TUSTEP und Unicode

Warum Unicode?

Anders als Menschen können Computer nicht wirklich Zeichen verstehen und verwalten. Für sie ist jedes Zeichen nur eine arbiträre Bitfolge, der willkürlich eine bestimmte Semantik hinterlegt wird.

Als Speicherpreise noch von zentralem Interesse waren, hat man sich bemüht, die pro Zeichen benötigte Bitfolge zu minimieren. Es gab Versuche mit 5-Bit-Code (Telex), dann auch mit 6- und 7-Bit-Codes (so z. B. in der ISO/IEC 646, deren bekannteste Anwendung ASCII ist). Die meisten der heute üblichen Verfahren gehen von einer 8-Bitfolge (= Oktet oder Byte) aus, was für 2^8 = 256 Zeichen ausreicht. Traditionell wird die Zuordnung Bitfolge ↔ Zeichen in einer 16x16-Matrix wiedergegeben.

256 Positionen, von denen aus technischen Gründen obendrein nicht alle nutzbar sind, reichen selbst in dem Teil Europas, der die lateinische Schrift benutzt, nicht für alle Buchstaben und Sonderzeichen aus, von anderen Schriften ganz zu schweigen. Als Antwort darauf haben sich viele verschiedene, einander widersprechende Konventionen für die Zuordnung von Buchstaben zu Bitfolgen herausgebildet. So wird das *ä* z. B. allein in Westeuropa in sehr verbreiteten Konventionen durch drei verschiedene Werte repräsentiert: 0xE4 (»Latin-1«), 0x84 (DOS-Codepages) und 0x8A (Apple Macintosh). Sehr viel mehr Kodierungen sind zu berücksichtigen, wenn man auch seltenere Konventionen einbezieht.

Solange man Daten primär nur auf einem Rechner oder in einem eng umgrenzten Kulturkreis einsetzt, wirkt sich das nicht aus. Daten*austausch* und langfristige Daten*haltung* sind nach diesem Modell allerdings praktisch unmöglich, falls man nicht die auf dem Ausgangsgerät verwendeten Konventionen kennt und über geeignete Konvertierungswerkzeuge wie z. B. das `#umwandle`-Kommando in TUSTEP verfügt. In einer Zeit, in der Daten intensiv über das Internet ausgetauscht und verstärkt als wiederverwertbare Resourcen eingesetzt werden, wird das Problem täglich drängender.

Das *Universal Character Set (UCS)*

Wenn eine 8-Bit-Folge nicht ausreicht, alle benötigten Zeichen zu kodieren, muss man entsprechend längere Sequenzen verwenden. Es wurden zwei konkurrierende Entwürfe erarbeitet, einer von der internationalen Standardisierungsorganisation ISO und ein weiterer vom Unicode Consortium, das im wesentlichen große amerikanische Softwarefirmen repräsentiert. 1993 wurden die Zeichenbelegungen der beiden Standards ISO/IEC 10646-1 und Unicode™ harmonisiert; sie werden seither in enger Kooperation erweitert. Verbleibende konzeptionelle Differenzen haben für den Endanwender praktisch keine Konsequenzen.

Letztendlich hat man sich auf eine zwei-Oktet-Folge geeinigt, mit der man 2^{16} = 65536 Zeichen repräsentieren kann, was für die üblichen Schriftzeichen der momentan verwendeten Staatssprachen reicht. Über sogenannte *surrogate pairs* kann man weitere 1024^2 = 1.408.576 Zeichen ansprechen, mit deren Hilfe man hofft, alle Schriftzeichen, die es je gegeben hat, erfassen zu können. In einer weiteren Ausbaustufe (»level-3-Implementierung«) erlauben es *combining diacritics* (fliegende Akzente), Diakritika mit (fast) beliebigen Grundbuchstaben zu kombinieren. Diese für viele wissenschaftliche Disziplinen essentielle Eigenschaft wird allerdings noch von fast keiner Software außer TUSTEP unterstützt.

Unicode läuft bereits jetzt in vielen Systemen, so z. B. in Windows NT, Aix und Java. Es setzt sich verstärkt auch als Transfermedium für Daten im Internet durch. Sowohl HTML 4 als auch XML rekurrieren auf Unicode als Grundzeichensatz, oft allerdings in einem speziellen normierten Transferformat UTF-8 (UCS Transformation Format, 8 bit form), das eineindeutig in 16 Bit abbildbar ist. Alle üblichen Browser unterstützen in ihren aktuellen Versionen UTF-8. Die korrekte Darstellung der Resultate hängt dann natürlich davon ab, ob der Endnutzer geeignete Unicode-Fonts installiert hat. Insbesondere bei nichtlateinischen Schriften muss man obendrein oft deutliche Abstriche bei der typographischen Qualität hinnehmen.

Die Kodierung der modernen Schriften im UCS nähert sich inzwischen der Vollendung; die Bearbeitung der historischen Schriften beginnt momentan hingegen erst richtig. Leider mangelt es den zuständigen Bearbeitern in diesem Feld oftmals sehr an Expertise, so dass die so entstehenden Vorschläge, die z. B. auch die ägyptischen Hieroglyphen betreffen, mangelhaft sind. Auch im Eigeninteresse ist hier die intensive Mitarbeit der entsprechenden Fachwissenschaften sehr anstrebenswert. Bereits jetzt fungiert die Abteilung Literarische und

Dokumentarische Datenverarbeitung hier als Schnittstelle zwischen Wissenschaftlern und Normungsorganisationen. Nachfragen aller Art sind herzlich willkommen.

Unicode-Unterstützung in TUSTEP

Für eine führende Anwendung im Bereich multilingualer Textkorpora wie TUSTEP ist Unicodeunterstützung hochrelevant. In Anbetracht des in TUSTEP konsequent angewandten Prinzips des *Script tagging* und der Kodierung von Akzentbuchstaben mit fliegenden Akzenten ist der Schritt hin zum Datenaustausch mit Unicode kleiner, als es vielleicht erscheint. *Intern* erlaubt Unicode beliebige Speicherformate, solange sie nur eindeutig in das UCS abbildbar sind, wobei *script tagging* ein für ernsthaftes Arbeiten mit nicht-lateinischer Schrift auf einer deutschen oder amerikanischen Tastatur bequemes und effizientes Vorgehen ist.

Bei der Arbeit mit TUSTEP ergeben sich keine Veränderungen. Man verwendet weiterhin die übliche TUSTEP-Zeichenkodierung.

Zum Datenaustausch im Unicode-Format gibt es ab der TUSTEP-Version 2000, die im Oktober 1999 freigegeben wurde, zwei neue Spezifikationswerte in `#umwandle`:
`code=unicode` liest und schreibt direkte 16-Bit-Zeichen,
`code=utf8` wandelt die Daten aus dem und in das *UCS Transformation Format* um, das im WWW üblich ist.

So wird z. B. aus #r+Materialy#r- automatisch Материалы und aus #g+Politeía#g- automatisch Πολιτεία − und das sowohl im Druck als auch im Netz.

Nehmen wir an, wir wollten eine kleine UTF-8-kodierte Datei `sem.txt` folgenden Inhalts nach TUSTEP importieren:

```
Museum Graeco Latinum
Материалы семинара по Платона
Πολιτεία
```

Nach dem üblichen Anmelden müssen wir nur das Kommando

```
#umw, sem.txt, tu_sem, co=utf8,
lo=+
```

absetzen und erzeugen so die folgende Datei:

```
Museum Graeco Latinum
#r+Materialy seminara po Platona#r-
#g+Polite%/ia#g-
```

Diese Datei kann man nun wie gewohnt bearbeiten. Mit

```
#umw, tu_sem, sem.txt, co=utf8,
lo=+
```

kann man das Ergebnis dann wieder in das Austauschformat bringen.

Zur Präsentation im Web sollte man dann auch die entsprechende »Meta-Information« im Header der HTML-Datei mitliefern:

```
<meta http-equiv="Content-Type"
content="text/html;
charset=utf-8">
```

Ganz analog arbeitet man mit dem Code `unicode`, dessen Ergebnis man in verschiedene Windowsprogramme übertragen kann. Hierbei hat man allerdings das Problem der Byteorder zu beachten, die sich zwischen sog. *little-endian* und *big-endian* Architekturen unterscheidet. Insgesamt dürfte daher zumindest noch UTF-8 die sichere Variante sein.

TUSTEP fungiert darüber hinaus als vollwertige *rendering engine* für die unterstützten Schriften (neben Lateinisch auch Griechisch, Kyrillisch, Hebräisch, Arabisch, Syrisch, Koptisch und Devanagari). In vielen Schriften wie z. B. im Arabischen ist diese Aufgabe komplex, da das Aussehen der Buchstaben von der Umgebung abhängt, in der sie stehen.

Auch der TUSTEP-Viewer, der beim Arbeiten mit dem Editor mit der Anweisung `mw+` gestartet werden kann, arbeitet intern mit Unicode und Unicode-Fonts. Er erlaubt dem Benutzer, gleich bei der Eingabe zu überprüfen, ob die Transliteration und die sonstige Kodierung korrekt ist. Auf diese Weise kombiniert TUSTEP die Bequemlichkeit der Dateneingabe mit einer Tastatur, die mindestens den ASCII-Zeichenvorrat unterstützt, mit den Vorteilen des *Universal Character Set*.

Marc Wilhelm Küster
kuester@zdv.uni-tuebingen.de