての利用

現在のように、いろいろな機種の(したがって文字コード体系の異なる)コンピュータがネットワークを介してつながる環境を考えると、これらの情報交換用文字コードとしての Unicode の利用がまず考えられる。すなわち、各既存システム内は、それぞれ従来どおりの文字コードを使用しながら、情報交換用コードとしてネットワークに情報を送信する時には送信側は Unicode に変換し、受信側ではこれを受信側の文字コードに変換することになる。例えば、国際的な電子商取引などのプロトコールの情報交換用文字コードに Unicode の利用を検討するまたは採用するといった動きがある。これは、Unicode が多言語の文字コード（この意味については「5．Unicode に対する様々な批判」で後述）であることから、いろいろな言語の文字コードをサポートすることを考えるよりも簡単であるといった理由である。特に、ISO 8859-1 (Latin 1)や ASCII 文字セットだけを想定していたプロトコールなどで、各国語をサポートするために検討される傾向にある。

現在、インターネットの日本語環境で使用されている情報交換用文字コードとしては、シフト

JIS コード、EUC (Extended Unix Code)、ISO 2022-JP (RFC 1468) があるが、Unicode が国際的な範囲で採用されていくと、今後日本語環境でも Unicode が普及していくことが予想される。

2．2 プロセスコードとしての利用

シフト JIS コードは英数文字などその基本部分は米国のコンピュータの文字コードと同じであり、その考案の意図はなるべく米国のソフトウェアに手を加えずに日本語を使用できるように(且つ JIS X 0201 を尊重する形で) 決めたとのことであり、シフト JIS コードがパーソナル・コンピュータで普及したのもこのためであるとも考えられる。しかしながら、シフト JIS コードは、1文字を1バイトまたは2バイトであらわすコード体系であり、米国で特別の注意を払われずに開発されたソフトウェアを日本語環境に移植するときには、この1バイト2バイトの混合した文字ストリングを正しく処理するための手直しが必要となる。もちろん、米国の開発者がこの文字ストリングの処理を開発時に組入れていれば、手直しの手間はかからない。こういった場合に1バイト2バイト混合の文字ストリング処理を誤りなく行う有効な手法の一つは1文字をすべて2バイトであらわすように正規化して2バイト固定長の文字列として扱うことである。これは、C 言語などにみられる wchar_t といったデータタイプの考え方である。この正規化のデータ表現は内部表現として実装系によって定義されるのであるが、この表現として、Unicode が使われる例が多くなっている。

ところで、開発者が必ずしもこういった正規化手法を使用してくれるとは限らず、あとで思わぬ手直しが必要となる場合が生じる。国際化のための文字ストリングを開発者が意識せずに必ず使用するという考え方を推し進めていくと、すべての文字がはじめからすべて2バイト固定長であればという考えに到達する。実のところ、Unicode コンソーシアムが Unicode を考案しはじめたときのねらいはここにあったといってよい。後述するタイ語などにおいて文字合成をすることや、さらに多くの文字を2バイト体系で表現するための UTF-16 などがこの固定長の考えに混乱を与えているようにみえる。世界の文字を集めるといったことによって、Unicode が多言語処理に関する過剰な期待や逆に批判を負うことになるが、Unicode をプロセスコードとしての価値でみつめると案外妥当な評価で位置付けられるように思われる。

2．3 データベースでの利用

現在すでに適用業務として既存の文字コードで膨大なデータやドキュメントを保管している場合、この章のはじめに述べたような理由により、積極的に Unicode に移行していくことは考えにくい。 したがって、既存のデータベース自体は既存の文字コードで存続し、データサーバーからネットワークへ出て行くところで必要であれば Unicode へ変換するといったシナリオになる。この利用は、情報交換用コードとして、Unicode を用いているのであって、データベースの文字コードとして用いている訳ではない。

最近では、Unicode をサポートするデータベース製品やファイルシステムが増えてきていることから、今後、Unicode を利用した新しい適用業務が開発され、Unicode を情報交換用コードとして用いたトランザクションやドキュメントがそのまま保管されるようになるとデータベースにも、徐々に Unicode が普及していくと思われる。

2．4 新しい世界での利用

ここまで、既存のシステムが存在するなかで、Unicode がどのように利用でき、どのように普及していくかを考察してきたが、ここで新しいパラダイムとしての Java での利用について考えてみる。Java は Write Once Run Anywhere といった、一度開発されたプログラムは Java 実行環境があれば、どんなプラットフォーム上でも実行可能という考えを前面に押し出し、このためにアーキテクチュアニュートラルなバイトコードを生

成し、それをJava実行環境でインタープリートするという設計になっている。ここで、文字コードに関して、プラットフォームごとに符号化体系が異なっていると、文字コードに関するプラットフォーム依存となってしまうが、Javaは Unicodeを採用することによりこの問題を解決している。現状では、既存のシステム上にJava仮想マシンが実装され、必要であれば、文字入力、表示出力、ファイル入出力のところで既存のコード系とのコード変換が行われる訳であるが、いずれにせよ、Javaという新しい世界の中では、Unicodeに統一されている。

3. UnicodeのJIS規格化と文字集合の拡張

日本では、1993年に日本規格協会のもとにUCS調査研究委員会を発足させ、ISO/IEC 10646-1のJIS規格化にとりくみ、1995年1月にJIS X 0221「国際符号化文字集合(UCS) - 第1部 体系及び基本多言語面」を発行している。これは、ISO 10646の日本語翻訳に附属書として日本文字部分レパートリを規定し、これまでJISの文字集合の標準であったJIS X 0201「7ビット及び8ビットの情報交換用符号化文字集合」、JIS X 0208「7ビット及び8ビットの2バイト情報交換用符号化漢字集合」、JIS X 0212「情報交換用漢字符合ー補助漢字」の表内文字との対応を参考としてあげている。一方、日本規格協会のもとJCS (Japanese Coded Character Set) 調査研究委員会は、1997年のJIS X 0208の改訂の原案作成を行ったが、このときには、文字の種類を増やさない方針で、この符号化文字集合にこれまで定義した漢字の典拠を再調査し、漢字に対する包摂の考えを明示的に導入し、包摂基準を設けている。このJCS調査研究委員会は、引き続きJIS X 0208とともに用いる拡張文字集合の調査検討に入っている。現在のJIS X 0208のいわゆる第1水準および第2水準に対して、この拡張では、第3水準、第4水準をそれぞれ2～3千字程度、合計5千字程度まで定める計画で始められている。この数字は、シフトJISコードの拡張限界から割り出されたもので、実際には高校の教科書程度を表記できるという目標と、典拠の確かなもの、また正字に対する包摂といったことを検討した上で決められると思われるので、現段階で公表されている情報からは最終的な数はまだ予測できない。俗に、第3水準と呼ばれることのある、JIS X 0212 補助漢字とは独立に定められるので、新しい拡張文字集合に、JIS X 0212に含まれている文字が入ることもあれば、入らないこともある。これは前述の基準でJIS X 0208に対して文字集合の拡張を考えるからである。また、これらの文字は新規格制定後、ISO 10646に追加提案されると思われる。

4. 他の漢字圏の国々の動向

Unicodeには、タイ、ラオスといった東南アジアの国々の文字も含まれているが、日本からみて一番気になるのは、漢字の統合(Unification)がされた、中国(中華人民共和国)、台湾(中華民国)、韓国であろう。ISO/IEC 10646は、中国ではGB 13000、韓国ではKS C 5700として国の標準規格となっている。台湾では、特に台湾の規格とするような別の名前または番号をふっていない。ところで中国では簡体字を台湾では繁体字を用いている(Unicodeでは簡体字は繁体字とは別の符号位置が割り振られている。繁体字に対応する簡体字がある場合と、両国で共通、すなわち基本的に昔のままの場合がある。)が、両国間でこれらの両方を使いたいとい要請が、Unicodeを推進する動機になっているように思える。韓国は、漢字にではなく、ハングル文字に対して強い変更要求をだした。Unicode 1.1すなわちISO/IEC 10646は6,656文字のハングルを定義していたが、Unicode2.0またISO/IEC 10646 Amendment 5でハングルのすべての11,172文字を定義した。このとき、4,516文字を追加したのではなく、ハングルのコンポジションロジックに沿った並びとなるように、当初の6,656文字を削除した上で新たに連続域に11,172文字を割り当てるという変更要求を通した。ISOでは互換性

の観点から、一度割り当てた文字の符号位置を変更するということは行わないだけにこの変更には韓国の標準グループの強い働きかけがあったといえる。

これらの国々では、PC(パーソナルコンピュータ)の文字コードを拡張して、Unicode の文字集合を含める動きがある。中国では、GB 2312 の標準に基づく PC コードを拡張して、Unicode の漢字部分を全部含めたものを実装し始めている。台湾では、Big-5 といわれる PC コードに Unicode の文字を加える拡張が議論されてきてはいるが、PC コードの拡張より、必要であれば Unicode 自体を使うという流れになりつつある。韓国では、11,172 のハングル文字を全部含めた PC コードへの拡張がメーカでは考えられているが、国の標準機関は Unicode そのものを推進しているようである。

5．Unicode に対する様々な批判

Unicode、または ISO/IEC 10646 は、欧米中心に作成されたものであり、漢字文化を破壊するものである等の批判をよく耳にするが、それらの批判を分類してみるとおおよそ以下のようになる。①規格制定時におけるある種強引ともいえるプロセスに対する批判、②文字合成シーケンスに関するもの、③漢字の統合(Unification)および多国語処理に関するもの、④文字の不足を訴えるもの。上記の批判のうち①は技術的なものであるとは言い難いので、本稿での議論から外すことにする。以下、②から④の批判に対して、情報処理の側面から解説する。文字符号の議論を行う際、常に問題になるのは、「文字」とはいったい何であるのか、もしくは、情報処理上何を「文字」として取り扱えば都合がよいのかということである。本稿では言語学の観点から各国語における「文字」とは何か、という議論は行わないことにし、あくまでも情報処理上の利便性からのみ「文字」を見ることにする。また Unicode という符号化文字集合自体、国ごと言語ごとに異なる「文字」概念の上に立脚した、各国の国内規格の中で規定されている「文字」を寄せ集めて作成されたマルチスクリプト符号化文字集合であるので、Unicode 自体に統一された「文字」概念が存在することも仮定しない。情報処理上の利便性に対する議論は、対象のスクリプトを母国語とする利用者にとっての利便性（日本人が日本語の文字を使用する場合）と、そのスクリプトに知識を持たない外国人利用者にとっての利便性（日本人が日本語の文章の中にキリル文字を混ぜる場合）の二つの観点から議論することにする。また上記の二つの観点から導かれる結論はおうおうにして相反することがあるので、本稿での議論は、Unicode の取った技術的な選択に対して、「正しい」、もしくは、「誤り」であるとの二値的な判断は行わない。

Unicode で言う文字とは、Unicode の符号表の上で独立した符号値を与えられた「もの」である。つまり、アクセント付アルファベットも、アクセント無しのアルファベットも、合成用のアクセント記号も、それぞれ独立した文字ということになる。ここで話をややこしくするのは、印字字形としては前者のアクセント付文字と、後者のアクセント無しのアルファベットとそれに続く合成用のアクセントからなるシーケンスは全く同一になるかもしれないということである。同様のことは、東南アジア各国で使用される文字についても言える。タイでは子音、母音、トーン記号を各々独立した文字であると考えているが、それらの文字の印字字形を見るかぎり、タイ語を母国語としない外国人の目には、あたかも母音や子音が「文字」を構成する一要素であり、それらの文字要素が複数組み合わされてはじめて一つの独立した「文字」となるように見える。印字字形に注目して「文字」という情報処理上の最小の情報要素（アトム）を考えるなら、Unicode の文字あたり固定長符号というコンセプトには嘘があることになる。しかしながら、テキストエディター等で行われている文字列処理を考慮に入れて「文字」を見るならば、かならずしもアトムが印字字形であるとは言えない。タイ語のテキストエディターでよく行われる文字の削除処理では、ＢＳキーでは子

音、母音などのUnicodeでいう「文字」が、ＤＥＬキーでは、一印字字形が削除されるからである。外国人にとっては、たぶん印字字形を「文字」として処理した方が都合がよい。また、タイ人にとって現在のタイ語のテキストエディターの処理が適当であるのか、筆者は判断を下すことが出来ない。もしかすると現在の処理方法は、８ビット符号の制約、もしくはバイト処理の制約から生まれ（ＢＳの場合）、多バイト文字処理もしくは「文字」単位での処理への移行期であるのかもしれない。しかし外国人の判断でタイ語の「文字」は印字字形単位であるべきだと主張することは、①でいう外国人の横暴をその「文字」を母国語とする人々に押し付けることになりかねない。

Unicodeは、明らかに異なる図形表現を持つ日本語・韓国語・中国語・ベトナム語の「文字」の統合(Unification)を行ったので、文字符号からその文字がどの「言語」に属するのか判断できない。またワードプロセッサ等のアプリケーションで印字を行った場合、異なる図形表現を持つべき日本語と中国語の文字（例えば「骨」という文字）が同じ形で印字されてしまう事が多いので、Unicodeでは多国語処理ができないとの批判がある。ここでいう「多国語処理」を同時多国語処理という意味と解釈し、さらにその同時多国語処理の要求は、文書構造をもたない単純な文字符号列からなる、いわゆるフラットテキスト上での処理要求であると解釈するならば、この批判はもっともである。しかし、この主張を行っている人が持っている電子文書のモデルと、Unicodeの開発者達が持っていた電子文書のモデルには違いがある。このモデルの違いを無視して議論を進めれば、議論はすれ違うばかりである。Unicodeの開発者達がもっていたモデルでは、同時多国語処理の対象となる電子文書は、構造を持った文書であり、文字列の照合順序等「言語」依存の情報処理の際参照されるべき「言語」属性は、単純文字符号列からなる基礎構造の上位構造で与えられるものであった。言い換えるならば、基礎構造であるフラットテキストで行える「多言語処理」は、その時々において参照すべき「言語」属性を特定した逐次多言語処理、もしくは、前述した日本語の文章の中にロシア語の単語（キリル文字）を含めるような単一言語マルチスクリプト処理である。ではなぜ同時多言語処理に適さないUnicodeが「マルチリンガル文字符号」と呼ばれるのであろうか。詭弁のようではあるが、Unicodeが目指す「マルチリンガル」とは、「多言語処理用」ということではなく、どの単一言語で使用される文字集合から見ても文字集合の観点からはスーパーセットになっているため、特定の言語処理用ではなく、複数の言語環境（一時には一つ）で使用できる、という意味での「複数（マルチ）」である。マルチリンガルと言うと、多(マルチ）言語（リンギスティック)処理が可能であるような幻想を利用者に持たせがちなので、素直にマルチ（多）スクリプト（文字群）文字符号と言ってくれた方が、英語を母国語としない者にとって親切な表現になったのではないかと筆者は考える。

最後に文字の不足の問題について言及する。たしかに現在Unicodeに定義されている文字は、文芸書の記述および電子的な公開、古典文書の電子的な保存、人名および地名の適切な表示を行う場合、充分ではない。実際、ISO/IEC JTC1/SC2では、ISO/IEC 10646に対する文字の追加作業を行っている。さて、ではどのような「文字」が不足しており、何を「文字」として追加すべきなのであろうか。この場合にも、前述の「印字字形」と情報処理用の情報単位としての「文字」の問題が顔を出してくる。情報処理上の利便性の観点から「文字」の議論を行う場合、アプリケーションを特定して、そのアプリケーションの機能上の要求から「文字操作」の利便性を議論する方法が有効である。「文字操作」に対する要求は、アプリケーションの機能に依存するので、アプリケーションを特定しないと議論が発散してしまうからである。本稿ではアプリケーションの例として、文書情報の電子的な保存、検索、公開機能を持つ電子図書館を取り上げてみる。電子図書館には、どのような文書もできるだけそこに書かれてい

る情報を損なわずに、電子的に記録保存したいという要求がある。「文字」の持つ属性としては、意味、発生された時の音、図形（字形）情報などがあるが、図形情報としては、正字も異体字も誤字もそれぞれ判別可能な独立した「文字」として記録したいという要求が存在する。また電子図書館の持つ情報公開機能を考えれば、蓄えられた電子文書は、不特定多数の利用者への情報公開に際して紛れ込む情報ノイズを最小にしたいという要求もある。つまり、「文字」のもつ図形情報についても、蓄積された環境と利用者の表示環境において「文字」の形の変形が起こっては困るのである。しかしながら、極端な話、誤字・異体字を含めて「文字」の図形情報に注目した「文字」集合を作った場合、その集合の要素数は非常におおきなものになってしまい、それぞれの要素に個別に符号を割り振った符号表と図形情報（フォント）のセットを全ての環境で標準的にサポートするというのは、かなりコストの高い解決法であると言わざるを得ない。一方、電子図書館には情報の検索という機能がある。電子図書館の利用者の要求には我が儘なものがあり、「文字」の図形表現については字形の僅かな違いも表現してもらいたいとの要求がある一方、その検索に関しては「文字」の図形表現の僅かな違いをある程度無視して、曖昧な検索を行いたいとの要求がある。例えば内田百間の「けん」は、印字字形では「月」でなければこまるが、検索では「日」でもヒットしてもらいたいとの要求である。この相反する要求を「文字符号」という単一の符号系だけで解決するのには無理があるように思える。つまり、なんらかの方法で「文字」の図形情報と、検索用の「文字」情報を区別して処理する必要がある。実際、多くの情報システムにおいて概念的には、文字の図形情報はフォントで、検索用の「文字」情報は文字符号でというように区別して処理を行っている。その上で、検索および内容情報処理用の文字符号としては、字形の僅かな違いを包摂して「文字」の数を制限し、一方図形情報は、フォントとして「文字符号」から切り放しておくことにより、印字字形の多様性が「文字符号」の数の制約を受けないように工夫している。一見この住み分けはうまくいっている見える。ただし実際の実装では、欧米の文書処理においては、情報交換の際に引き起こされる文字の図形情報の劣化が「文字」の持つ意味情報に及ぼす影響が小さいという性質によりかかって、フォントのインデックスを文字符号で代用したり、フォント情報の交換を行っていない場合が多い。その為、図形情報としての「文字」の不足が検索・内容情報処理用の「文字符号」の不足と混同して議論されるということが起こるのである。勿論、情報検索および内容情報処理用の「文字」の不足（乱暴な言い方をすれば正字の欠字）は、文字符号表上にその欠字を追加することによって対処されなければならない。しかし実際 Unicode、もしくは ISO/IEC 10646 に寄せられた「文字（字形）」の追加要求を見てみると、Unicode の「文字」の包摂から見て既に定義されているどの文字との関係も見いだせない「文字」の追加要求（欠字の指摘）は、数百字のオーダーであり、符号表の空き領域に充分定義できるものであると聞いている。その他の字形の追加要求は、印字字形としての多様性を求め、かつ情報交換時においてその図形情報が劣化しないことを求めたものであると考えられる。よって、情報の蓄積および交換の際に使用される電子文書が利用者定義のフォント情報を取り扱えるようになれば、「文字符号」の拡張なしに満たすことが出来る要求であるように思える。

[参考文献]

[1] The Unicode Consortium, The Unicode Standard Version 2.0, Addison-Wesley, July 1996.

[2] ISO/IEC 10646-1:1993, Information technology-Universal Multiple-Octet Coded Character Set(UCS)-Part 1: Architecture and Basic Multilingual Plane, 1993.

[3] 吉目木晴彦他, "電脳文化と漢字のゆくえ", 平凡社, 1998.