
Unicode: support for multiple languages at the Ohio State University Libraries

*Laura Tull and
Dona Straley*

The authors

Laura Tull is Systems Librarian and Dona Straley is Middle East Studies Librarian at the Ohio State University Libraries, Columbus, Ohio, USA.

Keywords

Languages, Catalogues, Libraries

Abstract

Unicode is a standard for a universal character set for all of the scripts of the world's languages. It is one of the fundamental technological building blocks for international exchange of textual information, via computers. It is particularly important to libraries that house collections in many languages and written in various scripts. Ohio State University Libraries houses collections written in many non-Latin scripts including Arabic, Hebrew, Chinese, Japanese, Korean and Cyrillic. Providing patrons with access to these materials has become much easier with the incorporation of Unicode into library systems and software. This article describes what needs to be in place on personal computers and in the library system in order to take advantage of Unicode as well as providing some guidelines to troubleshoot problems when they occur.

Electronic access

The Emerald Research Register for this journal is available at <http://www.emeraldinsight.com/researchregister>

The current issue and full text archive of this journal is available at

<http://www.emeraldinsight.com/0737-8831.htm>

Introduction

Unicode(tm) is a standard for a universal character set that can encode all of the characters used in the various written scripts of the world's languages. Unicode is maintained by the Unicode(r) Consortium, composed mainly of computer companies such as IBM, Apple Computer, Inc., Adobe Systems, Inc., and the Microsoft Corporation as well as library organizations such as The Research Libraries Group, Inc., the OCLC Online Computer Library Center, Inc. and several library system companies. It is one of the fundamental technological building blocks for easy international exchange of textual information via computers. A variety of national, international and industry standards have developed over the years for encoding characters but many are incompatible with each other making the exchange of information between computer systems difficult. Some of these characters sets, such as the American Standard Code for Information Interchange (ASCII), are also severely limited in the number of characters they can represent because they are based on a seven or eight-bit single byte encoding scheme, which can only accommodate a maximum of 128 or 256 characters. ASCII works fine for the English language, but more complex encoding schemes using multiple bytes were developed in East Asian countries to accommodate the thousands of ideographic characters used for Chinese, Japanese and Korean. For example, the Chinese Character Code for Information Exchange (CCCII) maps characters to three seven-bit code units. The East Asian Character Code (EACC), a character set used in MARC records, is based on CCCII.

These numerous, incompatible character codes, often encoded the same characters. Other encoding schemes were limited in the number of characters they could encode. This led several interested parties, now part of the organization known as the Unicode Consortium, to develop a standard for a

Unicode(tm) – “Unicode is a trademark of Unicode, Inc.”; Unicode(r) – “Unicode is a registered trademark of Unicode, Inc.”; CJK(r) is a registered trademark of The Research Libraries Group, Inc.



universal character set. Version 1.0 was published as *The Unicode Standard* in 1991 (Unicode Consortium, 1991). It is based on a set of ten design principles that would address these problems. Version 4.0 will be published in 2003 (Unicode Consortium, 2003). The first design principle is universality. The intent was to develop a single character set that would include all of the characters used in both modern and historic writing systems. Unicode was originally designed as a 16-bit encoding system, which would allow for 65,536 characters. This is a huge increase over the maximum of 256 characters in some eight-bit encoding schemes. In Version 2.0, Unicode was expanded to accommodate over a million characters. Another important guiding principle, unification, means that one unique code represents each character even if that character is used in multiple scripts, as is the case for the Han ideographic characters that are used in Chinese, Japanese, Korean and historical Vietnamese. Another principle is that characters, not glyphs, are encoded. A character is basically an abstract concept, whereas, glyphs are the concrete visual representations of characters. Characters can have many different representations when written or displayed on a screen. For example, the capital letter "A" in the Latin alphabet has a printed form and a cursive form. It can also take on many different visual representations depending upon the typeface or font used. Some Arabic characters take on a different form depending upon whether the character appears at the beginning, in the middle or at the end of a word. The design principle is to encode the character and not its various visual representations. The form that the character takes on a computer screen, for example, is left up to the font or text rendering software.

Besides encoding characters, the Unicode Standard also addresses some other issues that arise when dealing with multiple scripts. The directionality of a script must be addressed in order to display characters in the proper order. For example, text in both Hebrew and Arabic is written from right-to-left. This is not a problem when all of the characters run in the same direction. But what happens if two scripts that are written in opposite directions are mixed in the same paragraph, for example? Unicode

addresses this situation by describing an algorithm and providing formatting codes to allow the text to be displayed in the proper order.

Another crucial issue that application developers must address is the searching and sorting of the characters. Library systems are perfect examples of applications that must provide searching and sorting capabilities. When a user enters a search for an English language title in a library catalog, for example, they expect the list of results to be sorted in alphabetical order. However, this is not a simple matter when it comes to supporting multiple scripts because the expected sort order can vary even among cultures that use the same written script. In some scripts, that use non-alphabetic characters, a totally different sort order may be employed. For example, a common sort order for the ideographic characters in Chinese is by the number of strokes in each character (i.e. characters with less strokes sort before characters with more strokes). Although Unicode provides a collation algorithm, which specifies an ordering for all of the characters, it cannot always produce the results that make sense to users. The Unicode Standard provides implementation guidelines to help developers deal with this complex issue.

Each character in Unicode has a unique value or code point, but in order for characters to be stored or transmitted in computer systems, this value must be mapped or transformed into a sequence of bytes or code units that computers can understand. This mapping is called a character encoding form. Unicode supports three encoding forms: UTF-8 (i.e. Unicode transformation format, eight-bit form), UTF-16 and UTF-32. UTF-8 encoding form uses a sequence of one to four bytes. Its use of single bytes provides ASCII compatibility since the 128 characters in ASCII have identical values in Unicode. UTF-8 is the encoding form supported by most Web browsers. UTF-16 uses a sequence of one to two 16-bit code units. UTF-32 uses one 32-bit code unit for each character.

Libraries that collect materials in many different languages, especially those that do not use the Latin script, such as Arabic, Hebrew, Chinese, Japanese and Korean will want to take advantage of Unicode. Users of these

collections want to be able to display the titles and authors of these materials in the native scripts in the library catalog as well as be able to search for materials in this manner. Unicode's capabilities become apparent when a record that has multiple scripts, such as a record for a book that contains Arabic, Hebrew and Latin characters, displays on a computer screen with ease. The Unicode Standard has been available for more than a decade, but it has taken time for its use to be incorporated into the computer world.

Fortunately, many of the major players that developed the standard were leaders in the computer industry. It has been incorporated into programming languages such as Java from Sun Microsystems, and into operating systems such as Windows 2000 produced by Microsoft as well as Mac OS X produced by Apple. An important step for libraries was support for the use of Unicode in the MARC 21 Specifications for Record structure, Character Sets and Exchange Media[1]. Companies that produce library system software have also started to incorporate Unicode into their software (Tull, 2002). The adoption of this standard and its implementation in computer systems and applications are crucial to the display, searching and especially the exchange of textual information in multiple languages.

Support for materials in multiple languages prior to Unicode

Ohio State University (OSU) Libraries houses collections written in many scripts including Arabic, Hebrew, Chinese, Japanese, Korean and Cyrillic. Providing patrons with access to these materials has taken various forms over the years.

Access to Arabic and Hebrew materials led to a series of experiments in the library during the 1980s and 1990s. Libraries had been relying on romanization, the process of writing or transliterating a non-Latin character into a Latin character, to provide access to these kind of materials. Originally at OSU, public catalog cards were produced containing romanized or English access points, with the descriptive portion of the record typed in the Arabic or Hebrew script using a special typewriter. An

extra card was produced for special catalogs housed in the Jewish Studies (JDC) and Middle East Studies (MES) Reading Rooms, where it was filed first by language, then by title according to the vernacular alphabet. In 1984, OSU Libraries decided to romanize the complete record, thus providing patrons with the same subject and added entry access to these materials in the online catalog, as to those in English, French or German. One extra card was produced and sent to the appropriate reading room, The title was typed in Hebrew or Arabic script on the card, which was then filed in the vernacular catalog.

In 1987, the two reading rooms jointly acquired a Macintosh SE. Using vernacular word processing programs together with the database program, FileMaker Plus, electronic files were created and used to produce temporary cards containing the title as part of the pre-cataloging process. Adding these temporary cards to the vernacular catalogs provided patrons with more immediate access to uncataloged materials. OCLC cards replaced these temporary cards when the titles were cataloged. The next step was to replace printed cards with a printout of the electronic files and use the printouts as supplements to the vernacular catalogs (Straley, 1989; Zipin, 1990).

In 1991, OSU Libraries purchased the ALEPH system from Ex Libris for use as a supplemental public catalog for the display and searching of titles and authors in Arabic and Hebrew scripts. Records were extracted from the main library system (LCS) by language code, converted by programmers and added to ALEPH. Staff in JDC and MES added vernacular titles and vernacular cross-references for romanized authority forms of personal names. After the initial load of the records from LCS, new materials were added to ALEPH by importing OCLC records. Patrons could now search by title, keyword or personal name in ALEPH to find materials in Arabic, Hebrew, Ottoman Turkish, Persian, Yiddish, and other languages written in Arabic or Hebrew scripts.

Prior to Unicode implementation, the library also relied on romanized records from OCLC in the public catalog for Chinese, Japanese and Korean (CJK(r)) material. In the late 1970s and early 1980s, titles were cataloged using

romanization only. OCLC introduced its CJK Cataloging software in 1986 and OSU Libraries began to use it soon after it went into production (Wang, 1985). However, due to space limitations on the hardware where the LCS library system was housed, the 880 fields in the OCLC records were stripped from the records as they entered the system. The public could only search using the romanized forms of authors, titles, etc. In order to provide some public access to CJK titles, the library installed OCLC's CJK software with access to OCLC's WorldCat database on one public computer in the East Asian Studies Reading Room.

Although this was not the most efficient method of finding an item in the library, it provided a way to search for materials using the native scripts of these languages and would allow the public to identify titles in WorldCat that belonged to OSU.

Several things have happened in the last decade to greatly enhance the library's ability to provide access to materials in non-Latin scripts. First, the library migrated to Innovative Interfaces's INNOPAC system in 1994, allowing catalogers to export full CJK records, including the 880 fields, from OCLC into the local system. Second, Innovative developed several language modules that incorporated Unicode. They also developed the Millennium system, which has Java at its core and thus, Unicode as its native character set. Third, software companies, such as Microsoft, released newer versions of their operating systems such as Windows 2000, which fully incorporate Unicode. Fourth, OCLC developed its Arabic Cataloging software, which requires Windows 2000, allowing cataloging staff to import records that include the Arabic script directly into the library system. OCLC has not developed cataloging software for Hebrew or Cyrillic yet but with Innovative's Millennium software, staff can enter these scripts directly into the bibliographic records.

At this point, Innovative's system was capable of replacing the ALEPH system for access to Arabic and Hebrew materials. Vernacular fields can be entered directly into Millennium for those scripts, such as Hebrew and Cyrillic, which are not yet supported by OCLC. As part of the Unicode implementation project, the authors created a plan to incorporate all of the

vernacular fields from the ALEPH records into the Innovative records and abandon the costly business of maintaining two library systems.

At the time the library purchased the language modules from Innovative, the library's Information Technology Division had to examine our PC environment and make several changes to fully support Unicode. The public PCs were still using Windows 95 and the library's browser of choice was Netscape Navigator 4.7x. The staffs PCs were migrating to Windows NT.

Unicode support at the PC level

In order to take advantage of Unicode, a number of things need to be in place on a PC. The operating system needs to support Unicode. Fonts that can display the characters need to be available or installed. A Web browser that supports Unicode is necessary and needs to be properly configured to use a font that can display the appropriate characters. An input method or program that provides a way for users to enter non-Latin characters on a standard keyboard, is also required if users need to be able to enter searches in the library catalog or to add 880 fields to a MARC record using native scripts. In addition, any application from a simple word processor to a complex library system must also support Unicode. Taking advantage of the Unicode character set has become easier in recent years with the incorporation of support for Unicode into many operating systems and Web browsers as well as the availability of Unicode fonts (i.e. fonts that include large numbers of characters from the Unicode character set). An excellent Web site that can help with all sorts of PC level issues is Alan Wood's Unicode Resources[2]. The Unicode Consortium also has a Web page entitled "Unicode enabled products" that lists products, from operating systems to databases, that provide partial or full support for Unicode[3].

Operating systems and input methods

Operating systems have been adding more support for Unicode with each new version. A variety of operating systems now fully support Unicode. Windows NT, Windows 2000 and

Windows XP use Unicode at the system level. Previous versions of Windows provided limited support for Unicode. The capability to display non-Latin characters is different from the capability to input them. Windows 2000 was a great boon to those wanting to use Unicode because it included Global Input Method Editors (IMEs) along with the language packages, which allow users to enter non-Latin characters on a standard keyboard. It still involves an installation process, first installing the language group and then the IME, but this is a great improvement over finding and installing third party software to accomplish this. Fortunately, the library was on a migration path to Windows 2000 and at the time that the authors started testing Unicode, there were already a few staff PCs running Windows 2000.

Although the Macintosh operating system has included Unicode support since version 8.5, it was not until Mac OS X that more applications, written for this operating system, started taking advantage of Unicode. Input methods for various languages have been added with newer versions of the operating system. Although, Mac OS 8.5 provided the capability of displaying Chinese, Japanese, or Korean, a language kit was still required to be able to input characters. Mac OS 9 added support for display of more scripts, including Cyrillic, and included an input method for Japanese, Korean, and simplified and traditional Chinese characters. Apple is calling Mac OS X (10.2) a multilingual operating system. Not only does it include input methods for the scripts of more languages, such as Arabic, Persian and Hebrew, but it also provides the option of displaying application menus and dialogs in other languages.

Some libraries have moved away from commercially available operating systems for various reasons, including cost, and have chosen Linux, the open source Unix-based operating system for their PCs. Support for Unicode in Linux is harder to gauge because of its numerous implementations. The Linux Internationalization Initiative, now known as the Open Internationalization Initiative, was formed in 1999 to look at some of these issues. They defined a globalization specification, OpenI18N 1.2 (formerly known as LI18NUX 2000 Globalization Specification 1.0 with

amendment 4) released in March 2002 that includes support for Unicode[4]. The Free Standards Group has also implemented a certification program for implementations that comply with this standard[5]. Release notes for version 8.0 of the popular Red Hat Linux mention that it installs UTF-8 (Unicode) locales by default for languages other than Chinese, Japanese and Korean[6]. Apparently, this has caused problems for some third party applications that do not support Unicode locales.

Web browsers

Popular Web browsers such as Microsoft's Internet Explorer, Netscape's Navigator and the latest version of Opera also support Unicode. With Web browsers that can be set to view Unicode, switching back and forth between different character sets to display different scripts is no longer necessary. Unicode support in browsers has been added over time with more support in each release. Some Unicode support was available in Netscape Navigator 4.03 but it was not until Netscape 6.X that it became a Unicode-based Web browser. In a similar fashion, Internet Explorer 4.01 included some support for Unicode but Internet Explorer 5.5 provided the library with the display and input capabilities it needed for searching the catalog. Opera version 6.0 was the first version to include support for Unicode.

When the authors began testing Unicode in the library system, the standard browser on public PCs was Netscape Navigator 4.72. Staff PCs had Navigator 4.79. Microsoft's Global IMEs, available with Windows 2000, were not completely supported by Netscape Communicator until version 6.0. Because these versions of Navigator could not accommodate the input of non-Latin characters, essential for searching the catalog, the library's browser of choice for the Unicode catalog became Internet Explorer 5.5. Since our initial testing, Internet Explorer 6.0 and Netscape Navigator 7.0 have been released. However, the library has only recently begun to install Internet Explorer 6.0 and the authors have not tested their capabilities and support for Unicode.

Fonts

Prior to the availability of fonts that support a wide range of the Unicode characters, it was sometimes necessary to install several fonts to support different scripts and then switch back and forth between character encodings in the Web browser to display them. This could be a problem when more than one non-Latin script was in a catalog record or in a list of search results. Fortunately, there are now several fonts that contain an amazing number of the characters included in Unicode. One of the earliest fonts, Bitstream's CyberBit, was originally developed to assist members of the Unicode Consortium with testing. It contains over 29,000 glyphs, and was originally available as a free download but now must be licensed[7]. The Arial Unicode MS font is produced by Microsoft and comes with Microsoft Office and some of their other software products. It contains over 50,000 glyphs including Arabic, Hebrew, Chinese, Japanese and Korean, the ones we were particularly interested in. James Kass produces a shareware font, Code2000, which contains over 34,000 glyphs in version 1.12[8]. Details of the characters and Unicode ranges supported by each font are available at the producer's Web site. The font still must be downloaded and/or installed on the PC. Once that is done, some settings in the Web browser must be adjusted to associate the font with the appropriate scripts.

Library systems

Support for Unicode must also be available at the application level, so everything from a simple word processor to a complex library system must also include support for the standard. To take advantage of Unicode in our library system, the library purchased four language modules (i.e. Arabic, Hebrew, Cyrillic and CJK) from Innovative in 1999. Support for Unicode in library systems can be either "native" or "enabled". Native systems use Unicode as the underlying character set, built into the software. "Enabled" means that the characters are stored in another character set in the records, such as the MARC 21 character sets, and mapped to Unicode values. Innovative's library system stores most non-Latin characters in its own proprietary character sets and maps them to Unicode

values. Unicode is the native character set of their new Java-based product, Millennium, so it interacts well with the Windows IMEs when it comes to entering a search in the Web OPAC as well as creating and editing bibliographic records in non-Latin scripts.

The complexity of a library system means that many areas come into play to support non-Latin scripts besides Unicode. Innovative had to set up indexing rules for the 880 fields in the bibliographic records. Using the indexing rules applied to fields in the bibliographic records as a guide, it was a fairly easy process for them to identify which 880 fields should go into the various indexes. However, since there were already many existing records containing 880 fields, the library gathered these records into a file using the presence of 880 fields and the language code in the fixed length fields as the common elements, so that Innovative could re-index these records to apply the additional rules. Another issue is the sort order of the scripts when a results screen displays records from multiple languages. There is no official standard to determine this sort order, so in our case, we set the scripts to sort like this: Latin, CJK, Arabic, Cyrillic, and Hebrew. There is also the issue of the sort order of characters within each script. This can be a difficult area because the way characters are expected to sort varies from culture to culture and is dependent on the expectations of the user. Other areas of a library system that come into play are the load and export tables. Support for non-Latin characters has to be enabled for each of the processes a record goes through: load, index, display, sort, and export.

To test the language modules, the authors used a PC with Windows 2000, Internet Explorer 5.5, and the Arial Unicode MS font. We also installed an input method editor for Arabic, Chinese, Japanese and Korean. The collection managers, with expertise in a particular language, searched the Web OPAC checking for problems in three areas: indexing, sort order and display of the characters. They examined diacritics in the romanized fields in the bibliographic records and non-Latin characters in the 880 fields. There were few problems with the diacritics used in Western European languages, but there were some display issues for some diacritics used in the

romanized forms of Eastern European languages. Innovative was still developing some of the mapping tables for particular languages so we were able to pinpoint problem characters for them. See the troubleshooting section below for specific examples of the types of problems we encountered in the library system.

While the authors were testing Unicode in the catalog, the Cataloging Department was planning to convert Chinese records from Wade-Giles to Pinyin romanization using OCLC's conversion services. Collection managers reported odd displays for some characters in the CJK records, which we eventually attributed to encoding errors. Cataloging decided to piggyback on the Pinyin conversion project to correct encoding errors in older CJK records and to obtain 880 fields for our pre-1994 CJK records. Replacing existing records with OCLC's WorldCat records seemed to be the best option. This had the added advantage of giving us any updates and enhancements to those records. The library system already had a load table that would provide protection for particular fields in our records, such as call number or location, so that we could overlay our existing records without losing vital local information. OCLC also transferred some fields from our existing records, such as the 590 fields, into the WorldCat records so that we could retain these as well. The end result of this project was much improved access to all of our CJK materials.

Going public

During the testing period, the public was unaware that they could display the scripts of these languages using Unicode because the Unicode catalog was on a separate port on the server and there was no link to it on the library's Web site. One issue that the library had to consider before providing access to the public was whether these language modules could be incorporated into the regular Web OPAC. We concluded that it needed to remain on a separate port. Our reasoning was based on the fact that most of the public PCs were not configured to take advantage of Unicode, and that most people using their home or office PCs to search the catalog would not have their PCs

properly configured to use Unicode. If a PC has everything it needs to take advantage of Unicode, as described in previous sections, then the language modules that relied on Unicode could be part of the regular Web OPAC. Without these in place, users could be confused by displays that have incorrect characters simply because they do not have their character encoding set to Unicode in the browser.

The encoding does not have to be set manually if the library system software can transmit the character set information to the browser and the browser can detect it. Innovative had not developed that capability at the time we implemented Unicode, but it will be able to do this in release 2002, phase 3. Some older browsers, such as Netscape 4.79, will not automatically switch the encoding even if the character set information is available on the Web page. Users might also be confused if they do not have a font that supports all of the characters. The system cannot display that character so it displays a small box in its place instead. Although it would be possible to reconfigure all of the public PCs in the library to use Unicode, the library had no control over home or office PCs. By setting the default character set for the browser to Unicode (UTF-8) on the public PCs, users could also experience displays of incorrect characters when leaving the library catalog and visiting other Web sites that use a different character set. These factors led us to keep the module on a separate port and create a separate link to the Unicode catalog from the Web catalog.

The authors also considered how much information to convey to the public about what needed to be in place on their home or office PCs to use the Unicode catalog. Since there are multiple versions of Web browsers as well as operating systems and fonts, it would be quite a daunting task to create a set of help pages for all situations users might have. Fortunately, Alan Woods has developed a detailed and thorough Web site, "Alan Woods Unicode Resources," with all of this information readily available[9]. We were able to create a brief help screen that explained the basics with links to Mr Woods' Web site[10]. The Unicode catalog was finally made public in Fall 2002 for Arabic, Chinese, Japanese and Korean. We continue to work on Persian, Ottoman Turkish and Hebrew.

Troubleshooting problems

The authors encountered numerous problems while implementing Chinese, Japanese, Korean and Arabic. Some problems resided within the various software products we were using. Others were due to a lack of understanding on our part. Each software product, from the operating system to the library system, has the potential for being the source of a problem. They all must support Unicode and since Unicode is still being incorporated into many of these systems and software, it is sometimes difficult to pinpoint where the problem lies.

To troubleshoot problems, it may be necessary to list each piece of software and eliminate them one by one to deduce which one is the culprit. It is best to follow a standard rule of thumb and only change one variable at a time in order find the source of the problem. For example, if the issue seems to be with the Web browser, change one setting at a time. If the issue seems to be either the browser or the library system, change one setting in the browser at a time and hold the library system configuration constant. When you have eliminated the browser as the problem, hold it constant and change the library system. A list of possible suspects is given below along with some troubleshooting hints and experiences that the authors had while implementing Unicode. Researching the producer's Web site to see if there is a list of known bugs related to Unicode or to verify the amount of Unicode support available within that software is an important first step to troubleshooting any problems.

Operating system

Older operating systems either do not support Unicode or do not fully support Unicode. The authors used a PC with Windows 2000, which completely supports Unicode, so we did not run into any difficulties in this area. Operating systems that support Unicode are listed at the Unicode Consortium's "Unicode enabled products" Web page[11].

Web browser

If a character displays incorrectly, open a different browser and check the same display. If the character displays correctly in the new

browser or another version of the same browser, then the problem lies within the browser. Make sure the font is associated with the appropriate script in the browser configuration. The MES collection manager noticed a display problem with the Arabic letter "Heh" with "Yeh" above (Unicode U+06C0). All of the other Arabic characters displayed correctly in Internet Explorer 5.5 except this one. The character displayed correctly in Netscape Navigator so we knew we had a browser issue. The Arabic script was properly associated with the Arial Unicode MS font in the browser, but to make this particular character display properly, we also had to associate the Latin-based language script with the same font. Adjusting this setting in the browser's configuration solved the problem.

Other problems may not be solvable except by upgrading to a new version of the browser. The library's browser of choice before Unicode was Netscape's Navigator. However, although version 4.x could display non-Latin scripts, it could not accommodate input of non-Latin characters. This was not a problem in Internet Explorer 5.5, so we again concluded that this was a browser issue. A little more research revealed that the Global IMEs available in Windows would not be fully supported in Netscape until version 6. That was a key reason the library started using Internet Explorer 5.X as the recommended browser for the Unicode catalog.

Font

Some display issues can be caused by the lack of a glyph that represents the character in the font. The character may appear as a box on the screen. In order to verify this, check the font through a character map. In Windows 2000 the character map is located under Start/Programs/Accessories/System Tools/character map. Select the font to see the characters available. Select "Advanced view" to search for the character by its Unicode value or name. If your character does not exist in that font, find another font that contains all the appropriate characters.

Character encoding

Sometimes display of incorrect characters simply means that the character is not properly encoded in the record and needs to be

corrected. Cataloging staff had been loading CJK records into the library system from OCLC for several years, but with no opportunity to view the native scripts contained in the 880 fields in the Web OPAC. During the testing period, collection managers noticed a lot of incorrect characters displaying in many of the older CJK records. At first it was not apparent whether the problem was within the library system or just that the characters were not encoded correctly in the records. Problems due to the library system will probably be consistent, i.e. the character will not display properly in any record. If the problem occurs in many records, but not all, it could be an indication of an input issue or load issue depending on how the records were entered into the system in the first place. We load records into our system in a variety of ways, using a number of different load tables, so it is possible that a problem could occur at the load level. To puzzle through this, we checked the original record in OCLC to see if it was properly encoded and then exported it into our system as a duplicate record to see if it loaded the character properly. If it did, we knew that we had an encoding problem and just had to correct the record.

Input method

The first time the MES collection manager used the Farsi (i.e. Persian) keyboard layout in Windows 2000, she discovered that every time she typed a particular character, the catalog would not return a list of Persian records, although there were several Persian records in the system that contained that particular character. MES staff had entered several Persian records into the system using the Arabic keyboard. If they used the Arabic keyboard to search the catalog, the proper results displayed. At first, we suspected a problem with the Farsi keyboard. Perhaps the character was not mapping properly to its Unicode value. To eliminate this particular input method as the source of the problem, we entered the character directly from the character map, and tried the search again, but still no luck. Since we had used two different input methods and had the same result, it appeared that the problem was not with the Farsi keyboard after all. An alternative to using the character map would be

to use a different keyboard layout for the same script from a different software company or a shareware source to see if the problem still arose.

After doing a little research, we realized that we lacked some crucial information about this particular character. The same letter exists in both Arabic and Persian, but we did not realize that the two letters mapped to different Unicode code points. The keyboard itself may not display these values, but the producer's Web site should have a chart of the characters with the keyboard layout and their equivalent Unicode values. Microsoft's Web site has pages devoted to internationalization with a query form to "Ask Dr International," to ask questions related to Unicode and other issues[12]. Dr International immediately cleared up our misconception and clarified that the letter on the Farsi keyboard was the Arabic letter KEHEH, which maps to Unicode value, (U+06A9). The same letter on the Arabic keyboard is the Arabic KAF, which maps to a separate Unicode value (U+0643). Since the Persian letter is referred to as a KAF and the name of the character listed in the Unicode Standard is the Sindhi name of the character, KEHEH, this had also added to the confusion. Although both languages use the same letter, it has different Unicode code points because the isolated and final forms of the letter differ in appearance in the different languages. To correct the problem, we simply replaced the Arabic letter KAF with the Arabic letter KEHEH.

Cataloguing software

OSU Libraries uses OCLC's Arabic Cataloging software to catalog Arabic and Persian records and OCLC's CJK Cataloging software to catalog Chinese, Japanese and Korean materials. This adds another layer of software to the troubleshooting list.

MES staff was using the Arabic Cataloging Module to catalog Persian records. It contains a nice Arabic transliterator, which automatically converts romanized characters to Arabic characters. Based on the ALA-LC romanization table for Arabic, each romanized consonant and long vowel is mapped to a single vernacular

character. Automatically converting a romanized field to vernacular script takes less time and leads to fewer input errors. However, other languages such as Persian use extended Arabic characters as well as the basic Arabic characters. Currently, staff uses the Arabic transliterator for Persian but they must alter the extended Arabic characters manually. A recommendation for OCLC would be to develop transliterators for other languages using the language code in the record to interpret which transliterator to select.

Library system

Troubleshooting the library system can be tricky because the record goes through a variety of processes after it enters the system, from load tables, to indexing, sorting, and export tables. Companies implement Unicode on a language-by-language basis so some languages may have complete support and others may be under development. Sometimes a quick call to the library system company's help desk can clear up any known bugs, but when problems arise it is helpful to map the life of a record after it enters your library system.

We ran into some display, sorting and load issues. Some characters would not properly display on the screen. After eliminating the font and Web browser as the source of the problem, we referred these problems to Innovative. Innovative had added support for the basic Arabic characters, but support for Persian and other languages, which use the extended Arabic characters, was still under development. The MES collection manager noticed that after she exported Persian records from OCLC into the library system, some of these extended characters would not display.

After consulting with Innovative, we discovered they were not in the mapping tables of the library system. We asked the company to incorporate the characters most essential for our purposes into the mapping tables in order to be able to catalog these materials. Once that was set up, these characters displayed, sorted and indexed properly. However, after some time, MES staff noticed that the characters did not display properly in some records. After consulting with the company again and reviewing our internal procedures, the authors mapped out what was happening to these

records and discovered the cause of the problem. The records imported into the system from OCLC were filtered through one load table. They indexed and sorted properly. However, once a month cataloging staff gather all newly cataloged records into a file, and send them to OCLC's MARS service for authorities processing. Once the records were returned from OCLC, they were reloaded into the system using a different load table. The extended Arabic characters were not incorporated into the second load table. Once Innovative had this in place, we were ready to catalog Persian materials.

When the MES collection manager started searching the Web OPAC in Arabic, she noticed that certain characters were not sorting on the browse screens as she expected. There were five characters in the Arabic script that we wanted to sort as the same character:

- (1) Aleph (U+627);
- (2) Aleph with Madda above (U+0622);
- (3) Aleph with Hamza above (U+623);
- (4) Aleph with Hamza below (U+0625); and
- (5) Aleph Wasla (U+0671).

These characters were sorting in Unicode value order, as separate characters. We wanted them to sort and index as a single value as they are essentially the same character to users and the differences between them are generally transparent in print. However, people that contribute Arabic records to OCLC might or might not differentiate between them, hence our decision to have them all sort and index as a single character. Innovative adjusted the sorting tables to make these characters display in the expected order and reindexed the existing Arabic records to apply the change.

Another sorting issue arose during the testing of Chinese. When the collection manager searched the catalog, he did not understand the sort order of the records on the results screen. He was used to seeing Chinese sorted by the number of strokes in each character, characters with fewer strokes sorting before characters with more strokes. Innovative clarified that the default sort was by the Unicode value and that several sort options, one being sort by stroke, were under development. When these options became available, we opted to use sort by stroke. We had to consider the fact that the

Chinese characters used in Japanese and Korean would also sort by stroke, but that sorting for Japanese and Korean characters would still be by Unicode value, which although not ideal, was the only option available to us at the time.

Conclusion

The Unicode Standard is an incredible step forward for the global exchange of textual information. It is a foundation for ease of display and searching of multiple scripts. During this transitional period, software companies and library system vendors are incorporating support for Unicode into their systems language by language. Until such time as Unicode is fully incorporated into all of these software products, there may be problems for users, some of which are described in the preceding sections.

Library systems store cataloging records in their own proprietary character sets or in the character sets designated in the MARC 21 Specifications for Record structure, Character Sets and Exchange Media. It was not until 1998 that the US MARC Advisory Committee approved a proposal to allow bibliographic records to be encoded using Unicode, although restricting its use to 16,000 plus characters that correspond to those available in the MARC 21 repertoire of character sets[13]. This restriction would facilitate interchange of records during the transitional period when systems are incorporating Unicode. Storing the data in Unicode would do much to alleviate problems. However, the Unicode character set contains many more characters than the MARC 21 character sets and is still growing. Knowing that there are so many more characters available, makes waiting out this transitional period difficult. It is also difficult to gauge when this transitional period will be over. Continuing to expand the number of characters in the MARC 21 character sets would assist libraries to take further advantage of Unicode.

Notes

- 1 www.loc.gov/marc/specifications/spechome.html
- 2 www.alanwood.net/unicode/index.html
- 3 www.unicode.org/onlinedat/products.html
- 4 www.li18nux.org/docs/html/LI18NUX-2000-amd4.htm
- 5 www.opengroup.org/lsc/cert/
- 6 www.redhat.com/docs/manuals/linux/RHL-8.0-Manual/release-notes/x86/
- 7 www.bitstream.com/categories/support/other_support/cyberbit_main.html
- 8 <http://home.att.net/~jameskass/>
- 9 www.alanwood.net/unicode/index.html
- 10 http://library.ohio-state.edu/screens/libinfo_26.html
- 11 www.unicode.org/onlinedat/products.html
- 12 www.microsoft.com/globaldev/drintl/columns/default.aspx
- 13 www.loc.gov/marc/specifications/speccharucs.html

References

- Straley, D. (1989), "Increasing productivity and public satisfaction with Middle East and Judaic library services through use of a personal computer", *MELA Notes*, No. 47, pp. 33-5.
- Tull, L. (2002), "Library systems and Unicode: a review of the current state of development", *Information Technology and Libraries*, Vol. 21 No. 4, pp. 181-5.
- Unicode Consortium (1991), *The Unicode Standard, Version 1.0*, Addison-Wesley Developers Press, Reading, MA (note: this was published in two volumes in 1991 and 1992).
- Unicode Consortium (2003), *The Unicode Standard, Version 4.0*, Addison-Wesley, Reading, MA (note: a prepublication version is available at: www.unicode.org/versions/Unicode4.0.0/).
- Wang, A. (1985), "OCLC CJK automated library information network", *Journal of Library & Information Science*, Taipei, Taiwan, Vol. 11 No. 2, pp. 143-53.
- Zipin, A. (1990), "Vernacular access to titles in Hebrew script at the OSU Libraries", *MELA Notes*, No. 50/51, pp. 36-42.

Further reading

- Graham, T. (2000), *Unicode: A Primer*, M&T Books, Foster City, CA.