Benennung der Unicode CodePoints

Die CodePoints der BMP (Base Multilingual Plane) werden durch

U+hhhh

benannt, wobei hhhh der Hexwert des betreffenden Zeichens ist. Für die höherwertigen Planes wird die Hexzahl der betreffenden Plane vorangestellt, also

U+1hhhh

für ein Zeichen in der Plane 1.

Es gibt 17 Ebenen (Planes) à 65536 Zeichen, daher insgesamt 1.114.112 CodePoints. Davon sind aber nur ca. 250.000 definiert und mehr als 860.000 noch verfügbar.

U+0000 erster CodePoint in Plane 0

U+100000 letzter CodePoint in Plane 17

Beispiele für Unicode-Codepoints:

A	U+0041	lateinisches A
A	U+0391	griechisches A
A	U+0410	kyrillisches A
א	U+05d0	hebräisches A (Alef)

Diakritische Zeichen

Diakritische Zeichen sind besondere Auszeichnungen von Buchstaben, die eine abweichende Aussprache des Grundbuchstabens andeuten. Viele dieser Zeichen mit diakritischen Ergänzungen sind bereits als eigene Zeichen im Unicode definiert.

Etwa finden wir ab U+0080, dem Latin-1 Supplement, an der Stelle U+00C4 das Zeichen Ä, Umlaut-A. Aber nicht alle dieser möglichen Zeichenkombinationen sind auch als eigene Zeichen ausgeführt. Daher kann man diese fehlenden Zeichen durch eine Kombination aus dem Basiszeichen und einem der diakritischen Zeichen komponieren. Im Falle des Umlaut A wäre das das Basiszeichen A (U+0041) und das Zeichen „*Combining Diaresis*“(U+0308).

Zeichen	Kode
Ä	C4 00
Ä	41 00 08 03

Die Bytefolge 41 00, die üblicherweise das Zeichen A darstellt, auch einmal ein Umlaut-A sein kann, wenn nämlich die Byte-Folge für das diakritische Zeichen¨ 08 03 folgt. Ohne den Zusammenhang im Bytestrom zu kennen, kann man daher nicht eindeutig sagen, welches Zeichen mit der Bytefolge 41 00 gemeint ist.

Kodierung

Eine Kodierung legt nun fest, wie ein Zeichen als eine Folge von Bytes im Speicher oder in einer Datei gespeichert wird. Jede dieser Kodierungen wird als CodePage definiert.

Die naheliegendste Version ist natürlich die, dass man aufeinanderfolgenden Zeichen des Zeichenvorrats auch dieselben Nummern in einem Koderaum zuordnet, damit die bisherigen Nummerierungen, etwa die des ASCII-Kode und anderer Alphabete unverändert dargestellt werden.

In der ersten Definition des Unicode gab es eine Ebene (Plane) und daher 65536 Zeichen. Daher war es naheliegend, für ein Zeichen zwei Bytes zu verwenden. Damit war genau diese Zeichenzahl 2^16 darstellbar.

UTF-16

Diesen Code nannte man UTF-16 (*Universal Multiple-Octet Coded Character Set* **(UCS)** *Transformation Format*)

Schon bei der Definition des Unicode 1991 war bekannt, dass es weitere Planes geben wird und daher war auch schon bei der Definition von UTF-16 die Verarbeitung der höherwertigen Planes zu berücksichtigen. UTF-16 kann daher mit zwei Bytes alle Zeichen der BasePlane darstellen und mit einem besonderen Kodierungsverfahren auch alle anderen Zeichen der höherwertigen Planes Alle Zeichen außerhalb der Base Plane (d. h. U+10000 bis U+10FFFF) werden durch vier Bytes dargestellt.

Der Algorithmus für CodePoints über ffff ist wie folgt:

Der CodePoint hat folgenden höchsten Wert (21 Bit):

U+10ffff

Von diesem Wert wird 10000h (=65536) abgezogen und das nunmehr 20-stellige Ergebnis in zwei Blöcke zu je 10 Bit aufgeteilt.

Dem ersten Block wird die Bitfolge 1101.10 (=D8) und dem zweiten die Bitfolge 1101.11 (=DC) vorangestellt.

Alle so entstandenen Kodes befinden sich im Zeichenbereich zwischen D8xx (*High Surrogate*) und DCxx (*Low Surrogate*).

Diese Codierung hat zur Folge, dass alle *Code Points* aus den höherwertigen Planes durch zwei Bytes aus dem Bereich D8xx-DCxx kodiert werden. In diesem Bereich der Base Plane sind daher keine Zeichen kodiert.

Beispiel:

Ein ägyptischer Apis-Stier (einer von vielen) hat folgende Hieroglyphe (nur sichtbar mit dem kostenlosen Font Aegyptus):

U+F0986

Wir rechnen:

```
 F0986
+10000
 E0986 = 1110 0000 10-01 1000 0110

1101.1011 1000 0010 = DB82
1101.1101 1000 0110 = DD86
```

Dieses Zeichen wird daher im UTF-16-Kode durch die zwei Hexzahlen DB82 und DD86 kodiert.

Es ist bemerkenswert, dass diese Kodes mit keinem anderen Kode der BasePlane verwechselbar sind, weil ihre Zeichenraum bereits in der BasePlane genau für diesen Zweck reserviert worden ist. Die Definition des Zeichenraumes und der späteren Kodierung hängen also eng zusammen. (siehe Bild unten)

Die UTF-16-Kodierung ist daher eine Kodierung mit ungleicher Länge für Zeichen der Plane 0 und der anderen Planes,

UTF-32

Mit der Hinzunahme der höherwertigen Planes definierte man den (wegen seines Platzbedarfs eher selten verwendeten) Kode UTF-32, der für die Kodierung von einem CodePoint 4 Bytes=32 Bit benötigt. Dieser Kode erlaubt einen wahlfreien Zugriff auf irgendein Zeichen eines Textes, weil jedes Zeichen mit derselben Anzahl von Bytes kodiert wird.

Leider ist auch das nicht ganz richtig, denn es gibt eine große Zahl diakritischer Zeichen, die erst in Kombination mit einem Basiszeichen ein neues Zeichen ergeben.

Wenn daher diakritische Zeichen verwendet werden, gibt es nicht einmal bei einer UtF-32-Kodierung wahlfreien Zugriff. Wenn daher in einem Text zum Beispiel 10 diakritische Zeichen verwendet werden, dann ist die Gesamtzahl der dargestellten Zeichen um 10 kleiner als es der Datenlänge entspricht.

UTF-8

Da alle Internet-Protokolle auf dem 7-Bit-ASCII-Zeichensatz aufbauen, wäre es eine Bandbreitenverschwendung, würde man diese Protokolle mit einem platzhungrigen Kode wie dem UTF-16 oder gar UTF-32 abwickeln.

Anderseits reicht zwar ASCII allein für die Protokoll-Elemente aus, nicht aber für die Inhalte. Gesucht ist daher ein Kode, der sich für die Protokollelemente wie ein platzsparender 8-Bit-Kode verhält und für die Inhalte für die häufig benutzten Zeichen mit möglichst wenigen Bytes auskommt und nur für die sehr seltenen Zeichen auch einmal mehr Bytes verbraucht. Ein solcher Kode ist UTF-8.

Bits	CodePoint	Kodierung
7	0…127	0xxxxxxx
11	80…7FF	110xxxxx 10xxxxxx
16	800…FFFF	1110xxxx 10xxxxxx 10xxxxxx
21	10000…10FFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

UTF-7

Der UTF-7-Kode war dafür gedacht, den Unicode-Zeichenraum über 7-Bit-Kanäle zu übertragen. Diese Kodierung hat sich nicht durchgesetzt.

Speicherung der Kodes

Bei allen Kodierungen, die mehr als nur ein Byte pro CodePoint benötigen (UTF-16, UTF-32), kann die Speicherung mit dem niederwertigen Byte beginnen *(Little Endian)* oder mit dem höherwertigen Byte *(Big Endian)* beginnen. Wie der Rechner das handhabt, hängt von seiner inneren Verarbeitung ab. Auf Windows-Rechnern wird die Variante *Little Endian* verwendet.

Um beide Möglichkeiten anbieten zu können, gibt es die Kodes UTF-16 und UTF-32 in beiden Varianten mit jeweils anderer Codepage-Nummer.

CodePage	Kodierung	Zeichen	Kode
65000	UTF-7	A	41
65001	UTF-8	A	41
1200	UTF-16	A	41 00
1201	UTF-16-BE	A	00 41
12000	UTF-32	A	41 00 00 00
12001	UTF-32-BE	A	00 00 00 41

Wie speichert die Zeichen ein Rechner?

Windows benutzt intern zur Zeichenablage UTF-16-LE, Unix verwendet UTF-8.

Hier scheint Windows im Vorteil zu sein, denn der Zugriff zu einem Zeichen scheint rascher

Alle Zeichen der höherwertigen Planes werden bei UTF-16-Kodierung mit zwei Bytes in dem dafür reservierten Kodebereich D800-DCFF, der so genannten *Surrogates* projiziert.