Zeichenkodierung

by

Dr. Günter Kolousek

Überblick

- Kodierung (auch Code): Abbildung, die jedem Zeichen eines Quellalphabets (Menge!) eindeutig ein Zeichen eines Zielalphabets zuordnet.
  - kodieren vs. dekodieren
  - Kodierung mit
    - fixer Länge (z.B. ASCII)
    - variable Länge (z.B. UTF-8)
- Zweck
  - Speicherung
  - Informationsaustausch
  - Verarbeitung

ASCII

- ASCII
  - American Standards Code of Information Interchange
    - definiert durch ANSI (American National Standards Institute)
  - 26 Zeichen des lateinischen Alphabets in Klein- und Groß
  - Satzzeichen, Akzentzeichen,…
    - Fernschreiber: ä äquivalent zu a BS ¨
  - Steuerzeichen: CR, LF, FF,…
  - 7 Bits je Zeichen
    - damit 8. Bit für Fehlerüberprüfung ($\rightarrow$ parity checking)
- “extended ASCII”
  - verschiedene Erweiterungen auf 8 Bits

ISO 8859

- 8-Bit Zeichensätze
  - 7-Bits wie ASCII
- Varianten
  - 8859-1 … westeuropäisch (latin-1)
  - 8859-2 … mitteleuropäisch (latin-2)
  - …
  - 8859-15 … westeuropäisch (latin-9)
    - $\rightarrow$ €,…!

Unicode

- Standard zum Erfassen aller Zeichen
- UCS (Universal Character Set)
  - Menge aller Codepoints (engl. code point)
    - auch mehrere Codepoints, z.B. Ä: `U+00C4` oder `U+0041` `U+0308`, d.h. A + ¨)
  - Aussehen $\rightarrow$ glyph
  - Menge aller glyphs $\rightarrow$ font
- anfangs weniger als 65535 Zeichen (16 Bits)
  - früher: $\rightarrow$ Kodierungen UCS-2 und UCS-4 (ISO/IEC 10646)
    - UCS-2 $\equiv$ UTF-16 in genau 2 Bytes, UCS-4 $\equiv$ UTF-32
  - heute (seit 1991): UCS parallele Entwicklung zu Unicode
- heute mehr als 100000 Zeichen erfasst
  - Codepoints derzeit bis 0x10FFFF (21 Bits!)
  - Unicode 14: 144.697 Codepoints
  - Anzahl der zur Verfügung stehenden Bereiche (nach Abzug aller reservierten Bereiche): 1.111.998

Unicode – Kodierungen

- UTF: Unicode Transformation Format
- UTF-8
- UTF-16
- UTF-32

UTF-8

- Kodierung mit variabler Länge (1-4 Bytes)
- Regeln
  - Codepoint mit 7 Bits: `0xxxxxxx` (ASCII)
  - Codepoint mit 11 Bits: `110yyyyx 10xxxxxx`
  - Codepoint mit 16 Bits: `1110zzzz 10zyyyyx 10xxxxxx`
  - Codepoint mit 21 Bits: `11110uuu 10uuzzzz 10zyyyyx 10xxxxxx`
  - könnte erweitert werden (derzeit nicht definiert)
- Kein Problem mit “endianess” ($\rightarrow$ Daten und Interoperabilität)
  - $\rightarrow$ Folge von Bytes

UTF-16

- Kodierung mit variabler Länge (16 bzw. 32 Bits)
- Regeln
  - Codepoint C aus UCS-2, d.h. 16 Bits: C < 0x10000 ✓
    - d.h. in Bereichen U+0000–U+D7FF und U+E000–U+FFFF
    - Codepoint mit 7 Bits: `00000000xxxxxxx` (ASCII)
    - Codepoint mit 11 Bits: `00000yyyyxxxxxx`
    - Codepoint mit 16 Bits: `zzzzyyyyxxxxxx`
  - Codepoint C aus UCS-4, d.h. effektiv 21 Bits:
    - C' = C - 0x10000 $\rightarrow$ C' $\leq$ 0xFFFFF, daher 20 Bits!
    - `10-19 + 0xD800` ↓ `0-9 + 0xDC00` ↓
    - `110110hhhhhhhhhh` `110111llllllllll`

UTF-32

- Jeder Codepoint genau 4 Bytes
- praktisch identisch zu UCS-4
- Regeln
  - Codepoint mit 7 Bits: `00000000000000xxxxxxx`
  - Codepoint mit 11 Bits: `00000000000yyyyxxxxxx`
  - Codepoint mit 16 Bits: `00000zzzzzyyyyxxxxxx`
  - Codepoint mit 21 Bits: `uuuuuzzzzzyyyyxxxxxx`
- Vorteile
  - Zugriff über Zeigerarithmetik auf beliebiges Zeichen
    - aber nicht bei zusammengesetzten Zeichen (d.h. 1 Zeichen = mehrere Codepoints)
    - aber meist werden Zeichen zeichenweise gelesen!
- Nachteile
  - Platzbedarf!!

Kodierungen $\rightarrow$ ASCII

- `base64`: siehe Folien `http1a`
- `quoted-printable`
  - ASCII-Zeichen von 127-255 hexadezimal als =XY
- URL-Encoding: siehe Folien `http1a`
- Puny-Code
  - Unicode auf “a” bis “z”, “0” bis “9” und “-”