tekom-Regionalgruppe Berlin-Brandenburg · 27.9.2010

Von Zeichen und Wundern

Unicode erfolgreich nutzen

Yves Forkl

Satz-Rechen-Zentrum H+H GmbH & Co. KG

Überblick

- Das Problem: Chaos der Zeichenkodierungen
- Eine Lösung: Unicode
- Zwei Herausforderungen: Unicode kodieren und dekodieren
- Helfer im Alltag: Praktische Tipps zu Unicode

Das Problem:
Chaos der Zeichenkodierungen

Die „Schattenseite“ von ASCII

- *gut:* ASCII standardisiert (seit 1967!) 128 Zeichen für die ersten 7 bit, d.h. Byte-Werte 0 bis 127 als einen „Zeichensatz“; das 8. bit wurde damals für Steuerungszwecke verwendet
- *schlecht:* ASCII ignoriert die übrigen 128 Zeichen, die in einem aus 8 bit bestehenden Byte in den Byte-Werten 128 bis 255 möglich sind (wie ein Schatten des ASCII-Zeichensatzes...)

ASCII und die fatalen Folgen (1)

- Chaos Hunderter unterschiedlicher Zeichensätze mit eigen(sinnig)er Belegung der Byte-Werte 128 bis 255
- z.B. Windows-„Code pages“ 437, 850 u.a.; Windows-1252 = „Windows Westeuropäisch“
- z.B. ISO 8859-1 (= Latin 1, für Westeuropa), ISO 8859-2 (= Latin 2, für Mitteleuropa), ISO 8859-15 (= Latin 9, für Westeuropa), sowie Arabisch, Kyrillisch, Hebräisch usw.

ASCII und die fatalen Folgen (2)

- fehlendes Problembewusstsein in Sachen Zeichenkodierung (vgl. „erweitertes ASCII“, „ASCII mit Umlauten“, „einfacher Text“)
- geringe Standardisierung, dadurch komplizierte Sachlage und hohe Kosten durch inkompatible Kodierungen
- Unkenntnis oder schlichte Verzweiflung, darum bis zum heutigen Tag Verwirrung und massive Kodierungsfehler bei Entwicklern und Anwendern

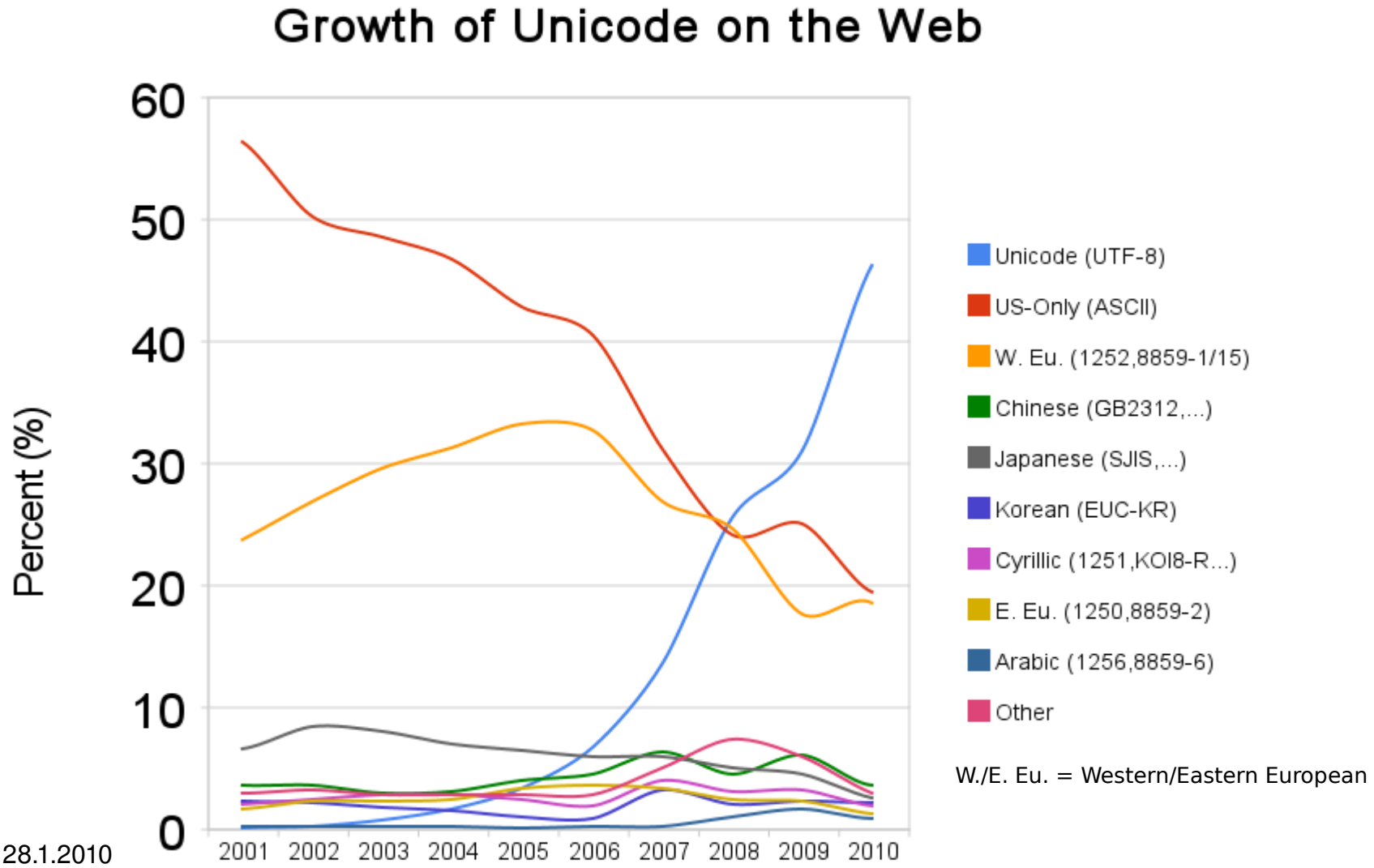
Eine Lösung:
Unicode

„Unicode“?

- „Unicode“ (universal character encoding) oder „ISO 10646 / UCS“? — fast egal
- Anfänge in den späten 1980er Jahren, 1991 Version 1.0, 2009 Version 5.2
- Zitat: *Unicode has been the most important contribution of these last years: thanks to this standard, computers have become a universal tool for communication and exchange.*

  Bernard Desgraupes, *Passeport pour Unicode*

Verbreitungsgrad von Unicode



W./E. Eu. = Western/Eastern European

Quelle:
Google Blog 28.1.2010

„Einer für alle, alle für eines“

- internationaler Standard, für alle verfügbar
- ein einziger Zeichensatz für alle Zeichen der Welt, zu allen Zeiten
- alle „Positionen“ entsprechen jeweils genau einem Zeichen
- ein Zeichen ist jetzt mehr als ein Byte (im doppelten Sinn)

Ein einziger Zeichensatz für alle Zeichen der Welt

- für jedes sinntragende Schriftzeichen oder Textelement aller bekannten Schriftkulturen und Zeichensysteme, mit eindeutiger Beschreibung jedes Zeichens
- inklusive Zeicheneigenschaften, z.B. Einteilung in Klassen, Sortierverhalten, fliegende Akzente, Normalisierung u.a.
- zusätzlich: feste Bereiche im Zeichensatz zur sogenannten privaten Nutzung

Alle „Positionen“ entsprechen jeweils genau einem Zeichen

- laut Wikipedia: *Ziel ist es, die Verwendung unterschiedlicher und inkompatibler Kodierungen in verschiedenen Ländern oder Kulturkreisen zu beseitigen.*
- systematische Benennung und Anordnung in Blöcken ähnlicher Zeichen (auch exotische und historische)
- laufende Erweiterungen, verwaltet durch Gremien; z.B. 2008 neu: Zeichen U+1E9E (Aufgabe: Was verbirgt sich dahinter?)

Codepoints: verhexte Nummern (1)

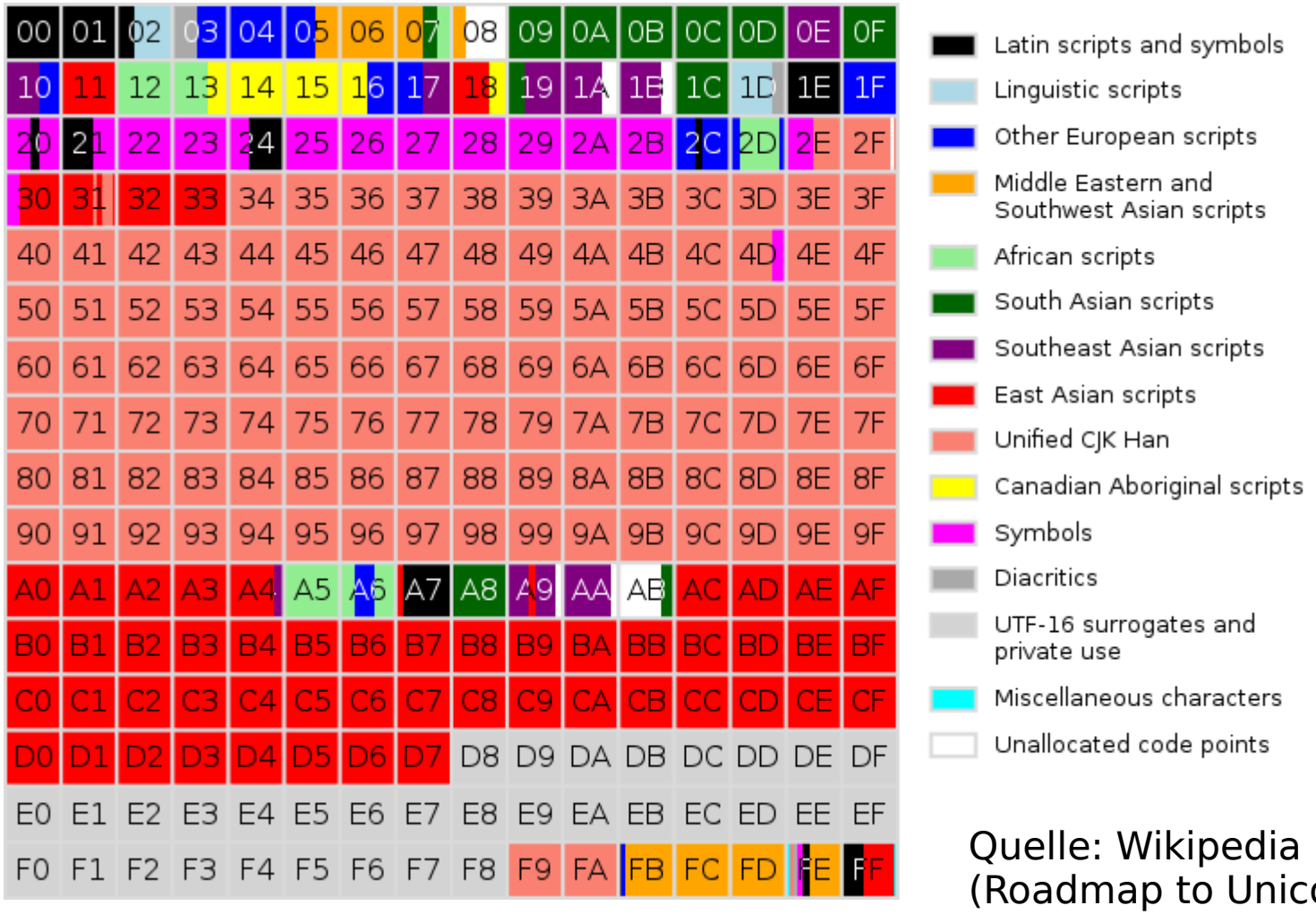
- *Codepoint*: laufende Nummer im Zeichensatz, global auf ewig eindeutig; kein Bytewert!
- erste 256 Unicode-Codepoints entsprechen ASCII und ISO 8859-1 (aber anders kodiert)
- Wert meist hexadezimal statt dezimal notiert:
  - $10_{10} = A_{16}$ | $15_{10} = F_{16}$ | $16_{10} = 10_{16}$ | $128_{10} = 80_{16}$
    $255_{10} = FF_{16}$ | $256_{10} = 100_{16}$ | $65.535_{10} = FFFF_{16}$
  - Grund: Hexadezimalzahlen sind „kompatibler“ mit Binärzahlen, d.h. mit bits und Bytes

Codepoints: verhexte Nummern (2)

- Notation zur Referenz auf ein Unicode-Zeichen: **U+***hhhh...* (hexadez. Ziffern), z.B.:
  - U+0020 (dez. 32) = Leerzeichen
  - U+0041 (dez. 65) = „A“
  - U+00FF (dez. 255) = „ÿ“
- Zeichengruppierung nach Hexadezimalzahlen (nach Stelligkeit in Bereiche, darin Blöcke)
- Unicode-Blöcke innerhalb der Bereiche, z.B.:
  - U+0000 bis U+00FF: wie ISO 8859-1
  - U+0370 bis U+03FF: griechische Zeichen
  - U+2000 bis U+206F: allgemeine Interpunktion
  - vgl. Übersicht auf der nächsten Folie

Unicode-Blöcke im Bereich 0

(Plane 0 = Basic Multilingual Plane)



Quelle: Wikipedia
(Roadmap to Unicode BMP)

Ein Zeichen: jetzt *mehr* als ein Byte

- derzeit in Unicode ca. 100.000 Zeichen (Stand 2008, V. 5.1.0) von etwa 1 Mio. möglichen; am wichtigsten: 65.536 Zeichen bis U+FFFF = *Basic Multilingual Plane* = Bereich 0
- Codepoint ist unabhängig vom Bytewert
- Zeichen und Bytes sind in Unicode anders als bei früheren Kodierungen (ASCII & Co.) nur mehr indirekt verbunden und verkörpern nun sogar verschiedene Konzepte

Unicode-Begriffe begreifen

- Konzepte der Unicode-Zeichenwelt: Zeichen, Codepoint, Glyph, Kodierung
  - Zeichen: unsichtbar, weil nur eine Verbindung zwischen Codepoint und Zeichenbeschreibung
  - Glyph: visuelle Darstellung einer Zeicheninstanz in einer bestimmten Schrift
  - Kodierung: Vorschrift zur computerinternen Repräsentation der Instanzen eines Zeichens als Bytes

Ein Zeichen: jetzt mehr als *ein* Byte

- ein Zeichen wird elektronisch erst dann greifbar, wenn es in Bytes transformiert wird, um diese zu speichern oder zu verarbeiten
- wenn Unicode-Zeichen in Bytes umgesetzt werden, wird ein Zeichen (oft) durch mehr als ein Byte repräsentiert
- aus gutem Grund gibt es mehrere Verfahren zur *Kodierung* eines Unicode-Zeichens in Bytes

Kodierung von Unicode-Zeichen als Bytefolgen

- Grundproblem: „Wie sag ich's meinem Computer?“, d.h. wie wird jedes Zeichen intern repräsentiert? (Wegfall der Äquivalenz Zeichen = Byte)
- *Transformation* von Codepoints in Bytefolgen: meist entspricht ein Codepoint mehreren Bytes
- verschiedene *Unicode Transformation Formats* (UTF)

Unicode transformation formats

- Erster, naiver Ansatz zur Kodierung: Wert des Codepoints in 2 Byte = 16 bit ausgedrückt, daher „UTF-16“
- reicht eigentlich nur für 65.536 Zeichen, aber mit etwas Trickserei auch für 1 Mio. Zeichen
- Probleme: verschwenderisch wegen 2 Byte für jedes Zeichen; kompliziert; abhängig von plattformspezifischer Byte-Reihenfolge

Byte-Reihenfolgen: Hilfe, wo ist vorne und hinten?

- Reihenfolge der Kodierung von Bytes für Ganzzahlen (z.B. ein Codepoint) ist abhängig vom Computertyp
- *Big-Endian* (vgl. Zahlwort „tausendzwei“) vs. *Little-Endian* (vgl. Zahlwort „einundzwanzig“)
- heute übliche PCs auf Intel-Basis (alle Betriebssysteme) sind Little-Endian
- bewirkt Uneindeutigkeit bei Datenaustausch mit Kodierung UTF-16

Ein BOMbon als Trost beim Byte-Reihenfolgen-Problem

- BOM (= *Byte Order Mark*) am Dateianfang, d.h. Byte-Reihenfolge-Markierung
- Trick: „Umnutzung“ des Unicode-Zeichens U+FEFF; Gegenstück mit Umkehrung der Byte-Folge in U+FFFE ist unzulässig
- nötig für UTF-16 und UTF-32, ergibt neue Kodierungen UTF-16LE vs. UTF-16BE usw.
- für andere Unicode-Kodierungen entbehrlich, trotzdem z.T. genutzt

ï»¿ oder: am liebsten UTF-8

- UTF-8: gängigste Unicode-Kodierung, auf 8 bit basierend
- unabhängig von Byte-Reihenfolge; BOM überflüssig für UTF-8, trotzdem unter Windows oft benutzt bzw. erwartet, unter Linux/Unix aber oft problematisch
- U+FEFF als BOM wird in UTF-8 zu Bytes EF BB BF, als ISO 8859-1 gelesen ergeben diese: ï»¿

Mehr über UTF-8

- sparsam: für ASCII-Zeichen nur 1 Byte nötig
- Nachteil: kompliziertes Verfahren für Umsetzung in Bytes statt nur Codepoint-Wiedergabe; je nach Bedarf 1 bis 4 Byte
- Zeichen aus ASCII-Bereich exakt wie ASCII kodiert, aber Zeichen 128-255 als jeweils 2 Bytes, inkompatibel zur Kodierung ISO 8859-1
- Standardkodierung für XML

Exkurs: Unicode und Schriften (1)

- Prinzip jeder Schrift: Abbildung eines Zahlenwerts (indirekt: eines Zeichens) auf einen Glyph
- für Unicode: Abbildung von Unicode-Codepoints auf geeignete Glyphen
- „pan-Unicode-Schriften“ für Abdeckung großer Bereiche eher selten (gängige Schriftformate: max. 65.535 Glyphen), sondern meist Auswahl von Unicode-Zeichen in der Schrift

Exkurs: Unicode und Schriften (2)

- viele Schriften mit Unicode-Zuordnung verfügbar, z.T. auch kostenlos
- keine Schrift hat alle Unicode-Zeichen
- z.B. Arial Unicode MS: 50.000 Glyphen; Schriften für Chinesisch z.T. 150.000 Glyphen
- systeminterne Technik der Schriftersetzung bei nicht verfügbaren Zeichen (analog zur Schriftfamilien-basierten Ersetzung)

Zwei Herausforderungen: Unicode kodieren und dekodieren

Erfolgreicher Unicode-Einsatz

- im Grunde: Erwerb von spezifischen Fähigkeiten, analog zu ABC-Schützen
  - Kodierungen *erkennen*:
    vor allem Datenfehlerkorrektur
  - Kodierungen *lesen*:
    vor allem korrekte Verwertung und Anzeige
  - Kodierungen *schreiben*:
    vor allem Eingabe und Konvertierung

(Fehl-) Kodierungen erkennen

- mit etwas Erfahrung gut machbar
- hilfreiche Infos zur Erkennung:
  - inhaltlicher und technischer Dateikontext
  - erwartete vs. erhaltene Zeichen / Bytes
  - erwartete vs. wirkliche Kodierung
  - benutzte Schriften
  - ggf. Tastatureingaben und problemauslösende Schritte

Kommissar Dekodierer klärt Kodierungsverbrechen auf

- oft detektivische Kleinarbeit zur Auflösung rätselhafter Kodierungsprobleme, anhand von Indizien; Beweismittel oft nicht vertrauenswürdig; viele falsche Fährten
- hinter jeder Anwendung lauert das Böse in Form von gewissenlosen Kodierungsverbrechern...

  (inspiriert von Tex Texin: Präsentation *Encoding Crime Scene Investigator*)

Detektivarbeit an Zeichenspuren

- Abgleich Erwartung vs. Ergebnis; Rückgriff auf übliche Motive und Erklärungsansätze
- Welche Zeichen sind vom Problem betroffen, etwa alle nicht-ASCII-Zeichen oder nur manche? Stattdessen anderes Zeichen, oder Ersatzdarstellung mit Fragezeichen bzw. Kästchen?
- Vorgehen: Annahmen über Ursache der Konvertierungsfehler machen und gegen die Daten validieren

Vorlesung: Einführung in die Zeichenpathologie

- Basis: Grundwissen über Unicode und Zeichensätze (sowie kaum Ekel vor Bytes...)
- Methodik: in den Daten typische Muster bzw. typische Ergebnisse von Fehlkonvertierungen erkennen, und Korrekturen erarbeiten
- Werkzeug: Editoren und Online-Dienste, die Unicode-Zeichen (bzw. andere Zeichensätze) korrekt interpretieren

Zeichenpathologie, Übung 1

- Welcher Kodierungsfehler liegt hier jeweils vor, z.B. bei Anzeige einer Webseite?

A) ÃœBen macht MÃ¼he

B) Ã�≈�ben macht MÃ�¬¼he

C) �ben macht M�he

D) Üben macht Mühe ? so wie Kühe

Lösungen zur 1. Übung

- A) Ãœben macht MÃ¼he
  → *UTF-8 als ISO 8859-1 gelesen*

  B) Ã�≈�ben macht MÃ�¬¼he
  → *UTF-8 doppelt angewendet*

  C) ben macht M he
  → *ISO 8859-1 als UTF-8 gelesen*

  D) Üben macht Mühe so wie Kühe
  → — *in Windows-1252 als ISO 8859-1*

Zeichenpathologie, Übung 2

- Für Fortgeschrittene: Welcher Kodierungsfehler liegt hier jeweils vor?

  E) \ben macht M|he

  F) =dcben macht M=fche

  G) %DCben%20macht%20M%FChe

  H) %C3%9Cben%20macht%20M%C3%BChe

Lösungen zur 2. Übung

- E) \ben macht M|he
  → *ISO 8859-1 mit gelöschtem 8. bit*

  F) =dcben macht M=fche
  → *quoted-printable, z.B. in Mails*

  G) %DCben%20macht%20M%FChe
  → *URL-Kodierung für ISO 8859-1*

  H) %C3%9Cben%20macht%20M%C3%BChe
  → *URL-Kodierung für UTF-8*

Analyse von Konvertierungsfehlern

- Typen von Konvertierungsproblemen:
  - Konvertierung fehlend (Annahme einer anderen Kodierung)
  - Konvertierung defekt (z.B. bei Mehr-Byte-Zeichen)
  - Konvertierung nicht zielführend
  - Konvertierung verdoppelt (z.B. bei UTF-8)
  - Konvertierung in verkehrter Richtung
  - unbeabsichtigte Konvertierung (z.B. 7 bit)

Herausforderung: Konvertierung zwischen Kodierungen

- Beziehungen zwischen Kodierungen beachten
  - z.B. ISO 8859-1 vs. Windows-1252
  - Byte-Repräsentation vs. Codepoint
  - auch Unicode entwickelt sich weiter
- Achtung bei irreversibler Konvertierung; z.B. von Unicode in ASCII-Umlaute (*ä* > *ae*, 7 bit), sonst Resultate wie:

  Autölektrikerin Zö liebt Pastäßen am Feür

Helfer im Alltag:
Praktische Tipps zu Unicode

Unicode lesen und schreiben

- Hilfsmittel für den Alltag mit Unicode in 3 Fragenkomplexen:
  - Was ist das für ein Unicode-Zeichen?
  - Wo steckt dieses Zeichen in Unicode?
  - Wie gebe ich das Unicode-Zeichen ein?
- für Anwender und für Entwickler gleichermaßen

Was ist das für ein Unicode-Zeichen?

- Wie man Codepoint und Name eines Zeichens auf dem Bildschirm herausfindet.
  Wichtigste Methoden:
  - direkte Übergabe über die Zwischenablage
  - visueller Abgleich anhand der Glyph-Gestalt
  - direkte Übergabe der UTF8-Repräsentation
- Achtung bei nicht unicodefähiger Anwendung!
  - UltraEdit vorher konfigurieren, siehe Bibliographie

Zwischenablage-Inhalt analysieren

- String analyzer von Richard Ishida
  - Einfügen in das orange eingerahmte Feld rechts
  - Ausgabe: Codepoint und Zeichenbeschreibung
- Unicode Reference von ZVON.org
  - Einfügen in das Eingabefeld
- UTF-8-Analysator der Webmasterei
  - Einfügen in das Eingabefeld
  - Anzeige:
    - „UC“ = dezimaler Codepoint
    - „UTF8“ = hexadezimale Byte-Folge in UTF-8
    - „Beschreibung“ = Unicode-Name
    - („Westeuropa“: Bytes gelesen als Windows-1252)

Visueller Abgleich der Glyphe (1)

- Frage: Welches Unicode-Zeichen sieht so aus wie meines? (1)
  - Elegant: decodeunicode. *Flash-gestützte Navigation durch alle Unicode-Blöcke sowie Suche nach Bestandteilen des englischen Namens oder nach hexadezimalem Wert.*
  - Umfassend: unicode.org-Zeichentabellen. Amtliche *PDF-Dateien mit Zeichendarstellungen, aber hier nur zugänglich über die Liste der (englisch benannten) Unicode-Bereiche. Etwas besser erreichbar über die Gruppierung der* Unicode-Blöcke nach typischem Zeichen.

Visueller Abgleich der Glyphe (2)

- Frage: Welches Unicode-Zeichen sieht so aus wie meines? (2)
  - Ohne Internetzugang (aber mit evtl. unsicherem Ergebnis): *Windows-Zeichentabelle* mit Zeichensatz „Unicode“ konsultieren, dann geeignete Unicode-Unterbereiche durchsehen. (Beschreibung sollte zum Zeichen passen)
  - In Word ab Word 2002: Über *Einfügen* | *Symbol* kann man das Zeichen in den Tabellen zu finden versuchen. Name und Codepoint des Unicode-Zeichens werden für das ausgewählte Zeichen im unteren Bereich des Dialogfelds eingeblendet.

Analyse von Bytes in UTF8

- Frage: Welches Unicode-Zeichen stellt diese Bytefolge in UTF8 dar?
  - decodeunicode, per *Advanced Search* (Zugriff über Klick auf Symbol „A über Lupe“):

    Eingabe der hexadezimal geschriebenen Bytewerte der UTF8-Repräsentation des Zeichens (ohne Leerzeichen) im Feld *UTF-8 Value*.

Wo steckt dieses Zeichen in Unicode?

- Wie man anhand einzelner Infos zu einem Unicode-Zeichen den Eintrag zum Zeichen mitsamt aller Infos findet.

  Wichtigste Ausgangspunkte:

  - Voller Name oder mindestens ein Teil des Namens bekannt
  - Nur Codepoint bekannt
  - Nur ungefähre Funktion bekannt

Zeichen über den Namen finden (1)

- Möglichkeiten:
  - Komfortabel: Bei decodeunicode in der *Advanced Search* (Zugriff über Klick auf Symbol „A über Lupe“) Name oder Teile davon bei *Name (Unicode name)* eingeben; danach ggf. Auswahl aus Liste
  - Akkurat, falls exakter Name bekannt: Im Unicode-Namensindex (alphabetisch) zum Eintrag des Zeichens gehen.
  - Mächtig und praktisch, aber Suche nicht auf Zeichenbeschreibungen begrenzbar: Suche in Unicode Resources von Alan Wood

Zeichen über den Namen finden (2)

- Weitere Möglichkeiten:
  - Spartanisch, aber schnell: in der Textdatei (1 MB!) Names List auf unicode.org mit der Suchfunktion des Browsers nach dem Teil des Namens oder der Beschreibung suchen.
  - Elegant, mit Folgesuche in Suchtreffern: in der Unicode Reference von ZVON.org im Suchfeld erst ein Fragezeichen eingeben und direkt danach das bereits bekannte Wort aus dem englischen Namen des Zeichens. (Manchmal fehlen Treffer!)
    Z.B. findet „?greek“ alle griechischen Zeichen.

Zeichen-Infos zum Codepoint

- Wie finde ich die Infos zum Unicode-Zeichen über seinen Codepoint?
  - decodeunicode: dezimalen/hexadezimalen Wert eingeben; ggf. Auswahl aus der Trefferliste
  - Unicode Reference von ZVON.org: entweder den dezimalen Wert hinter „#“ oder den hexadezimalen Wert hinter „#x“ eingeben
  - unicode.org-Zeichentabellen: hexadezimalen Wert ins Suchfeld; im Suchergebnis Link zur PDF-Datei, darin nach dem Zeichen (hex-Wert) schauen (1. Seite: Übersicht, danach: Beschreibungen)

Unicode-Zeichen thematisch finden

- d.h. Suche nach Unicode-Blöcken (*Ranges*)
  - Amtlich: In unicode.org-Zeichentabellen passende Unicode-Blöcke suchen, dann PDF-Dateien dazu prüfen. Glyphen immer angezeigt, da Grafiken.
  - Effizient und ungeschliffen: Test pages *for Unicode character ranges* von Alan Wood. Glyphen nur als Zeichen, also z.T. in Ersatzdarstellung.
  - Ästhetisch: decodeunicode, *Advanced Search*: interessanten Unicode-Bereich unter *Block* auswählen, danach ggf. Auswahl aus Trefferliste. Glyphen immer angezeigt, da Grafiken.

Codepoint dezimal ↔ hexadezimal

- Umrechnung des Codepoint-Werts zwischen dezimal und hexadezimal und umgekehrt
  - mithilfe des Windows-Taschenrechners
  - bei decodeunicode über die *Advanced Search* (Zugriff über Klick auf Symbol „A über Lupe“):
    - Dezimalen Wert: eingeben in Feld *DECIMAL VALUE*
    - Hexadezimalen Wert: eingeben in Feld *CODE POINT VALUE* (bewirkt String-Suche in hex. Codepoints)

      Ergebnis: Zeichenanzeige mit Codepoint als Hex-Wert; Dezimalwert unter Link *PROPERTIES*

Eingabe von Unicode-Zeichen

- Wie gebe ich das Zeichen ein, wenn ich seinen Codepoint kenne?
  - in Word ab Word 2002 (oder WordPad ab 2000): hexadez. Codepoint einfach eintippen, dann Alt-X (falls wirkungslos, z.B. Word 2007: alternativ Alt-C)
  - Zeichen irgendwo erzeugen (z.B. Internet), dann in Zwischenablage kopieren und am gewünschten Ort einfügen
  - mit eigenständigen Programmen zur Zeicheneingabe unter Windows

Unicode-Zeichen aus dem Internet

- viele Dienste im Internet, unter anderem:
  - allgemein: über Unicode Reference von ZVON.org Zeichen als kopierbaren Text rechts oben auf der ausgegebenen HTML-Seite erzeugen, nachdem der dezimale Wert (hinter „#“) bzw. der hexadezimale Wert (hinter „#x“) eingegeben wurde.
  - Unicode Character Pickers von Richida Ishida
  - speziell für lateinische Buchstaben mit Diakritika: latin character picker von Richida Ishida

Unicode-Zeichen lokal erzeugen

- Zeichen unter Windows offline erzeugen:
  - aus dem Fenster einer Unicode-fähigen Anwendung, wenn das Zeichen in Unicode-Schrift angezeigt wird; z.B. „Editor“/„Notepad“ oder Word (aber *nicht* in Symbol-Schrift usw.)
  - in Windows-Zeichentabelle: Wert *Unicode* im Feld *Zeichensatz* nutzen. Zeichen in den Unicode-Unterbereiche suchen, dann Kopieren über Zwischenablage, oder Direkteingabe über die rechts unten angezeigte Tastenkombination

Eigene Zeicheneingabe-Programme

- Bei häufiger Eingabe von Unicode-Zeichen: eines der (meist kostenlosen) Programme zur Zeicheneingabe unter Windows nutzen.
  - Näheres dazu beschreibt M. Müller-Hillebrand in seinem Beitrag Unicode-Zeichen eingeben, in dem speziell auch auf die Eingabe von Unicode-Zeichen in FrameMaker eingegangen wird.
- für ostasiatische Schriften sind *Input Method Editors* ohnehin unverzichtbar

Hilfe zur Zeichen-Konvertierung

- Wie kann ich meine Daten aus/nach Unicode konvertieren?
  - z.T. entbehrlich, wenn dynamische Konvertierung verfügbar, z.B. in Texteditor, Browser oder Mailer (z.B. Firefox: *Ansicht* | *Zeichenkodierung*)
  - für Anwender: z.B. Unicode Code Converter von R. Ishida (multiple Konvertierung von Unicode-Text in HTML, Bytewerte, Entities, Javascript u.v.a.m.)
  - für Programmierer: Bibliotheken wie iconv, ICU; Programm recode; XML-Technologie nutzen

Bibliographie und Links (1)

- Allgemein zu Unicode (1)
  - Wikipedia-de: Unicode
    http://de.wikipedia.org/wiki/Unicode
  - Unicode Consortium *(zentrale Anlaufstelle für Infos zu Unicode)*
    http://www.unicode.org/
  - Jens Meyer: Über Unicode. Lexitron
    http://www.lexitron.de/main.php?detail=true&eintrag=119

Bibliographie und Links (2)

- Allgemein zu Unicode (2)
  - W3C: Einführung in Zeichensätze und -kodes
    http://www.w3.org/International/getting-started/characters
  - W3C: Einführung in den Umgang mit Zeichenkodierungen in HTML und CSS
    http://www.w3.org/International/tutorials/tutorial-char-enc/
  - SELFHTML: Internationalisierung *(mit gut lesbaren Einführungen zu Zeichenkodierungsthemen)*
    http://de.selfhtml.org/inter/index.htm
  - Alan Wood's Unicode Resources
    http://www.alanwood.net/unicode/index.html

Bibliographie und Links (3)

- Zeichenübersichten mit Glyphen als Grafik:
  - amtliche Code Charts auf unicode.org (PDFs)
    http://www.unicode.org/charts/
  - Zugang zu Code Charts über typisches Zeichen
    http://www.unicode.org/charts/lastresort.html
  - decodeunicode
    http://www.decodeunicode.org/de/
  - Unicode Reference auf ZVON.org
    http://zvon.org/comp/s/unicode_reference/index.html

Bibliographie und Links (4)

- Zeichensuche über Namensbestandteile:
  - alphabetischer Unicode-Namensindex
    http://www.unicode.org/charts/charindex.html
  - Unicode Character Names List (Textdatei)
    http://unicode.org/Public/UNIDATA/NamesList.txt
  - Suche in Unicode Resources von Alan Wood
    http://www.alanwood.net/unicode/search.html
  - decodeunicodeorg
    http://www.decodeunicode.org/de/

Bibliographie und Links (5)

- Unicode-Hilfsmittel von Richard Ishida:
  - latin character picker
    http://people.w3.org/rishida/scripts/pickers/latin/
  - Unicode character pickers von Richard Ishida
    http://people.w3.org/rishida/scripts/pickers/
  - String Analyzer
    http://people.w3.org/rishida/tools/analysestring/
  - Unicode Code Converter
    http://people.w3.org/rishida/tools/conversion/

Bibliographie und Links (6)

- speziell zur Analyse von UTF8-Bytes:
  - UTF-8-Analysator der Webmasterei
    http://tools.webmasterei.com/utf8analysator/
- Verfahren/Programme zur Zeicheneingabe:
  - M. Müller-Hillebrand: Unicode-Zeichen eingeben
    http://cap-studio.de/wp/index.php/info/framemaker/unicode-zeichen-eingeben/

Bibliographie und Links (7)

- Test der Glyph-Anzeige im Browser (je nach Schrift):
  - Alan Wood's Test pages for Unicode character ranges
    http://www.alanwood.net/unicode/#links
  - Titus (Titus Is Testing Unicode Scriptmanagement)
    http://titus.fkidg1.uni-frankfurt.de/unicode/unitestx.htm
  - Nick Nicholas' Unicode Resources [v.a. Griechisch]
    http://www.tlg.uci.edu/~opoudjis/unicode/index.html

Bibliographie und Links (8)

- speziell für Redakteure und Setzer (1):
  - Wikipedia: Satzzeichen [mit Angabe des Unicode-Codepoints zu den einzelnen Zeichen]
    http://de.wikipedia.org/wiki/Satzzeichen
  - Leerzeichen in Unicode. *Eintrag im OpenOffice-Wiki*
    http://www.ooowiki.de/LeerZeichen
  - Rettet den Apostroph!
    http://einklich.net/etc/apostroph.htm
  - Rainer Seitel: Unicode 5.2, Zeichen und Symbole auf deutsch
    http://rainer-seitel.onlinehome.de/unicode-de.html

Bibliographie und Links (9)

- Schriften und Unicode:
  - Wikipedia-en: Unicode typeface *(zur Verfügbarkeit von Unicode-Zeichen in einzelnen Schriften)*
    http://en.wikipedia.org/wiki/Unicode_typeface
  - Yannis Haralambous: Fonts & Encodings, O'Reilly 2007, ISBN 978-0-596-10242-5.
- Unicode-Fähigkeit von Programmen:
  - UltraEdit für Unicode konfigurieren
    http://www.ultraedit.com/support/tutorials_power_tips/ultraedit/unicode.html

Bibliographie und Links (10)

- speziell für Programmierer (1):
  - Joel Spolsky: The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets (No Excuses!). *Joel on Software, 8.10.2003*. http://www.joelonsoftware.com/articles/Unicode.html
  - Richard Gillam: Unicode Demystified. A Practical Programmer's Guide to the Encoding Standard, Addison-Wesley 2002. ISBN 978-0-201-70052-7.
  - Jukka K. Korpela: Unicode Explained, O'Reilly 2006. ISBN 978-0-596-10121-3.

Bibliographie und Links (11)

- speziell für Programmierer (2):
  - Tex Texin: How to be a CSI (encoding Crime Scene Investigator)
    http://www.i18nguy.com/How-To-Be-A-CSI.pdf
  - Byte-Order-Mark FAQ
    http://www.unicode.org/faq/utf_bom.html
  - Markus Kuhn: UTF-8 and Unicode FAQ for Unix/Linux
    http://www.cl.cam.ac.uk/~mgk25/unicode.html

VIELEN DANK FUER IHRE

AUFMERKSAMKEIT !