Az iconv értékelése

Dolgozatom zárásaként meg szerettem volna vizsgálni, hogy az általam készített iconv milyen mértékben hasonlít egy igazi iconv programhoz. Ennek érdekében több szempontból összehasonlítottam a GNU iconv implementációjával.[[1]](#footnote-1) A két programot helyesség és teljesítmény szerint külön-külön megvizsgáltam, és a kapott eredményeket összevetettem, hogy megtudhassam, mennyire működőképes és használható az én implementációm.

Először is a helyes működést akartam megvizsgálni. Hogy ne legyen túl bonyolult a mérés, csak magyar nyelvű szövegek kódolását vizsgáltam. Kiinduló kódolásnak az ISO-8859-2 karakterkódolási szabványt választottam.[[2]](#footnote-2) Ez a kódolás 8 biten tárol kódpontokat és ASCII alapú, vagyis a kódolás a 0x7F karakterig megegyezik az ASCII kódtáblával és csak az azutáni értékeknek rendel új karaktereket. Gyakorlatban ez azt jelenti, hogy a bájt első bitje nulla, akkor a karakter ASCII karakter, ha pedig a bit egyes, akkor valamilyen különlegesebb karakterről van szó. Az ISO-8859 szabványok, mind ilyen elven működnek, a különbség közöttük a különleges karakterekben vannak, ugyanis e karaktereket földrajzi területek szerint válogatták csoportokba. Az általam használt ISO-8859-2 a kelet európai országok karaktereit tartalmazza (például a magyar ű betű = 0xFB), de pl. az 5-ös sorszámú a görög ( ћ = 0xFB), míg a 11-es a thai karaktereket ( ๛ =0xFB) foglalja magába. Összesen 16 különböző ISO-8859 szabvány van. Az ISO-8859-2 másik használatos elnevezése a Latin-2.

A konverzió cél kódolásának az ASCII-t választottam. Ahhoz, hogy egy tipikus magyar szöveget ISO-8859-2 kódolásból ASCII-vé konvertáljunk, csupán az ékezetes magyar betűket kell leképezni, valamely ASCII karakterré, logikus megoldás a nem ékezetes karakterré leképezni. A konverzióhoz a következő programot készítettem el a DSL nyelvemben:

**source** iso88592 {

**alias** latin2

**target** ascii {

// kisbetűk

0xe1 ~ 0x61 // á

0xe9 ~ 0x65 // é

0xed ~ 0x69 // í

0xF3 ~ 0x6f // ó

0xF6 ~ 0x6f // ö

0xF5 ~ 0x6f // ő

0xFA ~ 0x75 // ú

0xFC ~ 0x75 // ü

0xFB ~ 0x75 // ű

// nagybetűk

0xC1 ~ 0x41 // Á

0xC9 ~ 0x45 // É

0xCD ~ 0x49 // Í

0xD3 ~ 0x4f // Ó

0xD6 ~ 0x4f // Ö

0xD5 ~ 0x4f // Ő

0xDA ~ 0x55 // Ú

0xDC ~ 0x55 // Ü

0xDB ~ 0x55 // Ű

}

}

A generált forráskódot hozzáadtam az iconv függvényeimhez és már készen is állt a használatra. Egy gyors teszttel meggyőződtem a helyességéről.

|  |  |
| --- | --- |
| Konvertálás előtt | Konvertálás után |
| abcdefxyz  ABCDEFXYZ  áéíóöőúüű  ÁÉÍÓÖŐÚÜŰ | abcdefxyz  ABCDEFXYZ  aeiooouuu  AEIOOOUUU |

Ahhoz, hogy össze tudjam hasonlítani az én iconv-m és a GNU-féle iconv programok végeredményeit, ki kellett egészítenem a programomat, hogy az fájlból olvasson be sztringet, azt konvertálja le az iconv segítségével, majd fájlba írja ki a konvertált buffert. A megírt C program forráskódja:

const char \*source = "iso88592";

const char \*target = "ascii";

iconv\_t cd = iconv\_open(target,source);

if(cd!=-1){

FILE \*file\_in=fopen("from.txt","r");

FILE \*file\_ou=fopen("to.txt","w");

Ennél a résznél arra kellett figyelmet fordítanom, hogy az iconv megváltoztatja a bufferek pointereinek helyzetét. Hogy ne kelljen folyton új buffereket foglalnom a memóriában, elmentem a bufferek pointereit, hogy később visszaállíthassam a bufferek elejére őket. Látható, hogy a fájlból százasával konvertálom át a karaktereket.

char \*inbuf=(char\*) malloc(sizeof(char[100]));

char \*inbuf2=inbuf;

char \*outbuf=(char\*) malloc(sizeof(char[100]));

char \*outbuf2=outbuf;

if(file\_ou==NULL) printf("file error 2");

else if(file\_in==NULL)printf("file error");

else while(fgets(inbuf,100,file\_in)!=NULL){

size\_t inbytesleft=strlen(inbuf)+1;

size\_t outbytesleft=sizeof(char[100]);

outbuf=outbuf2;

iconv(cd,&inbuf,&inbytesleft,&outbuf,&outbytesleft);

fprintf(file\_ou,"%s",outbuf2);

inbuf=inbuf2;

}

fclose(file\_in);

fclose(file\_ou);

}

iconv\_close(cd);

A fenti program segítségével, már megtörténhetett a tényleges összehasonlítás. Az eredmények összehasonlításhoz egy tipikus, átlagos magyar szöveget kellett használom és ez a szöveg Jókai Mór regényének, A kőszívű ember fiainak az első pár fejezete lett.[[3]](#footnote-3) A szövegre lefuttattam az én iconv függvényeimet, majd a GNU iconv függvényét is, az utóbbit parancssorból:

iconv -f ISO-8859-2 -t ASCII//TRANSLIT from.txt -o to\_iconv.txt

Ezután a diff program segítségével összehasonlítottam a két fájlt.

diff -s -q to.txt to\_iconv.txt

Sajnálatosan az két fájl nem egyezett meg. A különbségek megvizsgálásával észrevettem, hogy A kőszívű ember fiai mégsem tipikus magyar szöveg. Ugyanis néha német szavak fordulnak elő benne, melyekben tartalmazzák az ä betűt. Ennek a kijavítására a DSL programba beillesztettem pár bájtkonverziót. Feltűnt, hogy egy ilyen hibának a kijavítása, mennyire egyszerű modellezés és kódgenerálás használatával.

0xE4 ~ 0x61 // a betű két ponttal

0xC4 ~ 0x41 // A betű két ponttal

Ezután újra lefuttattam a fájlok konvertálását és azok összehasonlítását.

diff -s -q to.txt to\_iconv.txt

Files to.txt and to\_iconv.txt are identical

A két fájl megegyezett, vagyis az én iconv-m képes olyan eredményt adni, mint egy elterjedt iconv program.

Az iconv helyességének igazolása után a teljesítményét akartam megmérni. Ehhez a bemeneti fájl méretét kellett megnövelnem, majd megmérnem mindkét program futásidejét. Remélhetőleg nem lesz jelentősen lassabb az én implementációm.

A fájl növeléséhez egyszerűen többször újra bemásoltam a fejezeteket A kőszívű ember fiaiból, ezzel megsokszorozva a méretét. A bemeneti fájl kicsit több mint 3MB volt. Mindkét program pillanatok alatt átkonvertálta a fájlt, természetesen helyes eredménnyel. A megfelelő méréshez tovább kellett sokszorosítanom a fájl méretét. Ezúttal 21MB-nyi szöveget kellett átkonvertálni. A programoknak észrevehetően több idő kellett, nagyjából egy másodperc, de így is ugyanakkor végeztek a konvertálással. Az én iconv programom úgy tűnik fel tudja venni a versenyt más programmal.

Az utolsó méréshez jelentős méretre, 105MB-re növeltem a bemeneti fájlokat. Itt már tetten érhető volt a programok gyorsasága. A GNU iconv 1.073s alatt konvertálta át a fájlt, míg az én iconv-m 4.125s alatt.

|  |  |
| --- | --- |
| GNU iconv | iconv\_him |
| real 0m1.073s  user 0m0.953s  sys 0m0.109s | real 0m4.125s  user 0m3.313s  sys 0m0.281s |

Az eltérés jelentős, úgy sejtem a GNU iconv a gazdaságosabban használja a memóriát, jobban rendszerezi a bájtkonverziókat. Ennek ellenére, meg vagyok elégedve a programom teljesítményével, ugyanis jelentős fájlméret esetén sincs nagyságrendekkel lemaradva.

Legvégül a programban lévő kódsorok száma szerint hasonlítottam össze a két programot. Ehhez a cloc programot használtam, mely megszámolja a fájlban/mappában lévő kódok sorait.[[4]](#footnote-4) Ahogy sejteni lehetett, az én iconv-m mérete csupán töredéke a GNU iconv programnak. Az én iconv-m mérete:

Language files blank comment code

-------------------------------------------------------------------------------

C 4 120 198 466

C/C++ Header 3 19 3 54

-------------------------------------------------------------------------------

SUM: 7 139 201 520

-------------------------------------------------------------------------------

A GNU iconv mérete:

Language files blank comment code

--------------------------------------------------------------------------------

C/C++ Header 272 2874 13730 98188

Bourne Shell 27 10307 8935 54501

m4 109 1670 1061 18727

C 61 1456 2691 9087

PO File 36 1710 4507 6696

...

--------------------------------------------------------------------------------

SUM: 540 18773 31184 194626

--------------------------------------------------------------------------------

A kódsorok számánál nyilván az is közre játszik, hogy jelenleg az én iconv-m csupán egyféle konverzióra képes (ISO-8859-2-ből ASCII-t készít), míg a GNU iconv több kódolosából tudunk választani és többségében mindkét irányba működik a konverzió.

Összességében, az kódgenerátorom és a konvertálóm együttese nem a legjobb iconv implementációt nyújtják. Azonban a DSL nyelv könnyű használata, a generált kód egyszerű üzembe helyezése, a helyes végeredmények elkészítése és a relatíve alacsony futásidő mind hozzájárulnak ahhoz, hogy az elkészült alkalmazás rugalmas, egyszerű, jól működő és a gyakorlatban felhasználható legyen.

Konklúzió

Szakdolgozatom keretein belül az informatika több területét meg kellett ismernem és egy olyan alkalmazást készítettem, amelyben régi és új ismeretek is felhasználásra kerültek. Szakdolgozatomban kódgenerálással, karakterkódolásokkal és személyes kedvencemmel, modellezéssel is foglalkoztam. Az elkészült alkalmazás magába foglal egy DSL nyelvet, egy generátort és az iconv függvényeket. A nyelvben karakterkódolások közötti konverziókat írhatunk le. A kódgenerátorral képesek vagyunk feldolgozni és működőképes C kódot generálni egy DSL-beli programból. Az implementált iconv függvények pedig a generált C kódot használják fel.

A dolgozat elkészítésének a motivációja a modell alapú fejlesztés kipróbálása volt. Modellek segítségével gyorsabb és egyszerűbb volt a fejlesztés, ezek az előnyök a kódgenerálásnál jöttek elő leginkább. Dolgozatomban azt is vizsgáltam, hogy érdemes-e régebbi problémákat újra elővenni, és újabb, modernebb eszközökkel megoldani őket. Véleményem szerint van létjogosultsága mind a modell alapokon történő fejlesztésnek, mind a régi problémák ismételt megoldásának. Bár szerintem nem lenne túl jövedelmező vállalkozás.

Az elkészült alkalmazásban természetesen vannak továbbfejlesztési lehetőségek. Az iconv függvényeket és a generált kódot lehetne optimalizálni, hogy jobban használják az erőforrásokat. Azonban a DSL nyelvben nagyobb hiányosságok vannak. Egyes karakterkódolások állapotfüggőek, vagyis a karaktereket az is befolyásolja, hogy őket milyen egyéb karakterek előzték meg. Olyan kódolások is léteznek, amelyekben egy darab karakter által elfoglalt bájtok száma változik, például az UTF8-ban, 1-4 bájtot foglal el egy karakter. Jelenleg ilyen struktúrát nem lehet leírni a DSL nyelvben, és emiatt korlátozott a használata. Ezeket a problémákat lehetne orvosolni.

Személy szerint élveztem a dolgozat és az alkalmazás elkészítését. Sok hasznos tudást szereztem a modellezésről és e tudást későbbi tanulmányokba is fel tudom használni. A DSL nyelvek és a kódgenerálás nekem mindig is egy egzotikusabb területe volt az informatikának, melyre mindig is kíváncsi voltam. Örülök, hogy a karakterkódolás elméletét is megismertem végre, mivel ez a téma túl ismeretlen volt számomra és mindig is ódzkodtam tőle. Összefoglalva, nem bántam meg, hogy ezt a témát választottam dolgozatom témájának és büszke vagyok mindarra, amit dolgozatom keretein belül megismertem vagy elkészítettem.

1. TODO https://www.gnu.org/software/libiconv/ [↑](#footnote-ref-1)
2. TODO https://www.fileformat.info/info/charset/ISO-8859-2/list.html [↑](#footnote-ref-2)
3. TODO http://mek.oszk.hu/00600/00695/html/ [↑](#footnote-ref-3)
4. TODO http://cloc.sourceforge.net/ [↑](#footnote-ref-4)