Character encoding

From Wikipedia, the free encyclopedia

A **character encoding** system consists of a code that pairs each character from a given repertoire with something else—such as a bit pattern, sequence of natural numbers, octets, or electrical pulses—in order to facilitate the transmission of data (generally numbers or text) through telecommunication networks or for data storage. Other terms such as **character set**, **character map**, and **code page** are used almost interchangeably, but these terms have related but distinct meanings described below.

Contents

History

Common examples of character encoding systems include Morse code, the Baudot code, the American Standard Code for Information Interchange (ASCII) and Unicode.

Morse code was introduced in the 1840s and is used to encode each letter of the Latin alphabet, each Arabic numeral, and some other characters via a series of long and short presses of a telegraph key. Representations of characters encoded using Morse code varied in length.

The Baudot code was created by Émile Baudot in 1870, patented in 1874, modified by Donald Murray in 1901, and standardized by CCITT as International Telegraph Alphabet No. 2 (ITA2) in 1930.

ASCII was introduced in 1963 and is a 7-bit encoding scheme used to encode letters, numerals, symbols, and device control codes as fixed-length codes using integers.

IBM's Extended Binary Coded Decimal Interchange Code (usually abbreviated EBCDIC) is an 8-bit encoding scheme developed in 1963.

The limitations of such sets soon became apparent, and a number of ad hoc methods were developed to extend them. The need to support more writing systems for different languages, including the CJK family of East Asian scripts, required support for a far larger number of characters and demanded a systematic approach to character encoding rather than the previous ad hoc approaches.

Early binary repertoires include:

- Bacon's cipher
- Braille
- International maritime signal flags
- Chinese telegraph code (Hans Schjellerup, 1869, modified 1872 and following)
  Encoding of Chinese characters as 4-digit decimals.

Code unit

The code unit, is a unit used for character encoding.

- With US-ASCII, code unit is 7 bits.
- With UTF-8, code unit is 8 bits.
- With EBCDIC, code unit is 8 bits.
- With UTF-16, code unit is 16 bits.
- With UTF-32, code unit is 32 bits.

Then the encoding attributes to single code unit or sequence of code unit a meaning.

Unicode encoding model

Unicode and its parallel standard, the ISO/IEC 10646 Universal Character Set, together constitute a modern, unified character encoding. Rather than mapping characters directly to octets (bytes), they separately define what characters are available, their numbering, how those numbers are encoded as a series of "**code units**" (limited-size numbers), and finally how those units are encoded as a stream of octets. The idea behind this decomposition is to establish a universal set of characters that can be encoded in a variety of ways.[1] To correctly describe this model one needs more precise terms than "character set" and "character encoding". The terms used in the modern model follow:[1]

A **character repertoire** is the full set of abstract characters that a system supports. The repertoire may be closed, i.e. no additions are allowed without creating a new standard (as is the case with ASCII and most of the ISO-8859 series), or it may be open, allowing additions (as is the case with Unicode and to a limited extent the Windows code pages). The characters in a given repertoire reflect decisions that have been made about how to divide writing systems into basic information units. The basic variants of the Latin, Greek, and Cyrillic alphabets, can be broken down into letters, digits, punctuation, and a few **special characters** like the space,[*citation needed*] which can all be arranged in simple linear sequences that are displayed in the same order they are read. Even with these alphabets, however, diacritics pose a complication: they can be regarded either as part of a single character containing a letter and diacritic (known as a precomposed character), or as separate characters. The former allows a far simpler text handling system but the latter allows any letter/diacritic combination to be used in text. Ligatures pose similar problems. Other writing systems, such as Arabic and Hebrew, are represented with more complex character repertoires due to the need to accommodate things like bidirectional text and glyphs that are joined together in different ways for different situations.

A **coded character set** (CCS) specifies how to represent a repertoire of characters using a number of (typically non-negative) integer values called *code points*. For example, in a given repertoire, a character representing the capital letter "A" in the Latin alphabet might be assigned to the integer 65, the character for "B" to 66, and so on. A complete set of characters and corresponding integers is a *coded character set*. Multiple coded character sets may share the same repertoire; for example ISO/IEC 8859-1 and IBM code pages 037 and 500 all cover the same repertoire but map them to different codes. In a coded character set, each code point only represents one character, i.e., a coded character set is a function.

A **character encoding form** (CEF) specifies the conversion of a coded character set's integer codes into a set of limited-size integer *code values* that facilitate storage in a system that represents numbers in binary form using a fixed number of bits (i.e. practically any computer system). For example, a system that stores numeric information in 16-bit units would only be able to directly represent integers from 0 to 65,535 in each unit, but larger integers could be represented if more than one 16-bit unit could be used. This is what a CEF accommodates: it defines a way of mapping a *single* code *point* from a range of, say, 0 to 1.4 million, to a series of *one or more* code *values* from a range of, say, 0 to 65,535.

The simplest CEF system is simply to choose large enough units that the values from the coded character set can be encoded directly (one code point to one code value). This works well for coded character sets that fit in 8 bits (as most legacy non-CJK encodings do) and reasonably well for coded character sets that fit in 16 bits (such as early versions of Unicode). However, as the size of the coded character set increases (e.g. modern Unicode requires at least 21 bits/character), this becomes less and less efficient, and it is difficult to adapt existing systems to use larger code values. Therefore, most systems working with later versions of Unicode use either UTF-8, which maps Unicode code points to variable-length sequences of octets, or UTF-16, which maps Unicode code points to variable-length sequences of 16-bit words.

Next, a **character encoding scheme** (CES) specifies how the fixed-size integer code values should be mapped into an octet sequence suitable for saving on an octet-based file system or transmitting over an octet-based network. With Unicode, a simple character encoding scheme is used in most cases, simply specifying whether the bytes for each integer should be in big-endian or little-endian order (even this isn't needed with UTF-8). However, there are also compound character encoding schemes, which use escape sequences to switch between several simple schemes (such as ISO/IEC 2022), and compressing schemes, which try to minimise the number of bytes used per code unit (such as SCSU, BOCU, and Punycode). See comparison of Unicode encodings for a detailed discussion.

Finally, there may be a **higher level protocol** which supplies additional information that can be used to select the particular variant of a Unicode character, particularly where there are regional variants that have been 'unified' in Unicode as the same character. An example is the XML attribute xml:lang.

The Unicode model reserves the term **character map** for historical systems which directly assign a sequence of characters to a sequence of bytes, covering all of CCS, CEF and CES layers.[1]

Character sets, code pages, and character maps

In computer science, the terms **character encoding**, **character map**, **character set** or **code page** were historically synonymous[*citation needed*], as the same standard would specify a repertoire of characters and how they were to be encoded into a stream of code units – usually with a single character per code unit. The terms now have related but distinct meanings, reflecting the efforts of standards bodies to use precise terminology when writing about and unifying many different encoding systems.[1] Regardless, the terms are still used interchangeably, with *character set* being nearly ubiquitous.

A **code page** usually means a byte oriented encoding, but with regard to some suite of encodings (covering different scripts), where many characters share the same codes in most or all those code pages. Well known code page suites are "Windows" (based on Windows-1252) and "IBM"/"DOS" (based on code page 437), see Windows code page for details. Most, but not all, encodings referred to as code pages are single-byte encodings (but see octet on byte size.)

IBM's Character Data Representation Architecture (CDRA) designates with coded character set identifiers (CCSIDs) and each of which is variously called a **charset**, **character set**, **code page**, or **CHARMAP**.[1]

The term **code page** does not occur in Unix or Linux where **charmap** is preferred, usually in the larger context of locales.

Contrasted to CCS above, a **character encoding** is a map from abstract characters to code words. A **character set** in HTTP (and MIME) parlance is the same as a character encoding (but not the same as CCS).

Legacy encoding is a term sometimes used to characterize old character encodings, but with an ambiguity of sense. Most of its use is in the context of Unicodification, where it refers to encodings that fail to cover all Unicode code points, or, more generally, using a somewhat different character repertoire: several code points representing one Unicode character,[2] or versa (see e.g. code page 437). Some sources refer to an encoding as *legacy* only because it preceded Unicode.[3] All Windows code pages are usually referred to as legacy, both because they antedate Unicode and because they are unable to represent all 2^{21} possible Unicode code points.

Character encoding translation

As a result of having many character encoding methods in use (and the need for backward compatibility with archived data), many computer programs have been developed to translate data between encoding schemes. Some of these are cited below.

Cross-platform:

- Web browsers – most modern web browsers feature automatic character encoding detection. On Firefox 3, for example, see the View/Character Encoding submenu.
- iconv – program and standardized API to convert encodings
- convert_encoding.py – Python based utility to convert text files between arbitrary encodings and line endings.[4]
- decodeh.py – algorithm and module to heuristically guess the encoding of a string.[5]
- International Components for Unicode – A set of C and Java libraries to perform charset conversion. uconv can be used from ICU4C.
- chardet (http://pypi.python.org/pypi/chardet) – This is a translation of the Mozilla automatic-encoding-detection code into the Python computer language.
- The newer versions of the unix File command attempt to do a basic detection of character encoding. (also available on cygwin and mac)

Linux:

- cmv - simple tool for transcoding filenames.[6]
- convmv – convert a filename from one encoding to another.[7]
- cstocs – convert file contents from one encoding to another
- enca – analyzes encodings for given text files.[8]
- recode – convert file contents from one encoding to another[9]
- utrac – convert file contents from one encoding to another.[10]

Windows:

- Encoding.Convert – .NET API[11]
- MultiByteToWideChar/WideCharToMultiByte – Convert from ANSI to Unicode & Unicode to ANSI[12]
- cscvt – character set conversion tool[13]
- enca – analyzes encodings for given text files.[14]

See also

- Alt code
- Character encodings in HTML
- Character encoding – articles related to character encoding in general
- Character sets – articles detailing specific character encodings
- Code page – various character set encodings used by IBM, Microsoft, SAP...
- Hexadecimal representations
- *Mojibake* – character set mismap.
- Mojikyo – a system ('glyph set') that includes over 100,000 Chinese character drawings, modern and ancient, popular and obscure.[15][16]
- TRON, part of the TRON Project, is an encoding system that does not use Han Unification, instead it uses 'control codes' to switch between 16bit 'planes' of characters.[16][17]
- Universal Character Set characters
- Windows code page – various character set encodings used by Microsoft Windows
- Charset sniffing – used in some applications when character encoding metadata is not available

Common character encodings

- ISO 646
  - ASCII
- EBCDIC
  - CP37
  - CP930
  - CP1047
- ISO 8859:
  - ISO 8859-1 Western Europe

  - ISO 8859-2 Western and Central Europe
  - ISO 8859-3 Western Europe and South European (Turkish, Maltese plus Esperanto)
  - ISO 8859-4 Western Europe and Baltic countries (Lithuania, Estonia, Latvia and Lapp)
  - ISO 8859-5 Cyrillic alphabet
  - ISO 8859-6 Arabic
  - ISO 8859-7 Greek
  - ISO 8859-8 Hebrew
  - ISO 8859-9 Western Europe with amended Turkish character set
  - ISO 8859-10 Western Europe with rationalised character set for Nordic languages, including complete Icelandic set
  - ISO 8859-11 Thai
  - ISO 8859-13 Baltic languages plus Polish
  - ISO 8859-14 Celtic languages (Irish Gaelic, Scottish, Welsh)
  - ISO 8859-15 Added the Euro sign and other rationalisations to ISO 8859-1
  - ISO 8859-16 Central, Eastern and Southern European languages (Albanian, Croatian, Hungarian, Polish, Romanian, Serbian and Slovenian, but also French, German, Italian and Irish Gaelic)
- CP437, CP737, CP850, CP852, CP855, CP857, CP858, CP860, CP861, CP862, CP863, CP865, CP866, CP869
- MS-Windows character sets:
  - Windows-1250 for Central European languages that use Latin script, (Polish, Czech, Slovak, Hungarian, Slovene, Serbian, Croatian, Romanian and Albanian)
  - Windows-1251 for Cyrillic alphabets
  - Windows-1252 for Western languages
  - Windows-1253 for Greek
  - Windows-1254 for Turkish
  - Windows-1255 for Hebrew
  - Windows-1256 for Arabic
  - Windows-1257 for Baltic languages
  - Windows-1258 for Vietnamese
- Mac OS Roman
- KOI8-R, KOI8-U, KOI7
- MIK
- ISCII
- TSCII
- VISCII
- JIS X 0208 is a widely deployed standard for Japanese character encoding that has several encoding forms.
  - Shift JIS (Microsoft Code page 932 is a dialect of Shift_JIS)
  - EUC-JP
  - ISO-2022-JP
- JIS X 0213 is an extended version of JIS X 0208.
  - Shift_JIS-2004
  - EUC-JIS-2004
  - ISO-2022-JP-2004
- Chinese Guobiao
  - GB 2312
  - GBK (Microsoft Code page 936)
  - GB 18030
- Taiwan Big5 (a more famous variant is Microsoft Code page 950)
- Hong Kong HKSCS
- Korean
  - KS X 1001 is a Korean double-byte character encoding standard
  - EUC-KR
  - ISO-2022-KR
- Unicode (and subsets thereof, such as the 16-bit 'Basic Multilingual Plane'). See UTF-8
- ANSEL or ISO/IEC 6937

References

- Mackenzie, Charles E. (1980). *Coded Character Sets, History and Development*. Addison-Wesley. ISBN 0-201-14460-3.

1. ^ *a b c d e* "Unicode Technical Report #17: Unicode Character Encoding Model" (http://www.unicode.org/reports/tr17/) . 2008-11-11. http://www.unicode.org/reports/tr17/. Retrieved 2009-08-08.
2. ^ "Processing database information using Unicode, a case study" (http://www.basistech.co.jp/knowledge-center/database/uc-trillium-2.ppt)
3. ^ Constable, Peter (2001-06-13). "Character set encoding basics" (http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&item_id=IWS-Chapter03#79e846db) . *Implementing Writing Systems: An introduction*. SIL International. http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&item_id=IWS-Chapter03#79e846db. Retrieved 2010-03-19.
4. ^ Homepage of Michael Goerz – convert_encoding.py (http://users.physik.fu-berlin.de/~mgoerz/blog/programs/convert_encoding/)
5. ^ Decodeh – heuristically decode a string or text file (http://gizmojo.org/code/decodeh/)
6. ^ CharsetMove - Simple Tool for Transcoding Filenames (https://cmv.fossrec.com)
7. ^ Convmv – converts filenames from one encoding to another (http://www.j3e.de/linux/convmv/man/)
8. ^ Extremely Naive Charset Analyser (http://directory.fsf.org/project/enca/)
9. ^ Recode – GNU Project – Free Software Foundation (FSF)

(http://www.gnu.org/software/recode/recode.html)
10. ^ Utrac Homepage (http://utrac.sourceforge.net/)
11. ^ Microsoft .NET Framework Class Library – Encoding.Convert Method (http://msdn.microsoft.com/en-us/library/system.text.encoding.convert(VS.71).aspx)
12. ^ MultiByteToWideChar/WideCharToMultiByte – Convert from ANSI to Unicode & Unicode to ANSI (http://support.microsoft.com/kb/138813)
13. ^ Character Set Converter (http://www.kalytta.com/tools.php)
14. ^ Extremely Naive Charset Analyser (http://www.john.geek.nz/2010/02/enca-binary-compiled-for-32-bit-windows/)
15. ^ "Mojikyo English page" (http://www.mojikyo.org/html/abroad/abroad_top.html) . mojikyo.org. http://www.mojikyo.org/html/abroad/abroad_top.html. Retrieved 2009 11 07.
16. ^ ***a*** ***b*** "Character Set List" (http://www.jbrowse.com/text/unij.html) . jbrowse.com. http://www.jbrowse.com/text/unij.html. Retrieved 2009 11 07.
17. ^ "TRON code website" (http://www2.tron.org/troncode.html) . tron.org. http://www2.tron.org/troncode.html. Retrieved 2009 11 07.

External links

- Character sets registered by Internet Assigned Numbers Authority (http://www.iana.org/assignments/character-sets)
- Unicode Technical Report #17: Character Encoding Model (http://www.unicode.org/unicode/reports/tr17/)

Retrieved from "http://en.wikipedia.org/w/index.php?title=Character_encoding&oldid=505348402"
Categories: Character encoding | Natural language and computing