Překódování textů v jazyku RPG

Vladimír Župka, 2021

Obsah

Obecně o kódování dat

V tomto článku se podíváme na možnosti překódování znakových dat. Kvůli rozsáhlosti problematiky se nebudeme zabývat všemi možnostmi, které nabízí samotný operační systém, i když jsou velmi zajímavé a užitečné. Soustředíme se jen na ty, které může využít programátor v jazyku RPG. Informace o kódování a překódování dat lze nalézt v publikacích

IBM i globalization na adrese

https://www.ibm.com/docs/en/i/7.4?topic=programming-i-globalization.

National Language Support - related APIs na adrese

https://www.ibm.com/docs/en/i/7.4?topic=ssw_ibm_i_74/apis/nls2.htm

Integrated File System APIs na adrese

https://www.ibm.com/docs/en/i/7.4?topic=ssw_ibm_i_74/apis/unix2.htm.

Architektura CDRA a čísla CCSID

V Souhrn i je zavedena architektura pro reprezentaci znakových dat CDRA – Character Data Representation Architecture. CDRA identifikuje znaky pomocí identifikátoru kódovacího schematu (encoding scheme identifier ESID), znakové sady (character set) a dalšími informacemi vztahujícími se ke kódování. Tyto pojmy nebudeme vysvětlovat, kdo by o ně měl zájem, může je nalézt v (nepříliš jednoduché) dokumentaci na adresách

https://www.ibm.com/downloads/cas/G01BQVRV

https://www.ibm.com/docs/en/i/7.4?topic=ssw_ibm_i_74/apis/nls4.htm

V Souhrn i je architektura CDRA reprezentována prostřednictvím čísel CCSID (Coded Character Set Identifier). Touto zkratkou se označuje číslo (z rozsahu od 0 do 65535), které reprezentuje speciální datovou strukturu uloženou v systému. Operační systém dovoluje zadat CCSID v různých svých částech:

- některé CL příkazy,
- databáze,
- kopírování souborů (databáze, IFS),
- zařízení (obrazovky, tiskárny),
- řízení práce (work management),
- systémové funkce API

Vývoj kódování v IBM i

Postupem doby se v IBM i vyvinulo několik způsobů kódování znakových dat:

- SBCS – Single Byte Character Set. Každý znak je zakódován v jednom bajtu. Kódovací schéma je EBCDIC (Extended Binary Coded Decimal Interchange Code), popř. ASCII (American Standard Code for Information Interchange). Podle jazyka odpovídá znakové sadě určité číslo CCSID.
- DBCS – Double Byte Character Set. Každý znak je zakódován ve dvou bajtech. Jde o původní, starší způsob kódování asijských znaků.
- UCS-2 – Universal Character Set 2. Každý znak je zakódován ve dvou bajtech. Jde o univerzální kódování všech světových jazyků podle normy ISO/IEC 10646, které se stalo základem kódování Unicode UTF-16.
- Unicode UTF-16 – Znak je zakódován ve dvou, někdy ve čtyřech bajtech.

- Unicode UTF-8 – Znak je zakódován v jednom až čtyřech bajtech. Znaky ASCII a ISO 8859-1 (Latin 1 - západní latinka) jsou zakódovány v jednom bajtu, znaky s diakritickými znaménky ostatních latinských abeced jsou kódovány ve dvou bajtech, znaky ostatních jazyků ve třech až čtyřech bajtech. Výhodou je, že většina latinkových textů má jednobajtové znaky. Nevýhodou je, že texty mají obecně proměnnou délku.

Unicode v operačním systému

V IBM i lze použít kódování Unicode v rámci architektury CDRA, tedy prostřednictvím čísel CCSID. O soustavě Unicode se lze dočíst na internetové stránce

http://www.unicode.org.

Kódování v tomto systému má tři varianty: UTF-8, UTF-16, UTF-32. Z nich jsou v IBM i použity UTF-8 a UTF-16.[1]

UTF-16 je rozšířením UCS-2 a k vyjádření znaku používá jedno až dvě 16bitová slova (pro některé asijské jazyky).

UTF-8 používá pro znaky sekvence proměnlivého počtu 8bitových slov (oktetů čili bajtů). Většina znaků založených na latince má sekvence jednobajtové, znaky s diakritikou jsou dvoubajtové, výjimečně tříbajtové. Znaky ostatních jazyků mohou být až 4bajtové. UTF8 je v poslední době de facto standardem pro internet.

Unicode je podporován v mnoha částech systému, ale nemůže být zadán jako CCSID

- v systémové hodnotě QCCSID
- v uživatelském profilu
- v popisu úlohy (job description)

Kdy je třeba překódovat data

Pro nás jsou nejzajímavější kódové soustavy umožňující pořizovat a zobrazovat znaky s diakritickými znaménky v obvyklých počítačových platformách, tedy zejména v systému Windows, někdy také MS DOS, a v nejrůznějších unixových systémech včetně macOS. Jsou to hlavně tyto soustavy založené na kódovacím schematu ASCII:

Západoevropské jazyky

- Code Page1252 (Windows Latin 1) – CCSID 1252
- Code Page 850 (DOS Latin 1) – CCSID 850
- ISO-8859-1 (ASCII Latin 1) – CCSID 819 (hlavně v unixových systémech)

Středoevropské jazyky

- Cp1250 (Windows Latin 2) – CCSID 1250
- Code Page 852 (DOS Latin 2) – CCSID 852
- ISO-8859-2 (ASCII Latin 2) – CCSID 912 (hlavně v unixových systémech)

Kromě těchto jednobajtových (SBCS) znakových sad jsou k dispozici univerzální vícebajtové soustavy pro jazyky celého světa rovněž založené na kódovacím schematu ASCII:

- UCS-2 – CCSID 13488
- UTF-16 – CCSID 1200
- UTF-8 – CCSID 1208

[1] Základní variantou Unicode je však UTF-32, kde každý znak je zakódován ve čtyřech bajtech. Jde o nejjednodušší univerzální kódování všech dosud známých jazyků. Klade ovšem velké nároky na paměť počítače. UTF-32 nemá v IBM i žádné CCSID. V ISO je jejím ekvivalentem UCS-4.

Pro uživatele IBM i to znamená, že bude často potřebovat převést znaková data z různých variant kódovacího schematu EBCDIC do těchto soustav a zpět. Programátoři statických i dynamických webových stránek se setkávají s nutností převodu dat do systému Unicode, nejčastěji UTF-8. Mnoho programovacích jazyků používá pro znaková data standardně UTF-8 nebo UTF-16.

Systémové funkce API pro překódování

Programátora budou jistě zajímat možnosti překódování znakových dat, které lze použít v programovacích jazycích systému IBM i. Podle časového vývoje jsou to tyto možnosti:

- Funkce QDCXLATE je historicky první a nyní již zastaralá. Umožňuje překódovat znaková data pomocí překódovací tabulky. Pracuje v režimu OPM (Original Program Model) a volá se dynamicky.
- Funkce CDRCVRT alias QTQCVRT je novější a je součástí architektury CDRA. Umožňuje překódovat data mezi různými kódy CCSID a pracuje v režimu OPM.
- Překódování IFS souborů lze provádět pomocí CL příkazů *Cpy*, *CpyFrmStmF*, *CpyToStmF*, *CpyFrmImpF*, *CpyToImpF*.
- Je-li potřeba během překódování souborů provádět úpravy dat, např. vynechávat, přidávat či modifikovat údaje, lze použít funkce API přímo v jazyku RPG. K tomu dobře slouží unixové funkce pro IFS soubory. Při otevírání souboru funkcí *open()* pocházející z unixu lze zadat parametry tak, aby při čtení nebo zápisu dat automaticky proběhlo překódování mezi dvěma různými kódy CCSID.
- Z unixu pochází také konverzní funkce *iconv()*. Dovoluje totéž jako CDRCVRT, ale je použitelná jen v režimu ILE (volaná staticky). Je doplněna o funkce *iconv_open()* a *iconv_close()*, což naznačuje podobu s funkcemi pro IFS soubory. Lze ji tedy použít k převodu IFS souborů, ale poněvadž k tomu nyní existují snadnější způsoby, je vhodná spíše pro přímou komunikaci přes sokety, kde nelze použít funkci open() se zadáním překódování.

V tomto článku si ukážeme na příkladech v jazyku RPG první dvě možnosti, tedy funkci QDCXLATE a funkci CDRCVRT.

Překódovací funkce QDCXLATE

V IBM i je od začátku k dispozici systémový program QDCXLATE, který používá překódovací tabulku. Slouží ovšem pouze pro texty se znaky kódovanými v jednom bajtu – SBCS. Je založen na 256bajtové překódovací (translační) tabulce, což je objekt typu *TBL. Tabulkové objekty existují v systému pro různé jazyky nebo skupiny jazyků a jsou uloženy v knihovnách QSYS a QUSRSYS. Tabulky se vyskytují ve dvojicích - pro překódování “tam a zpět”. Kromě toho si můžeme vytvořit vlastní překódovací tabulky ze zdrojového textu typu TBL příkazem CRTTBL a uložit do libovolné knihovny. Například tabulka T1250870 pro překódování z kódové stránky 1250 (čeština Windows) do stránky 870 (čeština EBCDIC) má tento zdrojový tvar:

```
     00. . . . . . . . . . . . . . . . 10. . . . . . . . . . . . . . . . 1F

00   00010203372D2E2F1605250B0C0D0E0F101112133C3D322618193F271C1D1E1F
20   404F7F7B5B6C507D4D5D5C4E6B604B61F0F1F2F3F4F5F6F7F8F97A5E4C7E6E6F
40   7CC1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E3E4E5E6E7E8E94AE05A5F6D
60   798182838485868788899192939495969798 99A2A3A4A5A6A7A8A9C06AD0A107
80   2021222324150617 2829BC2BAAFDB8B930311A333435360838399C3B8ADDB6B7
A0   417080BA9FB104B5BD09AF0A14CA1BB4902A9E9ABE2C3A3E9DA08FFF776457B2
C0   ED656266637869686771727 3DA7576FAACBBABEEEBEFECBFAE74FEFBFCADB359
E0   CD4542464358494847515253DF5556EA8C9B8BCECBCFCCE18E54DEDBDC8D44B0
```

Tabulku vytvoříme ze zdrojového členu tohoto tvaru pojmenovaného T1250870 a umístěného v souboru QTBLSRC, a to příkazem

```
CRTTBL TBL(*CURLIB/T1250870) SRCFILE(*LIBL/QTBLSRC) SRCMBR(*TBL) TBLTYPE(*CVT)
```

V tabulce odpovídá každé číselné pozici 0 až X'FF' představující znak v kódu 1250 číslo 0 až X'FF' představující znak v kódu 870. Například pozici X'30' odpovídá znak X'F0', což je znak '0' v kódovacím schematu EBCDIC. Některé "znaky" jsou ovšem jen řídicí bitové kombinace, které nemají žádnou grafickou podobu. Obrácená tabulka T8701250 vypadá takto:

```
     00. . . . . . . . . . . . . . . . 10. . . . . . . . . . . . . . . . 1F

00   20010203A609867F97A9AB0B0C0D0E0F10111213AC850887181992AE1C1D1E1F
20   80818283840A171B8889B18BB5050607090116939495960498 99B69B1415B71A
40   20A0E2E4FEE1E3E8E7E65B2E3C282B2126E9EAEBF9EDEEBEE5DF5D242A293B5E
60   2D2FC2C4BDC1C3C8C7C67C2C255F3E3FA1C9CACBD9CDCEBCC5603A2340273D22
80   A2616263646566768699CF2F0FDF8BAB06A6B6C6D6E6F707172B3F19AB8B2A4
A0   B97E737475767778797A8CD2D0DDD8AAFFA5BFDEAFA79E9F8E8FA3D18AA8B4D7
C0   7B41424344454647484 9ADF4F6E0F3F57D4A4B4C4D4E4F505152CCFBFC9DFAEC
E0   5CF7535455565758595AEFD4D6C0D3D530313233343536373839CFDBDC8DDABB
```

Zde naopak pozici X'F0' odpovídá znak X'30', což je znak '0' v kódovacím schematu

ASCII.

Překódovací funkce QDCXLATE má následující čtyři parametry:

- délka dat v pakovaném dekadickém tvaru – 5 číslic, z toho 0 desetinných míst
- vstupní a následně výstupní data v délce dané prvním parametrem
- jméno tabulky – 10 znaků
- jméno knihovny – 10 znaků

V jazyku RPG můžeme překódovat data z kódu 870 do kódu 1250 takto:

```
 ****************************************************************
 *
 *   QDCXLATE - Překódování dat v paměti s použitím API QDCXLATE
 *
 ****************************************************************  *    data pro volání
funkce QDCXLATE
d data            s             30a   inz('01233456789ěščřžýáíéúůďťň')
d data_kopie      s             30a   inz('01233456789ěščřžýáíéúůďťň')
d dataLenDec      s              5p 0 inz(%len(data))
d transTab        s             10a   inz('T8701250')
d lib             s             10a   inz('*LIBL')

 *   volání funkce QDCXLATE
c                   call      'QDCXLATE'
c                   parm                    dataLenDec
c                   parm                    data
c                   parm                    transTab
c                   parm                    lib

c     'QDCXLATE'    dump(a)

c                   eval      *inlr = *on
```

Výsledek volání CALL QDCXLATE (program má stejné jméno jako funkce) vidíme v proměnné DATA ve výpisu paměti:

```
DATA                 '           ÖŁY8¸Ť÷ŕZĎ9Ő¸2     '
                    '30313233333435363738 39EC9AE8F89EFDE1EDE9FAF9EF9DF22020202020'X
```

```
DATA_KOPIE            '01233456789ěščřžýáíéúůďťň     '
                      'F0F1F2F3F3F4F5F6F7F8F9DF9C478EB68D455551DE54EADD8B4040404040'X
DATALENDEC            00030.           '00030F'X
LIB                   '*LIBL     '     '5CD3C9C2D34040404040'X
TRANSTAB              'T8701250  '     'E3F8F7F0F1F2F5F04040'X
```

Překódovací funkce CDRCVRT

Tato funkce, která má též jméno QTQCVRT a má plný název *Convert a Graphic Character String*, je jednou z funkcí API architektury CDRA. Funkce CDRCVRT je již značně obecnější a zároveň snadnější k použití než QDCXLATE. Umožňuje překódování mezi různými kódy CCSID (nejen pro jednobajtové SBCS) a nevyžaduje znalost překódovacích tabulek. Zejména je užitečný převod z určitého kódu CCSID do některé z variant Unicode, tedy do CCSID 1200 nebo CCSID 1208 a obráceně.

Je ovšem zřejmé, že text v určitém jazyce kódovaný v určitém CCSID nemusí být možné překódovat do libovolného jiného kódu prostě proto, že v něm neexistují odpovídající znaky. Například český text kódovaný v CCSID 870 nelze překódovat do ruského kódu CCSID 872 (CYRILLIC PC-DATA) nebo do čínských znaků (CCSID 928). Funkce v tom případě ohlásí chybu 00010001 (Requested conversion is not supported) v prvním slově třísložkového pole zpětné zprávy (feedback).

Podrobný popis funkce je uveden v dokumentaci *National Language Support API – Character Data Representation Architecture (CDRA) APIs* – https://www.ibm.com/docs/en/i/7.4?topic=ssw_ibm_i_74/apis/CDRCVRT.htm.

Příkladem je následující program CDRCVRT, který se jmenuje stejně jako funkce.

```
 ****************************************************************
 *   CDRCVRT - Překódování dat v paměti s použitím API CDRCVRT
 ****************************************************************
 *   prototyp funkce CDRCVRT
d CDRCVRT         pr                  extpgm('CDRCVRT')
d  iCCSID                       10u 0
d  iStringType                  10i 0
d  iString                      30a
d  iStringLen                   10i 0
d  oCCSID                       10u 0
d  oStringType                  10i 0
d  convAlt                      10i 0
d  oStringLen                   10i 0
d  oString                      60a
d  oAreaLen                     10i 0
d  reserved                     10i 0
d  feedback                     10i 0 dim(3)
 *   proměnné pro CDRCVRT API
d iCCSID          s             10u 0
d iStringType     s             10i 0 inz(0)
d iString         s             30a   inz('01233456789ěščřžýáíéúůďťň')
d iStringLen      s             10i 0
d oCCSID          s             10u 0
d oStringType     s             10i 0 inz(0)
d convAlt         s             10i 0 inz(0)
d oStringLen      s             10i 0
d oString         s             60a
d oAreaLen        s             10i 0 inz(32767)
d reserved        s             10i 0
```

```
d feedback        s             10i 0 dim(3) inz(0)
  iCCSID = 870;      // EBCDIC Latin 2
  //oCCSID = 1250;      // Windows Latin 2
  //oCCSID = 912;       // ASCII Latin 2 - ISO8859-2
  //oCCSID = 872;       // Cyrillic PC-Data
  oCCSID = 1208;      // UTF-8
  iStringLen = %len(%trim(iString));  // délka vstupních dat
  oStringLen = 2 * iStringLen;        // délka výstupní oblasti

  // volání překódovací funkce CDRA
  callp CDRCVRT ( iCCSID:
                  iStringType:   // 0=čistý text, 1=text zakončený null
                  iString:       // vstupní text
                  iStringLen:    // délka vstupního textu
                  oCCSID:
                  oStringType:   // 0=čistý text, 1=text zakončený null,
                                 // 2=text doplněný mezerami
                  convAlt:       // konverzní alternativa: 0=installation default,
                                 // 1=CDRA-defined dafault, 57=enforced subset match
                  oStringLen:    // délka výstupního textu
                  oString:       // výstupní překódovaný text
                  oAreaLen:      // čistá délka překódovaných dat
                  reserved:
                  feedback    );

  dump(a)  'CDRCVRT';
  *inlr = *on;
```

Vstupní data jsou deklarována v proměnné délce 30 znaků, ale skutečná délka předávaná proceduře je kratší (bez koncových mezer). Výstupní proměnná je pro jistotu deklarována ve dvojnásobné délce, i když nebude vyčerpána.

Výsledek volání CALL CDRCVRT je vidět ve výseku výpisu paměti. Vstupní data jsou v proměnné ISTRING v čisté délce 25 bajtů. Výsledek překódování z CCSID 870 do CCSID 1208 je v proměnné OSTRING, čistá délka překódovaných dat OAREALEN je 39 bajtů. Parametr CONVALT (konverzní alternativa) je poněkud záhadný: hodnota 0 znamená totéž co hodnota 1. Konverze tedy postupuje standardním způsobem.

```
CONVALT       0               '00000000'X
FEEDBACK      DIM(3)
              0               '00000000'X
ICCSID        870             '00000366'X
ISTRING       '01233456789ěščřžýáíéúůďťň     '
              'F0F1F2F3F3F4F5F6F7F8F9DF9C478EB68D455551DE54EADD8B4040404040'X
ISTRINGLEN    25              '00000019'X
ISTRINGTYPE   0               '00000000'X
OAREALEN      39              '00000027'X
OCCSID        1208            '000004B8'X
OSTRING       '           DńE~DýErE´C¨C~CÝCzCŁEŞDşEvEh                      '
              '3031323333343536373839C49BC5A1C48DC599C5BEC3BDC3A1C3ADC3A9C3BAC5AFC48FC5A5C58840'X
              '4040404040404040404040404040404040404040404040'X
OSTRINGLEN    50              '00000032'X
OSTRINGTYPE   0               '00000000'X
RESERVED      0               '00000000'X
```

Znaku **ě** (X'**DF**') v kódu CCSID 870 odpovídá dvojice bajtů X'**C49B**' v kódu CCSID 1208 (UTF-8).