Karakter kódolás

A **betűkarakterek** egységes kódjának létrehozása a műszaki kommunikáció nagy eredménye. Kezdetben az Amerikai Szabványhivatal egy **7 bites** kódot szabványosított **ASCII** (American Standard Code for Information Interchange) néven. Ez a kód az angol ABC 26 nagy és kisbetűjét, a számjegyeket, írásjeleket, és az un. vezérlő karaktereket tartalmaz. Ez utóbbiak az írás formátumának vezérlésére és az egyes alkalmazások vezérlésére szolgáltak. A betűkarakterek kódját később ki kellett bővíteni. Részben a különböző nyelvek ékezetes és egyéb karakterei, részben a matematikai szimbólumok, illetve speciális grafikus karakterek miatt. Ez a **8 bites** karakterkód az **ANSI** kód nevet kapta (American National Standards Institute). A korai szövegszerkesztők széles körben használták az **IBM Code Page 852** nevű kódot. Ez tartalmazza a magyar nyelv 18 ékezetes betűkarakterét is.

```
ISO-8859-1 (Latin-1) A nyugat európai ékezetes karaktereket kódolnak
ISO-8859-2 (Latin 2) Kelet európai készlet. A magyar nyelv speciális jeleit
is tartalmazza: ű,ő.
```

Az **ISO-10646** szabvány definiál egy univerzális karakter halmazt (**UCS** – Universal Character Set). Biztosítja, hogy nem lesz információvesztés, ha egy tetszőleges írásjelet átalakítunk **UCS**-re majd vissza az eredeti kódolására (www.unicode.org). UCS tartalmazza az összes ismert nyelv írásjeleit. Nem csak a ma használatosakat, hanem a történeti népek holt nyelveinek jeleit is. Továbbá az ismert matematikai, tudományos szimbólumokat is. A szabvány folyamatosan bővül. A szabványt 1993-ban publikálták először (ISO-10646-1). Eredetileg 31 bites kódolás. A **0x0000** és **0xFFFD** terjedő 16 bites tartományt Basic Multilingual Plane-nek (BMP-nek) nevezzük.

A szabvány **ISO 10646-2** változata **2001**-ben jelent meg. A BMP-n kívüli tartományt tartalmazza. 2003-ban a két halmazt egyesítették a ISO 10646-ban.

Az UCS (unicode) szabvány szerint a kódolt karakterek nem csak egy számmal, hanem névvel is rendelkeznek. Az UCS kód szabványos előtagja az **U+**. Például **U+0041** jelentése „Latin capital letter A”. Az **U+0000** és **U+007F** közötti tartomány megfelel az ASCII 7 bites változatának. Az **U+0080** és **U+00FF** tartomány a **Latin-1**-nek felel meg.

A különbség az **ISO** és a **Unicode** között az, hogy ameddig az ISO 10646 egy kódtáblázatot jelent, addig az Unicode ezen felül tartalmaz **tipográfiai** szabványokat is. Megjelenítési eljárásokat (Arab, Héber írásjelekhez), több irányú szövegek kezelését egy dokumentumon belül, valamint rendező és szöveg összehasonlító algoritmusokat.

UTF-8 kódolás

Az UTF kódolások lényege az, hogyan tudjuk a 32 bites unicode karakterek kódolását rövidíteni, hogy ne legyen 1 leütés 4 byte hosszú.

Az alábbi táblázatból és a későbbi magyarázatból megérthető, hogyan használható az UTF-8 kódolás:

Unicode	**UTF-8**
00000000 00000000 00000000 0xxxxxxx	0xxxxxxx
00000000 00000000 00000yyy yyxxxxxx	110yyyyy 10xxxxxx
00000000 00000000 xxxxxxxx xxxxxxxx	1110xxxx 10xxxxxx 10xxxxxx
00000000 000xxxxx xxxxxxxx xxxxxxxx	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

Unicode	**UTF-8**
000000xx xxxxxxxx xxxxxxxx xxxxxxxx	111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx
0xxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx	1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx

A 7 bites ASCII kódokat változtatás nélkül kódoljuk. Az UTF-8 kódolás első bájtjában annyi 1-es szerepel az első 0 előtt, ahány bájt fogja követni az elsőt.

A ’ó’ Unicode kódja a decimális **243**, azaz hexadecimális **0x00F3**, bináris `00000000 00000000 00000000 11110011`. A második szabályra húzható rá (mivel az első csak 7 bitet kódolhat), tehát UTF-8 kódja bináris `11000011 10110011`, vagyis egy decimális **195**, azaz hexadecimális **0xC3**, majd ezt követően egy decimális 179, azaz hexadecimális **0xB3** byte.

Az UTF-16 os kódolást a Windows operációs rendszer és a Microsoft Office csomag előszeretettel használja. Lényege nagyon durván leegyszerűsítve annyi, hogy 2 byte kódol minimum egy unicode karaktert, és jelezheti a két bit sorrendjét is az állomány elején.

Megjegyzés: Az UTF-16 kódolás előírja, hogy a byte-sorrendet jelezni kell egy úgynevezett byte-sorrend jelző (byte order mark, **BOM**) segítségével, amelynek meg kell előznie a tényleges szöveget. Notepad++-ban ez látszik is, egyes szöveges állományok elején. A BOM-ként használt karakter a „nulla szélességű nem törhető szóköz”, amely értelemszerűen sosem fordul elő eredeti jelentéstartalmával szöveg elején. Unicode száma hexadecimálisan **FEFF**; az **FE FF** byte-sorozat jelenti a „big-endian”, azaz „nagy végű”, és az **FF FE** sorozat jelenti a „little-endian”, azaz „kis végű” byte-sorrendet.

Base64 kódolás

Ez valójában nem karakter kódolás, de didaktikailag ide illik.

Multipurpose Internet Mail Extensions (**MIME**) az internet hivatalos levélformátuma. A levelek az **SMTP** (Simple Mail Transfer Protocol) protokoll segítségével továbbítódnak a címzetthez. Az **SMTP** protokoll csak 7 bites **ASCII** karakterek továbbítását támogatja. A **MIME** szabvány többféle módszert támogat a bináris adatok továbbításához. Az egyik legismertebb a Base-64 kódolás.

A következő példában egy melléklettel rendelkező email-t láthatunk, azaz a forrásadatot amit a levelező programok értelmeznek:

```
Content-type: multipart/mixed; boundary="frontier"
MIME-version: 1.0
--frontier
Content-type: text/plain
This is the body of the message.
--frontier
Content-type: application/octet-stream
Content-transfer-encoding: base64
gajwO4+n2Fy4FV3V7zD9awd7uG8/TITP/vIocxXnnf/5mjgQjcipBUL1b3uyLwAVtBLOP4nV
AhSzlZnyLAF8na0n7g6OSeej7EjlF/aglS6ghfju FgRr+OX==
--frontier--
```

Hátulról a második és harmadik sorban láthatóak base64 kódolt adatok. Nézzük hogyan jön ez létre.

Base64 kódolás (vagy általánosabban adat reprezentáció) egy 64 jelből álló készleten alapul. Olyan, mintha **64**-es számrendszerbe írnánk át az adatainkat. A kódolást 6 bites csoportokon végezzük.

```
0..25     – 'A' .. 'Z'
26..51    – 'a' – 'z'
52..61    – '0' – '9'
62        – '+'
63        – '/'
```

Azaz 0 és 25 között használjuk az angol ABC első 26 nagybetűjét, majd 26 és 51 között a kisbetűket, valamint 52-61-ig a számokat majd a maradék 2 kódhelyre a + és / jelek kerülnek. Így összesen 64 féle szimbólumot adtunk meg.

Példa

Kódoljuk a következő bináris adatot Base-64 kódolás szerint:

`001100110011`

Bontsuk fel két 6 bites részre:

`001100` , `110011`

Ezek decimálisan 12 és 51. A táblázat alapján: 12 = M és 51 = z

Az eredmény:

Mz

A Base-64 kódolás 3 byte-onként történik és 3 byte kódolt eredménye 4 karakter lesz. Hogyan kódolunk, ha a kódolandó bájtok száma nem osztható 3-al? (ilyen kor 1 vagy 2 db egyenlőséggel jelöljük ezeket az eseteket az alábbi példák alapján)

Kódoljuk a `00000001`-t (ami 1 byte):

1. Egészítsük ki 3 byte-ra `00000001 00000000 '00000000 2. Bontsuk fel 6 bites csoportokra: 000000 010000 000000 000000 , és nézzük ki a táblázatból a karaktereket. 3. Az eredmény AQ== Most pedig kódoljuk a 00000010 00000001-t (ami 2 byte adatot jelent): 1. Egészítsük ki 3 byte-ra 00000010 00000001 00000000 2. Bontsuk fel 6 bites csoportokra: 000000 100000 000100 000000 3. Az eredmény AgE=''`

Base-64 dekódolás: 4 karakterenként visszafelé majd 8 bitenként csoportosítva. Az egyenlőség jelek száma alapján el lehet dönteni, hogy az utolsó 6 vagy 12 bitet figyelembe kell-e venni.

A kódolás fontos előnye, hogy sortörő karaktert is tartalmazhat. (minden olyan karaktert is, ami a kódtáblázatból hiányzik). De a dekódolásnál ezt figyelembe kell venni.