Elméleti háttér

Karakterkódolásnak nevezzük azt, amikor betűk, számok és karakterek egy halmazához valamilyen kódot rendelünk. Ez a kód kontextustól függően lehetnek például ábrák[[1]](#footnote-1) vagy elektromos impulzusok (Morse-kód). Az informatikában természetes számokat használunk, mivel a számítógépek ezeket tudják a legegyszerűbben tárolni. Egy karakterkódolásnál használatos struktúrákat a legelterjedtebb karakterkódoláson, a Unicode szabványon keresztül szeretném ismertetni.[[2]](#footnote-2) Ez a modell egy másik modellt, az IAB modellt egészíti ki.[[3]](#footnote-3)

Az első kérdés, ami egy karakterkódolásnál felmerül, hogy melyik karaktereket szeretnénk lekódolni. Ezen karakterek halmazának a neve Abstract Character Repertoire (ACR). Bár a definíció nem követeli meg, mégis elterjedt, hogy a karakterek halmaza egyazon ábécé betűiből és az ábécéhez tartozó írásjelekből áll. Egy ACR lehet zárt vagy nyitott. A zárt repertoár megváltoztatására vagy kibővítésére nincs lehetőség, míg a nyitott repertoárhoz hozzá lehet adni újabb karaktereket. Az IAB modellben az ACR a karakterkészletnek (character set) felel meg.

Következő szint a Coded Character Set (CSS). A CSS egy leképzés (mapping), ahol az absztrakt karakterekhez nemnegatív egész számokat rendelünk. A számoknak nem kötelező egymást követőeknek lenniük. A karakterek és a számok között 1-1 kapcsolat lehet. Ha egy karakterhez hozzárendelünk egy számot, akkor az a karakter része lesz a kódolt karakterkészletnek (coded character set). Ekkor a karaktert kódolt karakternek (coded character), a számot pedig kódpontnak (code point) nevezhetjük. (ISO) A kódolt karakterkészletet másik elnevezése a karakterkódolás (character encoding), vagy az IAB modellből ismert kódlap (code page).

A Character Encoding Form (CEF) szintén egy leképezés, mely a CSS-ben használt kódpontokat rendeli kódegységek sorozatához (sequences of code unit). Egy kódegység egy egész számot jelent, mely egy meghatározott bináris szélességet foglal el egy számítógépes architektúrában, például egy 8-bites bájt. Vagyis a CEF azt határozza meg, hogy az egyes karakterek hány kódegységen lehetnek tárolva, és ez a karakterkódolás egyik legfontosabb aspektusa. Kódegységek sorozatának mérete alapján megkülönböztetünk fix szélességű és változó szélességű CEF-eket. Szemléltetésként, az UTF-8-at említeném meg, ahol a karakterek 1-4 bájton vannak reprezentálva.

Az utolsó szinten a Character Encoding Scheme (CES) található. A CES egy olyan reverzibilis hozzárendelés, melyben a kódegységekhez rendelünk bájtok sorozatát. A CES esetében már foglalkozni kell a számítógépek architektúrájával is, hogy biztosítani tudjuk cross-platfrom esetén is az adatok perzisztenciáját, például, hogy egy architektúra big-endian (UTF-16BE) vagy little-endian (UTF-16LE).

Egy önálló szintet is fel szoktak sorolni, amely a Unicode modelljén kívül helyezkedik el. A Character Maps (CM) a fentebb ismertetett szinteket fűzi össze egy transzformációvá. A CM absztrakt karaktereket alakít bájtok sorozatává, vagyis a CSS, CEF és CES műveleteket végzi el.

Az ACR szintjén lévő karakterek absztrakt karakterek, melyek hagyományok segítségével kialakult fogalmak. Az írásjelek (glyph) az absztrakt karakterekhez tartozó konkrét képek vagy alakzatok. Egyazon absztrakt karakterhez hasonló, de részletekben eltérő írásjelek is tartozhatnak. Ezek a különböző írásjelek hozzák létre a különböző betűtípusokat. Fontos megemlíteni, hogy az írásjelek nem egy-egy kapcsolatban vannak a karakterekkel. Ebből kifolyólag ligatúrák jöhetnek létre, melyek közül a leghíresebb az f-i ligatúra: .

|  |  |
| --- | --- |
| fi | fi |
| Az f és i karakterek ligatúra nélkül | Az f-i ligatúra |

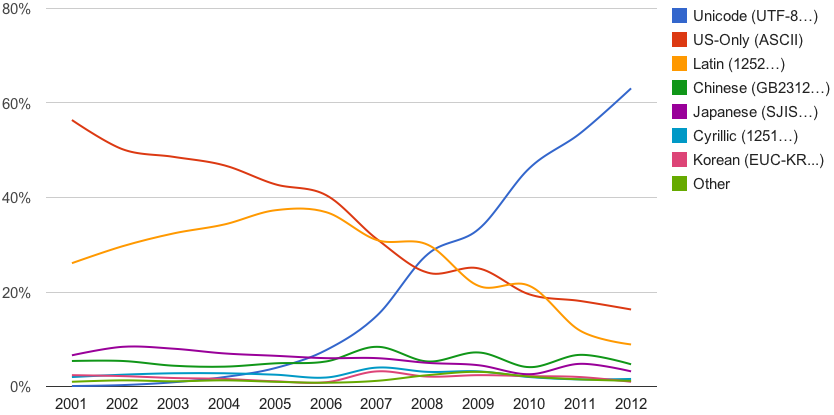
A karakterkódolások történetét tekintve, az egyik legelső kódolás az American Standard Code for Information Interchange, vagyis ASCII volt. Ez tartalmazta az angol ábécé kis és nagybetűit, számokat, írásjeleket. Azonban más nyelvekben lévő írásjeleket nem, ezért minden ország/nyelv/kultúra elkészítette a sajátját. Japánban több kódolás is elterjedt lett, ezek nem voltak egymással kompatibilisek. Ennek következtében a rosszul dekódolt karakterek semmilyen nyelvre nem hasonlító, olvashatatlan szöveget alkottak. Ez a jelenség annyira gyakori volt, hogy nevet is kapott: „mojibake”, szó szerint fordítva „karakter transzformáció”.[[4]](#footnote-4)

Ez a probléma a World Wide Web megjelenésével még nagyobb formát öltött. Egyértelművé vált, hogy szükséges egy világszerte egységes karakterkódolási szabvány, és ez a Unicode nevet kapta. Az Unicode célja, hogy globálisan elérhető összes karaktert és írásjelet összegyűjtse.

A Unicode több százezer karaktert és kódpontot párosít össze. Ezek tárolásához a kódolásoknak négy bitet kell használniuk karakterenként. Az UTF-32 pontosan így működik. Azonban a legtöbb esetben ez erősen pazarló tárolási módszer, hiszen a leggyakoribb karaktereket egyetlen bájtban is el lehet tárolni. Az UTF-8 valósítja meg ezt az elképzelést.[[5]](#footnote-5)

|  |  |  |
| --- | --- | --- |
| Karakter | Kódolás | bitek |
| A | UTF-8 | 01000001 |
| A | UTF-32 | 00000000 00000000 00000000 01000001 |
| あ | UTF-8 | 11100011 10000001 10000010 |
| あ | UTF-32 | 00000000 00000000 00110000 01000010 |

Az UTF-8 kódegysége változó hosszúságú, így csak annyi bájtot fog használni a karakterkódolására, amennyit az absztrakt karakter kódpontja megkövetel. A kódpontján kívül egyéb biteket is tárolni kell, mellyel azt jelezhetjük a dekódolónak, hogy több bájt tartozik ugyanazon karakterhez. Praktikussága miatt, az UTF-8 a világ egyik legelterjedtebb kódolása lett.[[6]](#footnote-6)



1. ábra

1. TODO kódolás technika könyv otthon [↑](#footnote-ref-1)
2. TODO http://www.unicode.org/reports/tr17/ [↑](#footnote-ref-2)
3. TODO http://www.ietf.org/rfc/rfc2130.txt [↑](#footnote-ref-3)
4. https://blog.prototypr.io/mojibake-the-unknown-very-common-problem-with-east-asian-language-input-804191067c18 [↑](#footnote-ref-4)
5. TODO http://kunststube.net/encoding/ [↑](#footnote-ref-5)
6. https://www.w3.org/International/questions/qa-who-uses-unicode [↑](#footnote-ref-6)