Kleines Handbuch Metadaten

Zeichensätze

Alexander Jahnke

1 Einleitung

Metadaten bestehen aus zwei Typen von Daten: Zum einen Referenzen auf andere Daten (z.B. URIs), zum anderen aus Text, der in irgendeiner menschlichen Sprache vorliegt. Solange schriftliche Textdaten auf Papier oder einem anderen nicht-elektronischen Datenträger vorliegen, können sie verhältnismäßig einfach ausgetauscht und – entsprechend der sprachlichen Kompetenz des Rezipienten – verarbeitet werden. Anders verhält es sich aber bei der elektronischen Verarbeitung von Textdaten. Hier müssen diese Daten in einer Form vorliegen, die der Computer speichern und verarbeiten kann. Auf einer abstrakten Ebene stellt man sich einen Computer meist als eine Maschine vor, die intern mit Zahlen arbeitet, d.h. jegliche Art von Daten muss auf die eine oder andere Weise durch Zahlen repräsentierbar sein um von einem Rechner verarbeitet werden zu können. Dass das für Textdaten die Zuordnung einzelner Buchstaben zu Zahlenwerten bedeutet, ist wahrscheinlich selbstevident – verschiedene Lösungsmöglichkeiten gab und gibt es jedoch hinsichtlich der Frage, welche Buchstaben denn genau welchen Zahlen zugeordnet werden sollen.

Heute gibt es mit Unicode einen sehr weit verbreiteten Standard, der diese Zuordnung für fast alle von Menschen verwendeten Schriftsysteme regelt. Der Definition dieses Standards geht allerdings eine über 100jährige Vorgeschichte voraus (d.h. die zugrundeliegende Frage geht schon in die Zeit vor der Erfindung des Computers zurück) und auch heute ist Unicode noch weit davon entfernt der ausschließliche Standard für die Codierung von Textdaten zu sein.

Für den Metadatenbereich bedeutet das, dass überall dort, wo Daten ausgetauscht, übertragen, angezeigt, ausgewertet oder mit anderen Daten verglichen werden sollen, auf die jeweils verwendete Codierung geachtet werden muss. In Kontexten in denen schon länger mit elektronischer Datenverarbeitung gearbeitet wird (z.B. im Bibliotheksbereich), entstehen auch heute immer noch Daten in anderen Zeichensätzen als Unicode. Der vorliegende Beitrag versucht zum einen, die auch heute noch verwendeten Zeichensätze zu beschreiben und zum anderen daran exemplarisch die Methoden aufzuzeigen, nach denen Zeichen grundsätzlich codiert werden können. Die tatsächliche Anzahl der existierenden Zeichencodierungen ist natürlich zu groß, als dass sie hier auch nur annähernd behandelt werden könnte – die Prinzipien nach denen Zeichensätze konstruiert werden sind hingegen überschaubar und beschreibbar. Wir versuchen uns dabei auf Metadatenanwendungen als den Einsatzbereich für Zeichencodierungen zu beschränken und lassen Besonderheiten, die sich aus anderen Einsatzbereichen wie bspw. dem Mailverkehr ergeben, weitestgehend unberücksichtigt.

Am Ende dieses Beitrags sind in einem kurzen Abschnitt einige hilfreiche Werkzeuge zum Umgang mit verschiedenen Zeichensätzen aufgeführt – die Auswahl stammt aus der praktischen Arbeit und ist damit eher willkürlich und natürlich unvollständig. Es ist wahrscheinlich nicht nötig zu betonen, dass keines dieser Werkzeuge von KIM entwickelt wurde oder in irgendeiner Weise von KIM gepflegt oder verantwortet wird.

Zur Terminologie ist noch voranzustellen, dass wir die Begriffe Zeichencodierung und Zeichensatz im Folgenden synonym verwenden. Damit ist immer die Menge der Zuordnungen einzelner (Schrift-)Zeichen zu bestimmten Zahlenwerten zum Zweck der automatischen Verarbeitung dieser Daten gemeint. Davon zu trennen – und hier überhaupt nicht behandelt – sind Fragen der graphischen Darstellung von Zeichen, also alles was mit Schriftvarianten, Schriftarten bzw. Fonts zu tun hat.

2 Die Vorgeschichte

Die Geschichte der Darstellung von Zeichen in einer Weise, dass eine Maschine sie, wenn nicht verstehen, dann aber doch verarbeiten kann, beginnt mit der Erfindung der (elektromagnetischen) Telegraphie. Hier steht zum ersten Mal an beiden Enden des Kommunikationskanals ein Apparat, die Signale sendet oder in Empfang nimmt. Unter den vielen verschiedenen ersten Ansätzen, Schriftzeichen durch elektrische Signale darzustellen ist das von Samuel Morse 1837 erfundene und von Friedrich Clemens Gehrke 1848 weiterentwickelte Morsealphabet[1] sicher das bekannteste – und wahrscheinlich das einzige, das auch heute noch hier und da ver-

[1] Vgl. Morsecode. In: Wikipedia, die freie Enzyklopädie. Bearbeitungsstand: 13. August 2012, 10:33 UTC. URL: http://de.wikipedia.org/w/index.php?title=Morsecode&oldid=106754096

wendet wird. Der Morsecode basiert auf Signalen unterschiedlicher Länge mit Pausen dazwischen und wird sequentiell gesendet, d.h. ein Signal nach dem anderen.

Mit dem Ziel die Übertragungsgeschwindigkeit einer telegraphischen Nachricht zu erhöhen, entwickelte der französische Ingenieur Jean-Maurice-Émile Baudot in den 1870er Jahren ein Verfahren, mit dem mehrere Signale quasi gleichzeitig über eine Leitung übertragen werden konnten.[2] Hierfür konzipierte er auch ein Alphabet das ein Schriftzeichen durch jeweils 5 gleichzeitig übertragene Signale darstellt. Das von Baudot entwickelte Sendegerät hatte dafür 5 Tasten, von denen man eine, mehrere oder alle gleichzeitig drücken musste, um ein einzelnes Zeichen zu senden.[3] Damit hatte er zugleich auch das zentrale Prinzip erfunden auf dem die elektronische Verarbeitung von Informationen so wie wir sie heute kennen beruht, nämlich die Darstellung von Zeichen bzw. Information durch eine Kombination mehrerer paralleler Signale.

Baudots 5 Tasten – wir würden heute von 5 Bits sprechen – lassen sich auf insgesamt 32 verschiedene Weisen kombinieren. D.h. damit lassen sich max. 32 verschiedene Zeichen darstellen, was allerdings nicht ganz ausreichend ist für ein Alphabet mit 26 Buchstaben und 10 Ziffern. Um sowohl Buchstaben als auch Zahlen und noch einige andere Sonderzeichen darstellen zu können, waren in Baudots Code die 5-Bit Kombinationen mit jeweils 2 Zeichen belegt. Ein spezielles Zeichen signalisierte dann, dass zwischen diesen beiden Belegungen umgeschaltet werden sollte – auch dieses Prinzip wurde später bei der Definition von Zeichensätzen für Sprachen mit einem großen Zeichenvorrat z.B. Japanisch oder Koreanisch wieder aufgegriffen.

Etwa 30 Jahre später entwickelte der aus Neuseeland stammende Erfinder Donald Murray Baudots Telegraphen weiter zu einem Vorläufer des späteren Fernschreibers. Er führte dabei einige Bit-Kombinationen in den Zeichensatz ein, die das Empfangsgerät steuern konnten, also bspw. einen Zeilenvorschub oder ein Klingeln auslösen konnten. Diese "Steuerzeichen" sind bis heute in den allermeisten EDV-Zeichensätzen erhalten geblieben, auch wenn einzelne von ihnen für die elektronische Datenverarbeitung eigentlich bedeutungslos sind. Murrays Überarbeitung des Codes wurde vom Comité Consultatif International Téléphonique et Télégraphique, der heutigen International Telecommunication Union (ITU), als CCIT-2 normiert und war der Standard für die Zeichencodierung im Telex-Netz.

3 Prinzipien nach denen Zeichensätze aufgebaut sind

Die Informationseinheit, die bei Baudot noch 5 Stellen umfasste (Der deutsch-amerikanische Ingenieur Werner Buchholz prägte in den 1950er Jahren dafür den Begriff Byte),[4] umfasst in Computern inzwischen standardmäßig 8 Stellen. 8 Stellen (Bits) ergeben 256 mögliche Kombinationen, d.h. 256 mögliche verschiedene Zeichen. Das sind genug Zeichen für die meisten europäischen Sprachen inkl. der gebräuchlichsten Sonderzeichen, aber auch für Sprachen wie Hebräisch oder Arabisch. Für Sprachen, die mehr Zeichen benötigen, wie z.B. Amharisch oder gar Japanisch oder Chinesisch, aber auch Sprachen in denen viele Diakritika verwendet werden, wie bspw. Altgriechisch, reichen diese 256 möglichen Zeichen jedoch nicht. Um diese Schriften adäquat auf dem Computer darstellen zu können hat man verschiedene Mechanismen entwickelt, wie man die Begrenzung auf 256 Zeichen umgehen kann. Eine Möglichkeit ist die bereits von Baudot und Murray angewendete Umschalttechnik, eine andere für die Darstellung von einem Zeichen von vornherein 2 Bytes zu verwenden, eine dritte, einzelne Zeichen in ihre Bestandteile zu zerlegen also bspw. ein **ä** als eine Kombination der Zeichen **a** und¨ zu verstehen. Entsprechend der angewendeten Mechanismen kann man die Zeichensätze in verschiedene Gruppen einteilen, nämlich

1. Zeichensätze, die mit 256 Zeichen auskommen (8-Bit Zeichensätze)
2. Zeichensätze, die Sonderzeichen in ihre Bestandteile zerlegen
3. Zeichensätze, die einen Umschaltmechanismus verwenden
4. Zeichensätze, die mehr als 1 Byte pro Zeichen verwenden

[2] Vgl. Émile Baudot. In Wikipedia, the free encyclopedia. Bearbeitungsstand: 27 August 2011 13:01 UTC. URL: http://en.wikipedia.org/w/index.php?title=%C3%89mile_Baudot&oldid=446968306
[3] Vgl. dazu und im Folgenden: Baudot-Code. In: Wikipedia, die freie Enzyklopädie. Bearbeitungsstand: 13. August 2012, 19:36 UTC. URL: http://de.wikipedia.org/w/index.php?title=Baudot-Code&oldid=106777132
[4] Vgl. dazu und im Folgenden: Byte. In: Wikipedia, die freie Enzyklopädie. Bearbeitungsstand: 10. August 2012, 08:04 UTC. URL: http://de.wikipedia.org/w/index.php?title=Byte&oldid=106620948

Die Gruppen 1 und 2 fasst man dabei üblicherweise unter dem Begriff Single Byte Character Set zusammen, da die Grundgröße mit der diese Zeichensätze arbeiten genau 1 Byte beträgt. Zeichensätze die mehr als ein Byte für die Definition eines Zeichens verwenden, nennt man Multi Byte Character Sets. Bei den Zeichensätzen, die Umschaltmechanismen verwenden, handelt es sich hauptsächlich um Zeichensätze für Sprachen aus dem fernöstlichen Bereich, in erster Linie Chinesisch, Japanisch und Koreanisch. Hier wird in der Regel das Prinzip des Umschaltmechanismus mit dem Prinzip des Mehrbytezeichensatzes verbunden, daher fassen wir beide auch in einem Abschnitt zusammen.

Wir werden sehen, dass alle diese Ansätze ihre Vor- und Nachteile und insbesondere ihr je eigenes, spezifisches Anwendungsgebiet haben. Diese Ansätze sind grundsätzlich nicht miteinander kompatibel und in der Regel immer nur für die Anwendung mit einer Sprache bzw. einer Schrift ausgelegt. Ein Zeichensatz wird stets für eine Datei als ganzes festgelegt und kann normalerweise innerhalb eines Dokumentes nicht geändert werden.[5]

Um dieser Einschränkung abzuhelfen, gibt es seit Anfang der 90er Jahre des vergangenen Jahrhunderts mit Unicode einen übergreifenden Standard, der alle Schriften der Menschheit, sowohl der Gegenwart als auch der Vergangenheit in einem Zeichensatz definiert. Unicode ist zwar inzwischen der Standardzeichensatz in modernen Betriebssystemen und Softwareanwendungen, dennoch existieren noch viele Metadatenanwendungen, gerade auch im Bibliotheksbereich, die mit anderen Zeichensätzen arbeiten. Da Unicode einen etwas anderen Ansatz verfolgt als die "klassischen" Zeichensätze werden wir es ausführlich in einem eigenen Abschnitt behandeln.

3.1 Zeichensätze, die mit 256 Zeichen auskommen

3.1.1 ISO 646/ASCII

Den 1963 erstmals veröffentlichten Zeichensatz mit der Normnummer ISO 646 kann man als direkten Nachfolger des Baudot-Murray Codes verstehen (CCITT 3). Er verwendet 7 Bits und definiert Groß- und Kleinbuchstaben des lateinischen Alphabetes, die Ziffern von 0 – 9 und einige Satz- und Sonderzeichen. Der Standard ist an nationale Besonderheiten anpassbar: 12 Zeichenpositionen können mit länder- bzw. sprachspezifischen Sonderzeichen belegt werden; die deutsche Variante (DIN 66003) bspw. definiert zusätzlich die Zeichen $, §, Ä, Ö, Ü, ä, ö, ü und ß.[6]

Die US-amerikanische Variante von ISO 646, im gleichen Jahr unter der Normnummer ASA X3.4[7] veröffentlicht, hat jedoch gegenüber allen anderen nationalen Varianten eine Vorrangstellung errungen und sich zum universellen Standard in der Datenverarbeitung entwickelt. Unter dem Namen ASCII (American Standard Code for Information Interchange), manchmal auch US-ASCII, bildet dieser Zeichensatz quasi den kleinsten gemeinsamen Nenner, was die Darstellung von Zeichen mit Hilfe eines Computers betrifft.[8]

ASCII definiert, beginnend auf der Zeichenposition 0x20 (also die Position 32 in dezimaler Schreibweise)[9] 95 graphische Zeichen, und zwar die Groß- und Kleinbuchstaben des lateinischen Alphabets, die Ziffern von 0-9, das Leerzeichen, verschiedene Satzzeichen und Symbole, sowie verschiedene Arten von Klammern und den Rückwärtsschrägstrich. Zusätzlich definiert ASCII 33 Steuerzeichen auf den Zeichenpositionen 0x00 bis 0x1F (0 -

[5] Eine Ausnahme ist die Codierung nach ISO 2022, wie sie bspw. auch in MARC21 und Unimarc eingesetzt wird (s. u.).

[6] DIN 66003:1999-02 : Informationstechnik, 7-Bit-Code. Ausgabedatum: 1992-02. Berlin: Beuth. (Vgl. auch DIN 66003. In: Wikipedia, die freie Enzyklopädie. Bearbeitungsstand: 17. März 2012, 07:42 UTC. URL: http://de.wikipedia.org/w/index.php?title=DIN_66003&oldid=100968696)

[7] Die derzeit aktuellste Version ist: ANSI INCITS 4-1986 (R2007) : Information Systems - Coded Character Sets - 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII) (Vgl. auch American Standard Code for Information Interchange. In: Wikipedia, die freie Enzyklopädie. Bearbeitungsstand: 12. Juli 2012, 13:44 UTC. URL: http://de.wikipedia.org/w/index.php?title=American_Standard_Code_for_Information_Interchange&oldid=105489330)

[8] Es gab mit EBCDIC einen Standard für Mainframe-Rechner, der nicht mit ASCII kompatibel ist. Dieser Standard ist aber heutzutage so gut wie bedeutungslos.

[9] Die Belegung der einzelnen Bits eines Bytes lässt sich als ein Zahlenwert zwischen 0 und 255 in binärer Schreibweise begreifen, dabei steht 0 für 00000000 (alle Bits unbelegt) und 255 für 11111111 (alle Bits belegt). Den Wert eines Bytes stellt man in der Regel durch eine zweistellige Hexadezimal dar, wovon die höherwertige Stelle den Wert der ersten 4 Bits angibt und die zweite Stelle den Wert der zweiten 4 Bits (4 Bits ergeben genau 16 mögliche Kombinationen/Zustände). So kann man die den einzelnen Zeichenpositionen zugeordneten Zeichen komfortabel in einer Tabelle mit 16 Spalten (die zweite hexadezimale Ziffer) und 16 Zeilen (die erste hexadezimale Ziffer) darstellen. In der gesprochenen Sprache verwendet man jedoch oft die Dezimalzahl um eine Zeichenposition zu benennen, daher geben wir sie im Folgenden immer in Klammern mit an.

31) und 0x7F (127), davon sind einige ein direktes Erbe des Baudot-Murray Codes, wie bspw. das Zeichen auf Position 0x07 (7), das im Telegraphenzeitalter dazu diente auf der Empfängerseite eine Glocke ertönen zu lassen, um den Beginn einer Übertragung zu signalisieren. Von diesen Steuerzeichen spielen diejenigen auf den Positionen 0x1D bis 0x1F (29 bis 31) in den auf ISO 2709 basierenden Austauschformaten, wie z.B. Marc 21 oder Unimarc eine besondere Rolle als Trennzeichen für Datensätze, Felder und Unterfelder.

Der ASCII-Zeichensatz ist die Keimzelle für fast alle weiteren Zeichensätze, die ihn in der einen oder anderen Form erweitern. Programmiersprachen beschränken sich bei der Definition ihrer Syntaxbestandteile in der Regel auf diesen Zeichenvorrat des ASCII-Zeichensatzes.

3.1.2 ISO 8859

Hinter der Normnummer ISO 8859 verbirgt sich eine ganze Familie unterschiedlicher Single-Byte-Zeichensätze.[10] Insgesamt gibt es 15 verschiedene, gekennzeichnet durch eine Anhängezahl hinter der Normnummer zwischen 1 und 16 (ISO 8859-12 ist bislang nicht definiert). Alle diese Zeichensätze sind im Bereich 0x00 bis 0x1F (0 bis 127) identisch mit dem ASCII-Zeichensatz. Im Bereich 0x80 bis 0x9F (128 bis 159) werden weitere Steuerzeichen definiert. Ab der Zeichenposition 0xA0 (160) unterscheiden sich die einzelnen ISO 8859-Zeichensätze. Hier befinden sich die Zeichen anderer Alphabete (Griechisch, Kyrillisch, Hebräisch, Arabisch und Thai) oder Sonderzeichen für Sprachen, die das lateinische Alphabet benutzen:

Teilnorm	Bezeichnung	Sprachen
ISO 8859-1	Latin-1	West- und Nordeuropäische Sprachen
ISO 8859-2	Latin-2	Südosteuropäische, insb. slawische Sprachen
ISO 8859-3	Latin-3	Türkisch, Maltesisch, Esperanto
ISO 8859-4	Latin-4	Baltische Sprachen, Grönländisch, Samisch
ISO 8859-5	Latin/Cyrillic	slawische Sprachen
ISO 8859-6	Latin/Arabic	Arabisch (nicht jedoch andere Sprachen mit arabischem Alphabet)
ISO 8859-7	Latin/Geek	Neugriechisch
ISO 8859-8	Latin/Hebrew	Iwrit (keine Vokalzeichen)
ISO 8859-9	Latin-5	Türkisch
ISO 8859-10	Latin-6	Nordeuropäische Sprachen (Weiterentwicklung von ISO 8859-4)
ISO 8859-11	Latin/Thai	Thailändisch
ISO 8859-13	Latin-7	Baltisch (Weiterentwicklung von ISO 8859-4)
ISO 8859-14	Latin-8	Keltische Sprachen
ISO 8859-15	Latin-9	West- und Nordeuropäische Sprachen (Weiterentwicklung von ISO 8859-1)
ISO 8859-16	Latin-10	Südosteuropäische Sprachen, Französisch, Deutsch, Irisch

Man erkennt, dass es gerade im Bereich der Sprachen mit lateinischen Buchstaben große Überschneidungen gibt, z.B. sind die deutschen Sonderzeichen (Umlaute, Eszett) in allen Zeichensätzen außer -5, -6, -7, -8 und -11 enthalten. Besonders weit verbreitet von diesen Standards ist ISO 8859-1, der bis auf den Bereich zwischen 0x80 und 0x9F (128 bis 159) identisch ist mit dem (proprietären) Windows-1252 Zeichensatz. Hier kommt es oft zu Verwechslungen; außerdem werden beide Zeichensätze oft fälschlicherweise als ANSI-Zeichensatz bezeichnet, was natürlich noch zusätzlich zur Verwirrung beiträgt.

Der Standard 8859-11 (Latin/Thai) gehört streng genommen in die Gruppe 2, denn die Vokal- und Tonzeichen werden als sog. Combining Characters definiert, also Zeichen, die in Verbindung mit einem anderen Zeichen ein neues graphisches Zeichen bilden.

[10] Vgl. ISO 8859. In: Wikipedia, die freie Enzyklopädie. Bearbeitungsstand: 20. Juli 2012, 15:28 UTC. URL: http://de.wikipedia.org/w/index.php?title=ISO_8859&oldid=105808073

3.1.3 Nationale Standards

Vor der Definition des ISO 8859 Standards gab es eine ganze Reihe nationaler Standards, die z.T. auch heute noch neben den ISO-Zeichensätzen in Gebrauch sind. So sind z.B. für Kyrillische Schriften die Zeichensätze KOI-8R und KOI-8U gebräuchlich,[11] oder TIS-620 für Thailändisch. Letzterer unterscheidet sich vom ISO 8859-11 nur an einer Zeichenpostion, aber anders als der ISO Standard ist er als ein gültiger Zeichensatz für den MIME-type "text/plain" bei der IANA registriert.[12]

Auf einen besonderen Zeichensatz sei hier allerdings hingewiesen, und zwar den Indian Script Code for Information Interchange (ISCII).[13] Hierbei handelt es sich um einen gemeinsamen Zeichensatz für die 10 in Indien verwendeten Schriftsysteme die aus der Brahmi-Schrift abgeleitet werden. Dabei werden hier keine Schriftzeichen kodiert, sondern die Grundsilben und die Vokalzeichen. Über eine spezifische Zeichenfolge (eine Art Umschaltmechanismus), wird das jeweils darzustellende Schriftsystem angegeben, bspw. "schaltet" die Bytefolge 0xEF 0x42 auf die Devanagari-Schrift, die Bytefolge 0xEF 0x43 auf Bengalisch usw.

3.1.4 Proprietäre Zeichensätze

Neuere Versionen der gängigen Betriebssysteme verwenden in der Regel Unicode, bzw. ein Unicode-Transformationsformat für die interne Darstellung von Zeichen. Nichtsdestotrotz finden sich in Daten, die mit älteren Programmen erzeugt wurden bzw. werden oft herstellerspezifische Zeichensätze, die in der Regel auch auf dem ASCII-Zeichensatz aufbauen, aber im Bereich ab der Position 0x80 (128) eigene Wege gehen. Die gängigen Windows-Zeichensätze (man spricht hier meist von Windows-Code Pages), stimmen zwar größtenteils mit ihren ISO 8859 Pendants überein, verwenden aber den Bereich 0x80 bis 0x9F für zusätzliche graphische Zeichen.[14] Anders verhält es sich bei den Zeichensätzen für DOS-PCs – hier ist die Belegung der Zeichen ab Position 0x80 eine ganz eigene,[15] ebenso auch bei MacIntosh-Rechnern mit Betriebssystemen vor der Version OS X.[16]

Alle 8-Bit Zeichensätze lassen sich durch ein einfaches Ändern der Bytewerte in einen anderen 8-Bit Zeichensatz konvertieren, vorausgesetzt, das jeweilige Zeichen ist im Zeichenvorrat des Zielzeichensatzes enthalten. Bei der Überführung in einen Zeichensatz der mit kombinierenden Zeichen arbeitet, müssen Sonderzeichen bisweilen in Folgen von mehreren Einzelzeichen umgewandelt werden. Eine Umwandlung nach Unicode ist ebenfalls normalerweise nur eine eins zu eins Ersetzung (s. dazu unten ausführlicher).

3.2 Zeichensätze, die Sonderzeichen in ihre Bestandteile zerlegen

Insbesondere in Bereichen, wo beschreibende Metadaten zu Quellen aus verschiedenen Sprachen zusammenkommen, wie bspw. in einer Bibliothek, braucht man Zeichensätze, die Sonderzeichen für die verschiedensten Sprachen, die das lateinische Alphabet verwenden, sowie für die gebräuchlichsten Transliterationsverfahren für nicht-lateinische Schriften bereitstellen. Ein nur auf einzelne Sprachen ausgerichteter Zeichensatz reicht hier meist nicht aus.

Sonderzeichen entstehen in den allermeisten Fällen so, dass einem "normalen" Grundbuchstaben ein Zeichen über- oder untergestellt wird, aus einem **e** bspw. wird dadurch, dass man ein **^** darüber schreibt ein **ê**. Dieses

---

[11] Eine ausführliche Übersicht über die verschiedenen Zeichensätze für Kyrillisch bietet: Czyborra, Roman: The Cyrillic charset soup. [Modifiziert:] Montag, 30. November 1998 18:18:57, URL: http://czyborra.com/charsets/cyrillic.html

[12] Vgl. Character Sets. Last updated 2011-10-30. URL:http://www.iana.org/assignments/character-sets/. Eine ausführliche Darstellung des thailändischen Schriftsystems bietet: Kroonboonyanan, Theppitak: Standardization and Implementations of Thai Language. [Erstellt am:] 15.03.1999. URL: http://www.nectec.or.th/it-standards/thaistd.pdf

[13] IS 13194:1991. Vgl. Indian Script Code for Information Interchange. In: Wikipedia, the free encyclopedia. Date of last revision: 6 August 2012 09:14 UTC. URL: http://en.wikipedia.org/w/index.php?title=Indian_Script_Code_for_Information_Interchange&oldid=506048201 und Indian Script Code for Information Interchange. New Delih : Bureau of Indian Standards, 1991. URL: http://varamozhi.sourceforge.net/iscii91.pdf

[14] Zu den Windows-spezifischen Zeichensätzen s. Code pages supported by Windows. Microsoft, 2012. URL: http://msdn.microsoft.com/en-us/goglobal/bb964654 (Abrufdatum: Mittwoch, 15. August 2012 12:16:44)

[15] Zu den DOS-spezifischen Zeichensätzen s. Code pages supported by Windows : OEM code pages. Microsoft, 2012. URL: http://msdn.microsoft.com/en-us/goglobal/bb964655 (Abrufdatum: Mittwoch, 15. August 2012 12:16:44)

[16] Vgl. Mac OS Roman. In: Wikipedia, the free encyclopedia. Date of last revision: 9 July 2012 21:41 UTC. URL: http://en.wikipedia.org/w/index.php?title=Mac_OS_Roman&oldid=501464029

Verfahren, das für denjenigen, der mit der Hand schreibt, völlig selbstverständlich ist, wenden auch einige, speziell für Bibliotheksdaten entworfene Zeichensätze an, z.B. der Zeichensatz ISO 5426. Dieser Zeichensatz findet in verschiedenen bibliographischen Austauschformaten Anwendung, z.B. in Unimarc oder in MAB-2, dem ehemaligen deutschen Austauschformat.[17]

ISO 5426 erweitert den ISO 646 Zeichensatz um einige Sonderzeichen, wie z.B. das Pfund-Zeichen (£) oder das Paragraph-Zeichen (§), einige Buchstaben, die nur in einigen europäischen Sprachen vorkommen, wie z.B. die æ-Ligatur und eine Reihe sogenannter kombinierender Diakritika, neben den verbreiteten, wie ´ oder ˜ auch seltenere wie bspw. das rumänische/lettische Sedilla (, z.B. als ţ - ähnlich dem französischen Cedille¸ z.B. im ç) oder den ungarischen Doppel-Akut ˝. Um nun ein zusammengesetztes Zeichen darzustellen, gibt man zuerst das kombinierende Zeichen (ggf. auch mehrere) ein, gefolgt vom Grundbuchstaben – ganz ähnlich wie es früher mit der Tottaste auf einer Schreibmaschine gemacht wurde. Der rumänische Buchstabe **ţ** also setzt sich zusammen aus aus dem Sedilla **,** (Position 0xD2, dez. 210) und dem Buchstaben **t** (Position 0x74, dez. 116). Es obliegt der Anwendung, die später die Daten verarbeitet, die Bytefolge 0xD2 0x74 entsprechend zu interpretieren, also z.B. dafür zu sorgen, dass das diakritische Zeichen korrekt unter dem Buchstaben **t** platziert angezeigt wird. Durch dieses kombinatorische Verfahren lassen sich mit ISO 5426 insgesamt 630 gültige Zeichen erzeugen.[18]

Auf genau dem gleichen Mechanismus, jedoch mit einem anderen Zeichenvorrat, basieren auch der sog. ANSEL-Zeichensatz und der ISO/IEC 6937 Standard. ANSEL steht für "American National Standard Extended Latin Alphabet Coded Character Set for Bibliographic Use" und ist unter der Normnummer Z39.47 vom American National Standards Institute genormt.[19] Er wird im bibliographischen Austauschformat MARC21 (und auch seinen Vorgängern USMARC und CANMARC) angewendet und kann insgesamt 607 gültige Zeichen darstellen.

ISO/IEC 6937 definiert deutlich weniger kombinierende diakritische Zeichen, dafür aber eine Reihe weiterer graphischer Sonderzeichen und kann 327 gültige Zeichen darstellen. Es ist zwar ein von der IANA registrierter Zeichensatz, wird aber nur wenig benutzt.[20] Nach demselben Prinzip funktioniert auch der ISO 5427-Zeichensatz, jedoch für Text in griechischer Schrift (Altgriechisch und Neugriechisch).

An dieser Stelle sind auch die proprietären Zeichensätze für die in vielen Bibliotheken verwendeten Softwarelösungen PICA[21] und Allegro-C zu nennen. Auch wenn die aktuelleren Versionen der PICA-Software inzwischen auf Unicode setzen, kommt der proprietäre PICA-Zeichensatz noch in der immer noch verwendeten WinIBW 2000 zum Einsatz. Er kann insgesamt 616 gültige Zeichen darstellen.[22] Für Allegro-C existieren 2 Zeichensätze, einer für die DOS-Version (Allegro Ostwest, 576 gültige Zeichen) und einer für die Windows Version (Allegro Windows, 584 gültige Zeichen).[23] Dabei ist jedoch zu berücksichtigen, dass Allegro-C Datenbanken auch mit anderen Zeichensätzen arbeiten können, insofern die unteren Zeichenpositionen (0x00 bis 0x7F) identisch sind mit dem ASCII-Zeichensatz. Metadaten, die aus einer Allegro-Anwendung stammen, liegen also nicht zwangsläufig in einem der beiden Allegro-Zeichensätze vor.

Daten, die in einem Zeichensatz mit kombinierenden Zeichen vorliegen, lassen sich nur in Ausnahmenfällen zufriedenstellend in einen 8-Bit Zeichensatz konvertieren, da die Menge der möglichen Zeichen in diesen meistens deutlich geringer ist. Bei der Umwandlung nach Unicode kann in der Regel jedes Zeichen des Ausgangs-

[17] Entsprechend spricht man bisweilen auch einfach vom MAB-2 Zeichensatz.
[18] Vgl. Zeichentabelle MAB2 (ISO 5426-1983). [Modifiziert:] Samstag, 4. Dezember 2010 16:33:41. URL: http://www.gymel.com/charsets/MAB2.html
[19] Extended Latin Alphabet Coded Character Set for Bibliographic Use : ANSI/NISO Z39.47-1993 (R2003). Bethesda, Maryland : NISO Press, 1993. URL: http://www.niso.org/kst/reports/standards/kfile_download?id%3Austring%3Aiso-8859-1=Z39-47-1993%28R2003%29.pdf&pt=RkGKiXzW643YeUaYUqZ1BFwDhIG4-24RJbcZBWg8uE4vWdpZsJDs4RjLz0t90_d5_ymGsj_IKVaGZww13HuDlcPhZSkCidxbUW-I78nvIzhvY3OKzIfc5OJ8tmgIv7H3kMbTpEKfFT0%3D
[20] Vgl. ISO/IEC 6937. In: Wikipedia, the free encyclopedia. Date of last revision: 24 July 2012 02:01 UTC. URL: http://en.wikipedia.org/w/index.php?title=ISO/IEC_6937&oldid=503872324
[21] eigentlich OCLC EMEA, jedoch ist die Bezeichnung PICA nach wie vor die gebräuchlichere. Vgl. PICA (bibliotheekautomatisering). In Wikipedia, de vrije encyclopedie. Tijdstip laatste herziening: 15 maart 2012 10:48 (UTC). URL: http://nl.wikipedia.org/w/index.php?title=PICA_(bibliotheekautomatisering)&oldid=29948315
[22] S. Zeichentabelle PICA. [Modifiziert:] Samstag, 4. Dezember 2010 16:33:41. URL: http://www.gymel.com/charsets/Pica.html
[23] Vgl. Zeichentabelle allegro Ostwest (DOS). [Modifiziert:] Samstag, 4. Dezember 2010 16:33:41. URL: http://www.gymel.com/charsets/allegro-ostwest.html und Zeichentabelle allegro Windows. [Modifiziert:] Samstag, 4. Dezember 2010 16:33:41. URL: http://www.gymel.com/charsets/allegro-windows.html

zeichensatzes durch ein Unicodezeichen dargestellt werden, d.h. ein Buchstabe wird in den entsprechend Buchstaben in der Unicodetabelle umgesetzt und ein kombinierendes diakritisches Zeichen in das entsprechende kombinierende diakritische Zeichen aus Unicode. Es gibt allerdings eine Besonderheit: In den hier beschriebenen Zeichensätzen werden die Diakritika dem Grundbuchstaben immer vorangestellt, während sie in Unicode immer dem Grundbuchstaben folgen – man muss also die Reihenfolge von Grundbuchstaben und diakritischen Zeichen umkehren.

Es gibt nur wenige Anwendungen, die die hier beschriebenen kombinierenden Zeichensätze ohne weiteres anzeigen können, vielmehr wird man für eine saubere Anzeige die Daten in der Regel nach Unicode konvertieren.

3.3 Zeichensätze für fernöstliche Sprachen

Schriftsysteme, die auf Ideogrammen basieren, benötigen naturgemäß sehr viel mehr Schriftzeichen als Systeme, die auf der Darstellung einzelner Phoneme basieren. Das bekannteste Beispiel dafür ist die chinesische Schrift, die insgesamt aus mehr als 80.000 Zeichen besteht, wovon 1.500 bis 2.000 regelmäßig benutzt werden.[24] Das Japanische verwendet neben den beiden Silbenschriften (Kana) ebenfalls chinesische Schriftzeichen (Kanji), ähnliches gilt in geringerem Umfang auch für das Koreanische (Hanja). Für die Darstellung chinesischer Zeichen sind zum einen der taiwanesische Standard Big5 (für traditionelles Chinesisch) und zum anderen die nationalen Standards der Volksrepublik China, GB 2312 und GB18030 (für vereinfachtes Chinesisch) vorherrschend. Bei all diesen Zeichensätzen handelt es sich um Mehrbyte-Zeichensätze, die jedoch alle nach einem unterschiedlichen System aufgebaut sind.

3.3.1 Big5

Big5 codiert etwas über 13.000 Zeichen und verwendet dafür entweder ein einzelnes Byte – für die Zeichen des ASCII-Zeichensatzes – oder eine Kombination aus 2 Bytes.[25] Bei Verwendung einer 2 Byte-Sequenz muss das erste Byte einen Wert zwischen 0x81 (129) und 0xFE (254), das zweite einen Wert zwischen 0x40 und 0x7E oder 0x81 und 0xFE (64-126 oder 129-254) haben. D.h. ein Bytewert in dem Bereich, der im ASCII-Zeichensatz den graphischen Zeichen zugeordnet ist (0x40 bis 0x7E), muss immer im Zusammenhang mit dem davorstehenden Zeichen interpretiert werden.

3.3.2 GB 2312 und ISO 2022

GB-2312 ist ein Zeichensatz der etwas über 7.000 Zeichen des sog. vereinfachten Chinesisch enthält. Dieser Zeichensatz definiert chinesische Schriftzeichen, aber auch andere Alphabete wie japanische Kana oder kyrillische Buchstaben.[26] Die tatsächliche Codierung in einer Datei erfolgt i.d.R. entsprechend dem im Standard ISO 2022[27] beschriebenen Verfahren (dann spricht man meist von ISO 2022-CN).

ISO 2022 standardisiert ein Verfahren, innerhalb einer Datei auf verschiedene Zeichensätze zugreifen zu können. Dazu werden erst einmal 4 mögliche Mengen graphischer Zeichen und 2 mögliche Mengen von Steuerzeichen definiert; diese bezeichnet der Standard mit G0 bis G4, bzw. C0 und C1. In einer 8-Bit Umgebung[28] liegen die Zeichenmengen C0 und G0 im Bereich 0x00 bis 0x1F bzw. 0x20 bis 0x7F, also dem Bereich, in

[24] Die Volksrepublik China definiert die Kenntnis von 1500 bis 2000 Schriftzeichen als Lesefähigkeit (vgl. dazu Chinesische Schrift. In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 26. Juli 2012, 18:52 UTC. URL: http://de.wikipedia.org/w/index.php?title=Chinesische_Schrift&oldid=106042113)

[25] Vgl. Big5. In. Wikipedia, the free encyclopedia. Date of last revision: 30 July 2012 03:06 UTC. URL: http://en.wikipedia.org/w/index.php?title=Big5&oldid=504869708

[26] GB2312. In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 12. April 2012, 14:40 UTC. URL: http://de.wikipedia.org/w/index.php?title=GB2312&oldid=101972263

[27] ISO 2022 ist identisch mit ECMA-35. Vgl. Character code structure and extension techniques: ECMA-Standard 35. 6th ed. 1994. URL: http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-035.pdf

[28] ISO 2022/ECMA-35 ist auf 7-Bit und 8-Bit Umgebungen anwendbar. Bei 7-Bit Umgebungen müssen alle alternativen Zeichenmengen in den Bereich zwischen 0x20 und 0x7F geladen werden können, die Steuerzeichen auf den ersten 32 Positionen sollen nicht überschrieben werden – daher auch die Begrenzung auf 94 bzw. 96 mögliche Werte pro Byte bei Mehrbyte-Zeichensätzen (bei 96 Zeichen überschreibt

dem normalerweise der ASCII-Zeichensatz inkl. seiner Steuerzeichen definiert ist und die Zeichenmengen C1 und G1 im Bereich 0x80 bis 0x9F, bzw. 0xA0 bis 0xFF, also die Zeichenpositionen zwischen 128 und 255. Die Zeichenmengen G2 und G3 existieren erstmal nur theoretisch und sind jeweils 94 oder 96 Zeichenpositionen groß.[29] Es gibt nun verschiedene Steuerzeichen, bzw. Kombinationen aus Zeichen, die die Zeichenmengen G2 oder G3 in den Bereich von G0 oder G1 "laden"[30] können – Um eine klare Unterscheidung zwischen den Zeichenmengen G0/G1 und dem Bereich der möglichen Bytewerte zu gewährleisten, nennt man die untere Hälfte, also Position 0x20 bis 0x7F GL (graphic characters left) und die obere Hälfte, also 0xA0 bis 0xFF GR (graphic charcters right). Die Umschaltzeichen sind in einer 8-Bit Umgebung:

Name	Byte(s)	Wirkung
Locking-Shift Zero (LS0)	0x0F	GL enthält jetzt die Zeichen aus G0 (standard)
Locking-Shift One (LS1)	0x0E	GL enthält jetzt die Zeichen aus G1
Locking-Shift Two (LS2)	0x1B 0x6E	GL enthält jetzt die Zeichen aus G2
Locking-Shift Three (LS3)	0x1B 0x6F	GL enthält jetzt die Zeichen aus G3
Locking-Shift One Right (LS1R)	0x1B 0x7E	GR enthält jetzt die Zeichen aus G1
Locking-Shift Two Right (LS2R)	0x1B 0x7D	GR enthält jetzt die Zeichen aus G2
Locking-Shift Three Right (LS3R)	0x1B 0x7C	GR enthält jetzt die Zeichen aus G3

Steuersequenzen, die aus mehreren Bytes bestehen, fangen immer mit dem Zeichen 0x1B an, das im ASCII-Zeichensatz das Steuerzeichen "Escape" (ESC) ist. Daher nennt man solche Bytefolgen auch "Escape-Sequenzen".

Ebenfalls durch Escape-Sequenzen ausgedrückt wird die Zuweisung von einem Zeichensatz zu einer der 4 Zeichenmengen. Um nun bspw. den GB 2312 Zeichensatz der Zeichenmenge G1 zuzuordnen gibt man in der Datei die Bytefolge 0x1B 0x24 0x29 0x41 – das ist in ASCII-Zeichen "ESC", "$", "(", "A" – an. Da G1, solange man nichts anderes angegeben hat, schon standardmäßig im Bereich GR liegt, kann die Sequenz *Locking-Shift One Right* entfallen.

GB 2312 ist ein Zeichensatz, der 2 Byte zur Codierung eines einzelnen Zeichens benötigt, denn ein Byte innerhalb einer Zeichenmenge kann ja nur 94 verschiedene Werte annehmen. Man kann sich das nun so vorstellen, dass der Zeichensatz sich aus 94 einzelnen jeweils 94 Positionen umfassenden Zeichensätzen zusammensetzt – das erste Byte gibt jeweils den Teilzeichensatz an und das zweite die Position darin.[31]

Die Sequenzen für die Zuordnung der einzelnen Zeichensätze zu den Zeichenmengen G0-G3 legen die einzelnen Teilstandards von ISO 2022 fest. ISO 2022-JP z.B. definiert Escape-Sequenzen für die Verwendung der japanischen Zeichensätze JIS X 0201, JIS X 0208 und weiterer.[32] Gleiches gilt mit ISO 2022-KR auch für Koreanisch. Bis zur Einführung von Unicode war ISO 2022 auch die Grundlage für die Zeichencodierung in Marc21, der hier angewendete Teilstandard ist bekannt unter dem Namen Marc-8.[33]

Beim Einsatz von ISO 2022 Codierungen ist es, ähnlich wie in Big5, nicht möglich ein einzelnes Byte außerhalb des Kontextes der ganzen Datei zu verstehen. Um die Zeichen richtig zu interpretieren und umwandeln zu können muss die Datei von vorne nach hinten Byte für Byte durchgegangen werden.

man ggf. das Leerzeichen und das Delete-Zeichen). Wir beschränken uns in der Darstellung der Einfachheit halber aber auf 8-Bit Umgebungen.

[29] Das hängt davon ab, ob die Zeichensätze, die hier verwendet werden 94^n oder 96^n Zeichen definieren.

[30] Es wird dabei technisch gesehen natürlich nichts "geladen" vielmehr wird die Bedeutung der nachfolgenden Bytes umdefiniert.

[31] Denkbar wäre auch der Einsatz von Zeichensätzen, die 94^3 (also 94 x 94 x 94) Zeichen definieren, in dem Fall würde man 3 Bytes benötigen.

[32] Vgl. dazu ISO/IEC 2022. In: Wikipedia, the free encyclopedia. Date of last revision: 20 March 2012 19:24 UTC. URL: http://en.wikipedia.org/w/index.php?title=ISO/IEC_2022&oldid=482951807

[33] Vgl. Chrarcter sets and encoding options. Part 2: MARC-8 encoding environment. December 2007. In: MARC 21 Specifications for record structure, character sets, and exchange media / Library of Congress, Network Development and MARC Standards Office. URL: http://www.loc.gov/marc/specifications/speccharmarc8.html. Gleiches gilt für Unimarc: vgl. Unimarc manual, bibliographic format. 2nd ed., update 5. München : Saur, 2005. IFLA series on bibliographic control ; 14, Appendix J 4.2.

3.3.3 GB 18030

GB 18030 ist eine Erweiterung von GB 2312 und definiert über 27.000 chinesische Schriftzeichen. Diese werden als 1-Byte Zeichen (ASCII), 2-Byte Zeichen (für die in GB 2312 definierten Zeichen) oder 4-Byte Zeichen codiert. Die 4-Byte Zeichen bestehen dabei aus zwei Hälften zu jeweils 2 Bytes, von denen das erste einen Wert zwischen 0x81 und 0xFE hat und das zweite einen zwischen 0x30 und 0x39, das ergibt 1.587.600 mögliche Zeichen, mehr als derzeit im Unicode-Standard definiert sind. Auch hier ist, wie bei einer Codierung nach ISO 2022, der Kontext entscheidend um einen Bytewert dem jeweils richtigen Zeichen zuordnen zu können.[34]

3.3.4 Shift_JIS und EUC-JP

Ein häufiger Zeichensatz für das Japanische ist der von Microsoft entwickelte Shift_JIS. In diesem Zeichensatz wird ein Zeichen durch ein oder zwei Bytes dargestellt. Im unteren Bereich bis 0x7F (127) ist er identisch mit der japanischen Version des ISO 646-Standards (ASCII, jedoch ohne den Rückwärtsschrägstrich und die Tilde). Im oberen Bereich zwischen 0xA1 und 0xDF sind Katakana-Zeichen (Hankaku kana oder halbbreite Katakana) definiert. Zwei-Byte-Sequenzen beginnen immer mit einem ersten Byte zwischen 0x81 und 0x9F oder 0xE0 und 0xEF, das zweite Byte hat einen Wert zwischen 0x40 und 0x9E, wenn der Wert des ersten Bytes ungerade war, bzw. zwischen 0x9F und 0xFC wenn der Wert des ersten Bytes gerade war. Die Zeichen der Zwei-Byte-Sequenzen sind dem Japanischen Standard JIS X 0208 entnommen, die Umrechnung der Codepositionen in die Shift_JIS Sequenzen folgt einer recht komplexen Formel.[35] Die in Unix-Systemen gebräuchliche Alternative EUC-JP (Extended Unix Code) arbeitet nach dem gleichen Prinzip der variablen Zeichenlänge, verzichtet aber auf die Codierung von Hankaku-Kana im Bereich zwischen 0xA1 und 0xDF. Die halbbreiten Katakana werden durch eine 2-Byte-Sequenz dargestellt, in der das erste Byte den Wert 0x8E hat. JIS X 0208 Zeichen werden ebenfalls durch 2 Bytes dargestellt, dabei müssen beide Bytes einen Wert zwischen 0xA1 und 0xFE haben. Ein erstes Byte mit dem Wert 0x8F leitet eine 3-Byte-Sequenz für die Darstellung von Zeichen aus dem Zeichensatz JIS X 2012 (weitere Kanji-Zeichen, die nicht in JIS X 0208 enthalten sind, griechische und kyrillische Buchstaben, sowie lateinische Buchstaben mit diakritischen Zeichen) ein; das zweite und dritte Byte hat dann jeweils einen Wert zwischen 0xA1 und 0xFE.

3.4 Unicode

Unicode ist ein Standard, der versucht alle Schriftzeichen der Menschheit (gegenwärtige und vergangene) in einer einzigen Codetabelle zusammenzufassen. 1991 wurde die erste Version dieses Standards mit etwas mehr als 7.000 Zeichen veröffentlicht, die koreanischen und chinesischen Schriftzeichen (bzw. ihre japanischen Entsprechungen) fehlten damals noch. In der aktuellen Version sind über 110.000 Zeichen enthalten, darunter auch nicht-alphabetische, wie die byzantinische Neumenschrift oder Symbole, die auf Landkarten verwendet werden. Unter der Normnummer ISO 10646, bzw. der Bezeichnung "Universal Character Set" ist Unicode international standardisiert.[36]

3.4.1 Aufbau

Genauso wie einige der Zeichensätze für chinesische oder japanische Schriftzeichen beschränkt sich Unicode auf die Definition von Zeichen, bzw. der Zuordnung von Zeichen zu bestimmten Positionen in der umfassenden Codetabelle und legt keine verbindliche Weise fest, wie die Zeichen tatsächlich durch einzelne Bytes darzustellen sind. Dafür gibt es eine Reihe sog. Transformationsformate, die alternativ verwendet werden können. Auch eine Darstellung in ISO 2022 (s.o.) ist theoretisch möglich, kommt jedoch in der Praxis nicht häufig vor. Ein weiteres, bereits in Abschnitt 3.2 besprochenes Prinzip, das in Unicode zur Anwendung kommt, ist die separate Definition von Buchstaben und diakritischen Zeichen, die beliebig zusammen gesetzt werden können (der

[34] Vgl. GB 18030. In: Wikipedia, the free encyclopedia. Date of last revision: 10 August 2012 16:19 UTC. URL: http://en.wikipedia.org/w/index.php?title=GB_18030&oldid=506746209
[35] Vgl. Shift JIS. In: Wikipedia, the free encyclopedia. Date of last revision: 24 April 2012 10:15 UTC. URL: http://en.wikipedia.org/w/index.php?title=Special:Cite&page=Shift_JIS&id=488963153
[36] Vgl. Unicode. In: Wikipedia, die freie Enzyklopädie. Bearbeitungsstand: 12. August 2012, 23:12 UTC. URL: http://de.wikipedia.org/w/index.php?title=Unicode&oldid=106738367

Unicode Standard verwendet dafür den Begriff "dynamic composition").[37] Zusätzlich werden aus Gründen der Kompatibilität zu bereits vorhandenen Zeichensätzen, also bspw. die der ISO 8859-Gruppe, auch präkombinierte Zeichen definiert.

Die Unicode-Zeichentabelle ist in insgesamt 17 Bereiche, sog. "planes" unterteilt, von denen jeder 65.536 (d.i. 2^{16}) Zeichenpositionen umfasst.[38] Damit ist jedes Zeichen in einem Bereich durch einen 16 Bit, bzw. 2 Byte breiten Wert darstellbar. Die Position eines Zeichens, bzw. eines "code points" in der Unicodetabelle gibt man normalerweise durch eine 4- bis 6stellige hexadezimale Zahl an, der man die Zeichenfolge "U+" voranstellt. Die letzten 4 Ziffern der Hexadezimalzahl kennzeichnen die Position innerhalb eines Bereiches (plane), die erste oder ersten beiden Ziffern bezeichnen den Bereich selbst (man verzichtet bei der Kennzeichnung des Planes auf führende Nullen) – U+042F bspw. bezeichnet das Zeichen an der Position 1.071 (0x042F) im Plane 0, nämlich den kyrillischen Buchstaben **Я**; U+2007B das Zeichen an der Position 123 (0x007B) im Plane 2, nämlich das chinesische Schriftzeichen 龘. Gegenwärtig sind 6 Bereiche im Gebrauch:

Nummer	Bezeichnung	Verwendung
0	Basic Multilingual Plane	Schriftzeichen vorw. moderner Schriften
1 (0x1)	Supplementary Multilingual Plane	weitere historische Schriftzeichen
2 (0x2)	Supplementary Ideographic Plane	weitere chinesische Schriftzeichen
14 (0xE)	Supplementary Special Purpose Plane	Formatierungszeichen
15 (0xF)	Private Use Plane	für anwendungsspezif. Zeichen
16 (0x10)	Private Use Plane	für anwendungsspezif. Zeichen

3.4.2 Diakritische Zeichen und Normalisierung

Unicode definiert Grundbuchstaben und diakritische Zeichen als separate Einheiten, die miteinander kombiniert werden können. Dabei folgen die diakritischen Zeichen – anders als bspw. in ISO 5426 – immer dem Grundbuchstaben. Die Reihenfolge, in der die diakritischen Zeichen angegeben werden ist, soweit sie nicht um die Position konkurrieren, gleichgültig. Werden Diakritika auf derselben Position miteinander kombiniert, werden sie von unten nach oben, bzw. von links nach rechts angeordnet,[39] z.B.:

a + ˆ + ´ = á̂

a + ´ + ˆ = á̂

Gleichzeitig definiert Unicode präkombinierte Zeichen, insofern sie tatsächlich in Gebrauch sind und ggf. schon in anderen Zeichensätzen vorkommen (d.h. es gibt bspw. präkombinierte Zeichen für á und â nicht aber für á̂). Präkombinierte Zeichen (primary composite) und Zeichen, die aus einer Folge von Grundbuchstaben und diakritischem Zeichen bestehen, gelten in Unicode – wenn sie das gleiche Zeichen darstellen – als äquivalent, genauer gesagt kanonisch äquivalent (canonical equivalent), d.h. sie müssen von einer Anwendung, z.B. einem Suchindex, in genau gleicher Weise behandelt werden.

Um Kompatiblität mit anderen Zeichensätzen sicherzustellen sind manche Zeichen in Unicode codiert, die eigentlich richtigerweise durch eine Kombination von einzelnen Buchstaben dargestellt werden müssten. Z.B. U+FB00, die Ligatur **ff** , die gleichbedeutend ist mit der Zeichenfolge **ff** (U+0066, U+0066); das Verhältnis zwischen U+FB00 und U+0066, U+0066 nennt man Kompatibilitätsäquivalenz.

Unicode definiert nun einen Mechanismus, der die Gleichbehandlung dieser äquivalenten Zeichen sicherstellen soll: einen Algorithmus mit dem alle zueinander äquivalenten Zeichen in eine jeweils identische Bytefolge gebracht werden können. Möglich sind dabei vier "normalisierte" Formen:

[37] The Unicode standard. Version 6.1.0. Mountain View, CA : The Unicode Consortium, 2012. ISBN 978-1-936213-02-3. URL: http://www.unicode.org/versions/Unicode6.1.0/, S.18.
[38] ebd., S.33f.
[39] ebd., S.44f.

NFD — Eine Zeichenkombination oder ein einzelnes Zeichen wird in ihre/seine kanonischen Äquivalente überführt, dann in ihre Bestandteile zerlegt, woraufhin die kombinierenden diakritischen Zeichen geordnet werden. Dies ist die am stärksten "zerlegte" (längste) Darstellungsform

NFKD — Wie NFD, jedoch wird Kompatibilitätsäquivalenz bei der Zerlegung nicht berücksichtigt, d.h. bspw. dass U+FB00 nicht in U+0066, U+0066 überführt werden würde.

NFC — Eine Zeichenkombination oder ein einzelnes Zeichen wird in ihre/seine kanonischen Äquivalente überführt, in ihre Bestandteile zerlegt, die kombinierenden diakritischen Zeichen werden geordnet und dann wieder zu einem canonical composite zusammengesetzt. Dies ist die am weitesten zusammengesetzte, d.h. kürzeste Darstellungsform. Diese Form ist die vom W3C für die Darstellung im Web empfohlene.[40]

NFKC — Wie NFC, jedoch wird (wie bei NFKD) Kompatibilitätsäquivalenz bei der Zerlegung nicht berücksichtigt.

Um bei der Zerlegung (decomposition) die einzelnen Bestandteile in eine einheitliche Reihenfolge zu bringen, ist für jedes Unicodezeichen eine sog. Canonical Combining Class (ccc) definiert, das ist ein Zahlenwert zwischen 0 und 255. Dabei haben die Zeichen, die nicht zu anderen hinzugefügt werden können, also v.a. die Grundbuchstaben, die Klasse ccc=0. Nur diese Zeichen können am Beginn einer Zeichenkombination stehen.[41] Die folgenden kombinierenden Zeichen werden, um die Normalform NFD zu bilden, dann aufsteigend nach ihrem ccc-Wert sortiert.

Treten in einer solchen Sequenz 2 kombinierende Diakritika mit dem gleichen ccc-Wert auf, bleibt die Reihenfolge der ursprünglichen Eingabe erhalten. Das bedeutet, dass 2 Zeichenfolgen, die Diakritika mit gleichem ccc-Wert, aber in einer anderen Reihenfolge enthalten, nicht dasselbe Zeichen darstellen und dementsprechend auch nicht kanonisch äquivalent sein können. Z.B. **á̂** und **ấ**: hier haben ˆ und ´ dieselbe Canonical Combining Class, können also nicht eindeutig geordnet werden und bezeichnen demnach unterschiedliche Zeichen. Die kanonische Zerlegung eines präkombinierten Zeichens ist im Unicode-Standard jeweils unter dem Schlüssel Dm (decomposition mapping) in der Unicode-Datenbank[42] festgeschrieben.

Bei der Bildung der NFC bzw. NFKC-Form wird nun versucht, soweit wie möglich aus der geordneten Zeichenfolge präkombinierte Zeichen zu bilden und zwar von links nach rechts, d.h. ein Zeichen **ậ** würde sich in der NFC-Form aus **ạ** und ˆ zusammensetzen und nicht aus **â** und ̣ da die Reihenfolge der NFD-Form **a** (ccc=0), ̣ (ccc=202), ˆ (ccc=230) von links nach rechts ausgewertet, zuerst zur Bildung des präkombinierten Zeichens **ạ** führt.[43]

3.4.3 Codierungsformate

Der Unicodestandard definiert verschiedene Methoden, mit denen ein Code Point, also die Position eines Zeichens in der Zeichentabelle, in Bytewerte umgewandelt werden kann. Eine Möglichkeit ist es, jedes Zeichen durch eine Gruppe von jeweils 4 Bytes darzustellen, das erfordert wenig Berechnungsaufwand bei der Ermittlung der jeweiligen Zeichen durch die verarbeitende Anwendung, jedoch ungewöhnlich viel Speicherplatz bzw. Bandbreite bei der Übertragung der Daten über das Netz. Diese auf 32 Bit großen Einheiten basierende Codierung nennt man UTF-32 (UTF steht für "Unicode transformation format"), manchmal auch UCS-2.

Da die Zeichen aus den Planes 1 bis 16 im Vergleich zu denen aus dem Basic Multilingual Plane eher selten verwendet werden, kann man durch die Darstellung einer Zeichenposition durch eine 16 Bit große Einheit (2 Bytes), gegenüber UTF-32 meist die Hälfte des Speicherplatzes sparen. Ein Unicodezeichen aus dem Bereich 0 wird ohne weitere Umwandlung als eine 2 Byte lange Zahl dargestellt. Für Zeichen aus höheren Bereichen wird

[40] Vgl. ebd., Anh. 15.
[41] ebd., S. 101ff.
[42] http://www.unicode.org/Public/UNIDATA/. Zum einfachen Nachschlagen ist die Verwendung von Anwendungen wie Codepoints oder Graphemica (s. Abschn. 4) bequemer.
[43] Ausführlich zur Bildung der Normalformen vgl. The Unicode standard, Anh. 15.

von der Codeposition der Wert 0x10000 (65.536) subtrahiert; die daraus resultierende Bit-Sequenz wird in 2 Blöcke zu je 10 Bit aufgeteilt, von denen dem ersten die Bitfolge 110110 und dem zweiten die Bitfolge 110111 vorangestellt wird. So erhält man 2 2-Byte lange Zahlen mit einem Wert zwischen 0xD800 und 0xDFFF, die in Verbindung miteinander ein Zeichen codieren. Die Positionen 0xD800 bis 0xDFFF sind eigens für diese Codierungsform freigelassen worden, so dass eine 4-Byte Sequenz eindeutig am vorliegenden Bytewert erkannt werden kann – die Unterscheidung ob es sich um den ersten Teil oder den zweiten Teil dieser Sequenz handelt (man sprich hier von high surrogate und low surrogate) ist auch eindeutig möglich.[44]

Für UTF-16 gibt es, wie auch für UTF-32, zwei Varianten, was die Reihenfolge der Bytes eines jeden Paares bzw. einer Vierergruppe betrifft. Es ist möglich entweder das höherwertige Byte zuerst zu speichern und dann das niedrigere oder umgekehrt. Für das Zeichen **a** würde das bedeutet, dass die Bytefolge in UTF-16 entweder 0x00 0x61 sein kann oder 0x61 0x00. Dieses Phänomen kommt unter dem Endianness auch in anderen Bereichen der Informatik vor – man unterscheidet "big endian" (z.B. UTF-16BE) und "little endian" (z.B. UTF-16LE), wobei big endian die als natürliche empfundene Reihenfolge ist (also 0x00 0x61 für **a**).

Um in UTF-16 codierte Daten zwischen Systemen mit unterschiedlicher Endianness fehlerfrei austauschen zu können, kann man an den Anfang einer Datei das Unicodezeichen U+FEFF (zero-width no-breaking space) – ein Leerzeichen ohne Breite – setzen. Verwenden nun beide am Austausch beteiligten Systeme eine unterschiedliche Endianness, wird dieses Zeichen vom empfangenden System als U+FFFE – das ist eine nicht definierte Zeichenposition – interpretiert. Das empfangende System weiß nun, dass es die Reihenfolge der Byte-Paare beim Auswerten ändern muss. Dementsprechend nennt man dieses Zeichen Byte Order Mark (BOM).[45]

Für Text der vornehmlich in westeuropäischen Sprachen verfasst ist, ist das Codierungsformat UTF-8 noch effizienter, was den benötigten Speicherplatz bzw. die zu übertragende Datenmenge betrifft. UTF-8 ist ein Format das Bytesequenzen unterschiedlicher Länge verwendet, und zwar werden die Sequenzen länger, je höher die Position eines Zeichens in der Zeichentabelle ist. Eine Bytesequenz wird nach dem derzeitigen Unicodestandard jedoch nicht länger als 4 Byte. Dieses Codierungsformat ist das am weitesten verbreitete Darstellungsformat für Unicode und oftmals werden die Begriffe Unicode und UTF-8 auch synonym verwendet.

Zeichen aus dem Bereich von 0x00 bis 0x7F (0 bis 127), der in Unicode identisch ist mit dem ASCII-Zeichensatz, werden durch ein einzelnes Byte ausgedrückt. Damit liegt jede nach US-ASCII codierte Datei automatisch zugleich in gültigem UTF-8 vor. Zeichen ab der Position 0x80 (128) werden folgendermaßen codiert: Die Binärdarstellung einer Zeichenposition wird in Gruppen zu 6 Bit eingeteilt, die Gruppe mit den höchstwertigen Stellen muss kürzer sein als 6 Stellen, und zwar mindestens um die Anzahl der Gruppen die man gebildet hat – ist das nicht der Fall füllt man mit führenden 0-Stellen auf, bis eine neue Gruppe entsteht. Jeder vollständigen 6er Gruppe wird "10" vorangestellt (d.h. es wird 128 hinzuaddiert), so dass sich Bytewerte zwischen 0x80 und 0xBF ergeben. Der ersten Gruppe stellt man für jede Gruppe der Sequenz eine 1 voran, d.h. hat man 2 Gruppen, eine vollständige 6er Gruppe und eine unvollständige, beginnt das erste Byte mit "11", darauf muss eine 0-Stelle folgen. Ist das Byte jetzt noch kürzer als 8 Bit werden weitere 0-Stellen eingefügt.

Zum Beispiel das äthiopische Zeichen ሁ (hu), U+1201:

1.	0x1201 (4609) in Binärdarstellung:	1001000000001
2.	Gruppierung zu jew. max 6 Stellen:	1 001000 000001
3.	den vollständigen 6er Gruppen wird 10 vorangestellt	1 10001000 10000001
4.	Im ersten Byte wird die Anzahl der Gruppen abgebildet:	1111 10001000 10000001
5.	gefolgt von 0:	11101 10001000 10000001
6.	und mit 0-Stellen aufgefüllt:	11100001 10001000 10000001
7.	das ergibt in hexadezimaler Darstellung:	0xE1 0x88 0x81

Damit ist jeweils am ersten Byte einer Sequenz abzulesen, wie lang diese ist und die Folgebytes sind durch ihren Wertebereich auch jeweils als solche erkennbar.

[44] Vgl. UTF-16. In: Wikipedia, die freie Enzyklopädie. Bearbeitungsstand: 14. Juni 2012, 09:12 UTC. URL: http://de.wikipedia.org/w/index.php?title=UTF-16&oldid=104372993

[45] Vgl. auch Byte Order Mark. In: Wikipedia, die freie Enzyklopädie. Bearbeitungsstand: 25. Juli 2012, 17:05 UTC. URL: http://de.wikipedia.org/w/index.php?title=Byte_Order_Mark&oldid=106000822

UTF-8 kennt nur eine mögliche Bytereihenfolge und ein Byte Order Mark ist daher nicht erforderlich, dennoch fügen einige Programme die Bytefolge 0xEF 0xBB 0xBF (also die UTF-8 Repräsentation von U+FEFF) in UTF-8 codierte Dateien ein – was wiederum bei anderen Anwendungen bisweilen zu Problemen führen kann.

UTF-8, UTF-16 und UTF-32 sind die vom Unicode Standard selbst definierten Codierungsschemata. Darüber hinaus gibt es noch andere wie bspw. UTF-7 oder Punycode, die im Metadatenbereich allerdings keine besondere Rolle spielen und daher hier auch nicht weiter behandelt werden sollen.

Unicode in der Codierungsvariante UTF-8 ist der Standardzeichensatz für Daten, die in XML codiert vorliegen – auch in allen anderen Bereich setzt er sich stärker und stärker durch. Kritik, die an diesem Standard geäußert wurde bezieht sich auf die Vereinheitlichung der ideographischen Zeichen chinesischen Ursprungs (die sog. Han Unification), die auch die japanischen Kanj und koreanischen Hanja umfasst und sich sehr stark an der chinesischen Tradition orientiert ohne japanischen und koreanischen Gegebenheiten ausreichend Rechnung zu tragen. Das hat in diesen Ländern zum Teil zu einer Zurückhaltung gegenüber Unicode geführt.[46] Für den Metadatenbereich, gerade auch im europäischen Raum ist er inzwischen der wichtigste Standard für die Zeichencodierung.

4 Nützliche Tools

Character set converter – online tool
Formularbasierter Konverter für die meisten existierenden Zeichensätze von Joel Yliluoma.
<http://kanjidict.stc.cx/recode.php>

Codepoints
Eine Datenbank mit allen Unicodezeichen, inkl. Beschreibung zu ihrer Verwendung und Darstellung in verschiedenen Unicode-Transformationsformaten. <http://codepoints.net/>

Graphemica
Zeichendatenbank, die für jedes Unicodezeichen verschiedene Codierungen, graphische Repräsentationen und ähnliche Zeichen aufführt. <http://graphemica.com/>

Unisearcher
Zeichendatenbank, in der u.a. anhand verschiedener Codierungsvarianten (z.B. UTF-8, Shift_JIS) gesucht werden kann <http://www.isthisthingon.org/unicode/>

Universal Cyrillic Decoder
Eine Webanwendung mit der man kyrillischen Text von einer beliebigen Codierung in eine andere konvertieren kann. <http://2cyr.com/decode/>

UTF-8 Codetabelle
Eine Tabelle der UTF-8 Sequenzen aller Unicodezeichen von Thomas Schild. <http://www.utf8-zeichentabelle.de/>

Zeichentabellen
Umfangreiche Zeichentabellen zu den im bibliothekarischen Bereich eingesetzten Zeichensätzen bietet Thomas Berger auf seiner Website unter: <http://www.gymel.com/charsets/>

[46] Vgl. dazu ausführlich: Han-Vereinheitlichung. In: Wikipedia, die freie Enzyklopädie. Bearbeitungsstand: 27. Juli 2012, 11:17 UTC. URL: http://de.wikipedia.org/w/index.php?title=Han-Vereinheitlichung&oldid=102198599