Překódování IFS souborů v jazyku RPG pomocí funkce iconv()

Vladimír Župka, 2021

Obsah

Překódování pomocí unixové funkce iconv()

Tato metoda je použitelná obecně - v paměti i u IFS souborů, ale uplatní se nejspíše u komunikací TCP/IP přes tzv. sokety.

Vlastní překódovací funkce, která vyhovuje unixovému standardu X/Open, se jmenuje *iconv()* a je doprovázena funkcemi *iconv_open()* pro otevření "konverzního deskriptoru" a *iconv_close()* pro jeho uzavření. Funkce jsou ovšem popsány pro jazyk C v dokumentaci *https://www.ibm.com/docs/en/i/7.4?topic=ssw_ibm_i_74/apis/nls3.htm*. Jsou obsaženy v servisním programu QTQICONV zapsaném ve spojovacím seznamu – binding directory QUSAPIBD.

Otevřením deskriptoru je umožněno několikanásobné použití funkce *iconv()* pro stejný typ konverze (stejnou dvojici čísel CCSID), což je dnes vhodné spíše při převodu komunikačních dat se sokety než při převodu IFS souborů.

Funkce *iconv_open()* vytváří tzv. deskriptor konverze (conversion descriptor). To je datová struktura, kterou dále použije funkce *iconv()* pro překódování dat a naposledy funkce *iconv_close()*, která ji zruší.

Funkce `iconv_open()` – Code Conversion Allocation API

Prototyp funkce `iconv_open()` v jazyku C:

```
#include <iconv.h>
iconv_t iconv_open (char* tocode, char* fromcode)
```

Parametry *tocode* a *fromcode* jsou ukazatele na datové struktury představující způsob konverze (specifikátory konverze), zejména zadávají čísla CCSID pro vstup a výstup.

Struktura *fromcode*

- CHAR(8) 'IBMCCSID'
- CHAR(5) Character representation of CCSID number
- CHAR(3) Conversion alternative
- CHAR(1) Substitution alternative
- CHAR(1) Shift-state alternative
- CHAR(1) Input length option
- CHAR(1) Error option for mixed data
- CHAR(12) Reserved (binary zeros)

Struktura *tocode*

- CHAR(8) 'IBMCCSID'
- CHAR(5) Character representation of CCSID number
- CHAR(19) Reserved (binary zeros)

První dva údaje v každé struktuře jsou celkem jasné, ostatními se nebudeme zabývat, mohou to být znakové nuly, až na pole *Reserved*, kde musí být binární nuly.

Návratová hodnota *iconv_t* je tzv. deskriptor konverze (conversion descriptor), který se používá v dalších funkcích analogicky jako deskriptor souboru v IFS.

Prototyp funkce `iconv_open()` v jazyku RPG:

```
d iconv_open      pr            52a   extproc('iconv_open')
d  toCodeP                        *   value
d  fromCodeP                      *   value
```

Deskriptor konverze v jazyku C:

```
typedef struct {
   int return_value;  // chybový kód
   int cd[12];        // binární údaje deskriptoru
} iconv_t;
```

Deskriptor konverze v jazyku RPG:

```
d cd              ds                  inz
d  return_value                10i 0
d  cd_array                    10i 0 dim(12)
```

Funkce `iconv()` – Code Conversion API

Prototyp funkce `iconv()` v jazyku C:

```
#include <iconv.h>
size_t iconv (cd, inbuf, inbytesleft,
              outbuf, outbytesleft)

iconv_t     cd;             // konverzní deskriptor
char        **inbuf;        // ukazatel na ukazatel na vstupní data
size_t      *inbytesleft;   // zbývající počet bajtů vstupních dat
char        **outbuf;       // ukazatel na ukazatel na výstupní data
size_t      *outbytesleft;  // zbývající počet bajtů výstupních dat
```

Typ size_t je stejný jako typ int. Návratová hodnota je nula (výsledek bez chyby) nebo -1 (chyba). Výsledek konverze je v paměťové oblasti outbuf. Zbývající počet bajtů je zpočátku délka oblasti pro překódování a v průběhu konverze se postupně zmenšuje. Po skončení konverze je to údaj o tom, kolik bajtů z konce oblasti zůstalo nezpracováno.

Prototyp funkce `iconv()` v jazyku RPG:

```
d iconv           pr            10i 0 extproc('iconv')
d  cd                           52a   value
d  inBufPP                        *   value
d  inBytesLeftP                   *   value
d  outBufPP                       *   value
d  outBytesLeftP                  *   value
```

Zde je inBufPP ukazatel na jiný ukazatel inBufP, který musí být definován na jiném místě v programu a který teprve ukazuje na výstupní oblast. Podobně outBufPP.

Funkce *`iconv_close()`* – Code Conversion Deallocation API

Prototyp `iconv_close()` v jazyku C:

```
#include <iconv.h>
int iconv_close (iconv_t cd)
```

Prototyp `iconv_close()` v jazyku RPG:

```
d iconv_close     pr            10i 0 extproc('iconv_close')
d  cd                           52a   value
```

Tato funkce zruší deskriptor konverze.

Příklad na překódování pomocí funkce `iconv()`

Funkci *iconv()* použijeme k překódování IFS souboru, i když pro tento úkol existuje jednodušší způsob (použití funkce open()). Tuto funkci bychom měli ukázat spíše na soketové aplikaci, ale ta by vyžadovala mnohem rozsáhlejší výklad i program. Omezíme se tedy jen na IFS soubory. Překódujeme tedy existující textový IFS soubor kódovaný v určitém CCSID, např. 870, do existujícího souboru kódovaného v jiném CCSID, např. 1208 (UTF-8).

Abychom měli co a kam převádět, vytvoříme si v běžném adresáři dva soubory s předepsaným kódem CCSID. K tomu použijeme program CRTIFSF, který je vytváří pomocí funkce open() – viz předchozí článek o překódování. Do vstupního souboru zapíšeme nějaká textová data, např. programem EDTF, a výstupní necháme prázdný. Oba soubory budou umístěny v běžném adresáři pod jmény, která si zvolíme.

Poznámka1 : V příkladu je použit zdrojový člen PROTOTYPY, v němž jsou uvedeny prototypy unixových funkcí pro práci s IFS soubory.

Poznámka 2: Parametr BNDDIR ve formuláři H musí být doplněn o spojovací seznam (binding directory) QUSAPIBD, kde je k dispozici funkce `iconv()` a s ní související prostředky.

Poznámka 3: Funkce `iconv()` je v příkladu použita k překódování v paměti (z bufferu `inBuf` do bufferu `outBuf`), to znamená, že k jejímu použití není zapotřebí žádných souborů, funguje i v paměti. Jen názvy iconv_open a iconv_close naznačují zamýšlené použití v IFS nebo v soketech.

Program ICONVIFSF – překódování IFS souboru s použitím funkce `iconv()`

```
 ****************************************************************
 *
 *   ICONVIFSF - Překódování IFS souboru ze vstupního souboru
 *               do existujícího výstupního souboru.
 *               s použitím funkce iconv().
 *               Obě CCSID si program zjistí sám funkcí stat().
 *
 ****************************************************************

h dftActGrp(*no) actGrp(*new) bndDir('QC2LE': 'QUSAPIBD')
 /copy QRPGLESRC,PROTOTYPY

 // Prototyp programu
d ICONVIFSF       pr
d  inPath                       256a
d  outPath                      256a

 // Rozhraní programu
d ICONVIFSF       pi
d  inPath                       256a
d  outPath                      256a

 // konverzní deskriptor pro funkci iconv (délky 52)
 *   ------------------------------------------------
d cd              ds                  inz
d  return_value                10i 0
d  cd_array                    10i 0 dim(12)

 // prototyp procedury iconv_open
 // -----------------------------
```

```
d iconv_open      pr            52a   extproc('iconv_open')
d  toCodeP                        *   value
d  fromCodeP                      *   value

 // prototyp procedury iconv
 // ------------------------
d iconv           pr            10i 0 extproc('iconv')
d  cd                           52a   value
d  inBufPP                        *   value
d  inBytesLeftP                   *   value
d  outBufPP                       *   value
d  outBytesLeftP                  *   value

 // prototyp procedury iconv_close
 // ------------------------------
d iconv_close     pr            10i 0 extproc('iconv_close')
d  cd                           52a   value

 // pracovní proměnné pro konverze
d inBuf           s             30a
d inBufP          s               *   inz(%addr(inBuf))
d inBufPSav       s               *   inz(%addr(inBuf))
d inBytesLeft     s             10i 0 inz(%len(inBuf))
d outBuf          s            120a
d outBufP         s               *   inz(%addr(outBuf))
d outBufPSav      s               *   inz(%addr(outBuf))
d outBytesLeft    s             10i 0 inz(%len(outBuf))

 // ukazatele na ukazatele na zbývající počet bajtů
d inBytesLeftP    s               *   inz(%addr(inBytesLeft))
d outBytesLeftP   s               *   inz(%addr(outBytesLeft))

 // délky bufferů
d inBufLen        s             10i 0 inz(%len(inBuf))
d outBufLen       s             10i 0 inz(%len(outBuf))

 // čísla CCSID ve znakovém tvaru
d inCCSID         s              5a
d outCCSID        s              5a

 // ukazatele na ukazatele na data pro funkci iconv
d inBufPP         s               *   inz(%addr(inBufP))
d outBufPP        s               *   inz(%addr(outBufP))

 // specifikátory inicializované binárními nulami
d fromCode        s             32a   inz(*allx'00')
d toCode          s             32a   inz(*allx'00')

 // ukazatele na specifikátory IBMCCSID...
d fromCodeP       s               *   inz(%addr(fromCode))
d toCodeP         s               *   inz(%addr(toCode))

 // návratový kód z funkce iconv
d rc              s             10i 0

d fdIn            s             10i 0
d fdOut           s             10i 0

d oflag           s             10i 0
d mode            s             10i 0
```

```
d rsize           s             10i 0
d wsize           s             10i 0

 /free

  // cesty k souborům zakončené nulou x'00'
  inPath = %trimr(inPath) + x'00';
  outPath = %trimr(outPath) + x'00';

  // zjistím CCSID vstupního souboru (st_ccsid ze struktury StatBuf)
  rc = stat(%addr(inPath): %addr(StatBuf));
  if rc = -1;
     ErrNoP = GetErrNo;
     ErrMsgP = StrError(ErrNo);
     dump(a) 'chyba funkce stat na vstupním souboru';
     *inlr = *on;
     return;
  endif;
  inCCSID= %editc(st_ccsid: 'X');

  // otevřu vstupní soubor
  mode  = S_IRWXU + S_IRWXG + S_IRWXO;
  oflag = O_RDWR + O_TEXTDATA + O_CCSID;
  fdIn = open(inPath: oflag: mode: st_ccsid);
  if fdIn = -1;
     ErrNoP = GetErrNo;
     ErrMsgP = StrError(ErrNo);
     dump(a) 'chyba funkce open na vstupním souboru';
     *inlr = *on;
     return;
  endif;

  // zjistím CCSID výstupního souboru
  rc = stat(%addr(outPath): %addr(StatBuf));
  if rc = -1;
     ErrNoP = GetErrNo;
     ErrMsgP = StrError(ErrNo);
     dump(a) 'chyba funkce stat na výstupním souboru';
     callp close(fdIn); // musím uzavřít vstupní soubor!
     *inlr = *on;
     return;
  endif;
  outCCSID = %editc(st_ccsid: 'X');

  // otevřu výstupní soubor
  mode  = S_IRWXU + S_IRWXG + S_IRWXO;
  oflag = O_RDWR + O_TRUNC + O_TEXTDATA + O_CODEPAGE;
  fdOut = open(outPath: oflag: mode: st_ccsid);
  if fdOut = -1;
     ErrNoP = GetErrNo;
     ErrMsgP = StrError(ErrNo);
     dump(a) 'chyba funkce open na výstupním souboru';
     callp close(fdIn); // musím uzavřít vstupní soubor!
     *inlr = *on;
     return;
  endif;

  // vytvořím specifikátory IBMCCSID... pro konverzi
  fromcode = 'IBMCCSID' + inCCSID + '0000000' +
```

```
                      x'00000000000000000000000';
 tocode = 'IBMCCSID' + outCCSID +
                      x'00000000000000000000000000000000000000';

 // vytvořím deskriptor konverze pro funkci iconv
 cd = iconv_open(toCodeP: fromCodeP);

 // přečtu začátek souboru do bufferu
 rsize = read(fdIn: inBufP: inBufLen);

 // nastavím parametry pro funkci iconv
 inBufP = %addr(inBuf);        // adresa vstupního bufferu (proměnlivá)
 inBufPSav = %addr(inBuf);     // adresa vstupního bufferu (stálá)
 inBytesLeft = rsize;          // zbývající počet vstupních bajtů
 outBufP = %addr(outBuf);      // adresa výstupního bufferu (proměnlivá)
 outBufPSav = %addr(outBuf);   // adresa výstupního bufferu (stálá)
 outBytesLeft = outBufLen;     // zbývající počet výstupních bajtů

 // zpracuji soubory
 dow rsize > 0; // dokud je co číst ze vstupního souboru
                // provedu překódování s použitím konverzního deskriptoru
    rc = iconv ( cd:       // konverzní deskriptor
           inBufPP:        // ukazatel na ukazatel na vstupní data
           inBytesLeftP:   // ukazatel na počet zbývajících vstupních dat
           outBufPP:       // ukazatel na ukazatel na výstupní data
           outBytesLeftP); // ukazatel na počet zbývajících výstupních dat

    // zapíšu překódovanou část z výstupního bufferu do výstupního souboru
    wsize = write(fdOut: outBufPSav: outBufP - outBufPSav);

    // vrátím původní délku výstupního bufferu
    outBytesLeft = outBufLen;

    // znovu nastavím ukazatele na začátky bufferů
    outBufP = outBufPSav;
    inBufP = inBufPSav;

    // přečtu další část souboru do vstupního bufferu
    rsize = read(fdIn: inBufP: inBufLen);

    // opravím zbývající počet vstupních bajtů pro funkci iconv
    inBytesLeft = rsize;
 enddo;

 rc = iconv_close (cd);  // zavřu deskriptor konverze
 rc = close (fdIn);      // zavřu vstupní soubor
 rc = close (fdOut);     // zavřu výstupní soubor

 *inlr = *on;
 return;
/end-free
```