Welche Kodierung hat eine Textdatei?

Wie kann man nun unterscheiden, welche Kodierung in einer Textdatei gespeichert ist? Interpretieren kann man die gespeicherten Zeichen offenbar auf viele verschiedene Arten. Als 8-Bit-Datei ist überhaupt vieles unklar, denn man muss immer dazu sagen, um welche Kodepage es sich handelt. Im Zweifel muss man mit den Codepagenummern experimentieren, bis man die richtige gefunden hat.

Um die Anzahl der Möglichkeiten einzuschränken gibt es bei den Codepages für Unicode-Daten die optionale Möglichkeit, den Text mit einer besonderen Byte-Kombination einzuleiten, damit ein lesendes Programm die Kodierung automatisch einstellen kann. Diese Bytekombination heißt Byte Order Mark (BOM). Es gilt:

```
2b 2f 76 38 Unicode UTF-7
2b 2f 76 39 Unicode UTF-7
2b 2f 76 2b Unicode UTF-7
2b 2f 76 2f Unicode UTF-7
ef bb bf    Unicode UTF-8
ff fe       Unicode UTF-16-LE (Windows)
fe ff       Unicode UTF-16-BE
ff fe 00 00 Unicode UTF-32
00 00 fe ff Unicode UTF-32-BE
```

http://de.wikipedia.org/wiki/Byte_Order_Mark

Diese BOMs sind nicht zwingend vorgeschrieben, werden aber häufig verwendet.

Es kommt vor, dass man bei Ausgabe einer Textdatei in einem Cmd-Fenster (DOS-Box), diese Byte-Order-Marks sieht, denn die Konsole in einem Cmd-Fenster interpretiert diese Zeichen nicht als Unicode und gibt sie byteweise aus.

Wenn daher aktuelle Textdokumente mit diesem BOM gespeichert sind, besteht Eindeutigkeit über die verwendete Kodierung. Stammt das Textdokument dagegen aus den Anfangszeiten von DOS oder Windows, muss man die zutreffende Codepage suchen, bevor man den Text in die Unicode-Welt konvertiert.

In Word äußert sich das so, dass bei Dateien ohne BOM zuerst einmal eine Fehlermeldung kommt und man sich danach für eine Codepage entscheiden kann. Öffnet man in Word eine Textdatei mit BOM, gibt es den Fehler nicht aber man wird zur Sicherheit dennoch auf die Copepade-Auswahl umgeleitet und die festgestellte Kodierung wird voreingestellt.

Nachteile des BOM

Es gibt wenigstens drei Probleme beim Umgang mit dem BOM (Byte Order Mark):

- Dateien, die keinen Text enthalten, sind nicht mehr leer, weil sie ja eine mehr-bytige Kennzeichnung haben.
- Dateien, die ASCII-Texte enthalten sind keine ASCII-Dateien mehr, weil das BOM vorangestellt ist. Daher kann es bei Programmen, die reines ASCII als Input erwarten, zu Schwierigkeiten kommen.
- Man kann Textdateien nicht mehr verketten, weil an der Nahtstellte das BOM der nächsten Datei eingefügt wird und daher die resultierende Datei fehlerhaft ist. Das Betriebssystem hat diese Neuerung der Kennzeichnung von Text-Dateien nicht mitgemacht. Wünschenswert wäre daher eine Option beim Kopierbefehl, der die Berücksichtigung des BOM erzwingt.

Die vier Speichermethoden des Notepad

Wir geben im Notepad folgenden Mustertext, ein vereinfachtes Alphabet, ein.

```
!"§123ABCabc
ÄÖÜßäöüß
[\]{|}~
```

Der Text enthält in drei Zeilen jeweils drei Zeichen aus einem Zeichenbereich. Es sind insgesamt 33 Zeichen. Das „große scharfe s“ kann mit Alt-Gr Shift ß eingegeben werden. Wenn das nicht funktioniert, die Windows-Zeichentabelle verwenden und das Zeichen 1e9e im Unicode-Feld eingeben und auf „Auswählen“ klicken.

Es gibt vier Möglichkeiten zu speichern: ANSI, Unicode, Unicode-BE und UTF-8.

ANSI (=Windows Codepage 1252 „Westeuropäisch“)

```
21 22 a7 31 32 33 41 42 43 61 62 63 0d 0a c4 d6   !"§123ABCabc..ÄÖ
dc 3f e4 f6 fc df 0d 0a 5b 5c 5d 7b 7c 7d 7e 0d   Ü?äöüß..[\]{|}~.
0a                                                .
```

Die Datei hat exakt gleich viele Bytes wie der ursprüngliche Text Zeichen hat. Klar, es handelt sich um eine 8-Bit-Kodierung. Jedes Zeichen wird durch ein Byte dargestellt. Was fehlt, ist das das Große Scharfe-S, weil dieses Zeichen nicht im Zeichenraum 0..255 enthalten ist. Daher steht an dieser Stelle ein Fragezeichen (3f).

Man sieht es dieser Datei nicht an, welche Codepage zur Anwendung kommt. Würde man daher diese Seite in einer DOS-Box anzeigen, sieht man, wegen der dort auf 850 geänderten Codepage folgendes:

```
!"°123ABCabc
-Í_?õ÷³■
[\]{|}~
```

Man sieht, dass man bei den 8-Bit-Codepages unbedingt wissen muss, welche Kodierung verwendet wurde, sonst gibt es Datensalat.

Um diese Mehrdeutigkeiten auszuschalten, wird bei den Kodierungen für den UniCode jeder Textdatei optional ein *Byte Order Mark* vorangestellt.

UTF-16 (UCS-2) Unicode LE

```
ff fe 21 00 22 00 a7 00 31 00 32 00 33 00 41 00   !"§123A
42 00 43 00 61 00 62 00 63 00 0d 00 0a 00 c4 00   BCabc..Ä
d6 00 dc 00 9e 1e e4 00 f6 00 fc 00 df 00 0d 00   ÖÜßäöüß.
0a 00 5b 00 5c 00 5d 00 7b 00 7c 00 7d 00 7e 00   .[\]{|}~
0d 00 0a 00                                       ..
```

Die UTF-16-Kodierung verwendet zwei Bytes pro Zeichen und speichert (in der in Windows verwendeten Variante *"Little Endian"*) das niederwertige Byte auf die niederwertige Adresse. Daher sollte die Datei 66 Zeichen lang sein. Sie ist aber 68 Zeichen lang. Den Grund sieht man in den beiden ersten Bytes **"ff fe"**, die eine Kennzeichnung für UTF-16 darstellen. Diese Kennzeichnung hat den Namen *Byte Order Mark* (BOM). Sie ist nicht verpflichtend, hilft aber beim Einlesen von Texten, die richtige Kodierung zu finden.

UTF-16 (UCS-2) Unicode BE

```
fe ff 00 21 00 22 00 a7 00 31 00 32 00 33 00 41   !"§123A
00 42 00 43 00 61 00 62 00 63 00 0d 00 0a 00 c4   BCabc..Ä
00 d6 00 dc 1e 9e 00 e4 00 f6 00 fc 00 df 00 0d   ÖÜßäöüß.
00 0a 00 5b 00 5c 00 5d 00 7b 00 7c 00 7d 00 7e   .[\]{|}~
00 0d 00 0a                                       ..
```

Die Variante *"Big Endian"* dreht die Speicherreihenfolge der Bytes in einem Zeichen und auch des BOM um, verhält sich ansonsten aber wie die Windows-Variante.

UTF-8

```
ef bb bf 21 22 c2 a7 31 32 33 41 42 43 61 62 63   !"§123ABC
0d 0a c3 84 c3 96 c3 9c e1 ba 9e c3 a4 c3 b6 c3   ..ÄÖÜßäöü
bc c3 9f 0d 0a 5b 5c 5d 7b 7c 7d 7e 0d 0a         ß..[\]{|}~..
```

Wer viel englischen oder deutschen Text bearbeitet, verliert in der UTF-16-Kodierung viel an Speicherplatz, weil die meisten Zeichen aus dem Zeichenraum der ersten 256 Zeichen stammen und die 16-Bit-Kodierung für alle diese Zeichen zusätzlich „00“ speichert. Die UTF-8-Kodierung spart viel von diesem Platz, allerdings zu dem Preis, dass die Zeichen verschieden viele Bytes beanspruchen. Unser Beispiel mit 33 Zeichen beansprucht in UTF-8 46 Bytes. Die Speicherreihenfolge ist immer dieselbe. Das BOM besteht aus den drei Bytes: **ef bb bf**.

Anmerkung: Bei einer Notepad-Datei kann man es sich aussuchen, wie der Text gespeichert werden soll. Man kann wählen: ANSI, Unicode, Unicode Big Endian und UTF-8.

Eigentlich ist aber nur die letzte Bezeichnung „UTF-8“ korrekt, denn „Unicode“ ist keine Kodierung. Unicode ist ein Alphabet. Die Kodierung sagt, wie die Buchstaben zu Bytes werden. Gemeint ist im Notepad mit „Unicode“ die Kodierung UTF-16, bzw. UTF-16-BE. Auch „ANSI“ ist sonderbar, denn es gibt keine Kodierung dieses Namens. Um herauszufinden, was damit gemeint ist, speichert man die Bytes 30H bis 255H und vergleicht mit den Zeichen der verschiedenen Kodepages. Demnach ist „ANSI“ die Windows Codepage 1252 „Westeuropäisch“.