Překódování IFS souborů v jazyku RPG pomocí funkce open()

Vladimír Župka 2021

Obsah

Unixová funkce open()

Ke zpracování IFS souborů se používá různých metod. Jedna z nich je součástí jazyka C a zahrnuje funkce fopen(), fread(), fwrite(), fclose() aj. Jiná je součástí unixových systémů a zahrnuje funkce open(), read(), write(), close() a mnohé další.

Unixový způsob nám umožní vytvořit soubor se zvolenou kódovou stránkou a překódovat data pomocí určitých parametrů funkce *open()*. Tato funkce má podle dokumentace následující prototyp (v jazyku C):

```
#include <fcntl.h>
int open(const char *path, int oflag, . . .);
```

Symbol *fcntl.h* je ve skutečnosti zdrojový člen FCNTL databázového souboru H obsaženého v knihovně QSYSINC. Jsou v něm zapsány pomocné údaje pro fukce ovládající IFS soubory, mezi nimi i pojmenované konstanty určené k volání funkce open(). Konstanty jsou v jazyku C zapsány takto:

```
/*   QSYSINC/H.FCNTL                                          */
#define O_RDONLY   00001        /* Open for reading only       */
#define O_WRONLY   00002        /* Open for writing only       */
#define O_RDWR     00004        /* Open for reading and writing */
#define O_CREAT      00010      /* Create file if it doesn't exist */
#define O_EXCL       00020      /* Exclusive use flag          */
#define O_CCSID      000040     /* The O_CCSID flag is used to
                                   indicate that the conversion
                                   ID parameter should be
                                   interpreted as a CCSID and
                                   mixed conversion is
                                   allowed                @B5A*/
#define O_TRUNC      00100      /* Truncate flag               */
#define O_TEXTDATA     0100000000  /* text data flag            */
#define O_TEXT_CREAT 000200000000 /* This flag is used in
                                   conjunction with the following
                                   flags:  O_CREAT,
                                   O_TEXTDATA, and O_CCSID or
                                   O_CODEPAGE. If all of those
                                   flags are not specified, then
                                   the O_TEXT_CREAT flag is
                                   ignored.  This flag provides
                                   for text conversion on a newly
                                   created file without needing
                                   a second open.         @BAA*/
```

Hodnoty konstant (literály) jsou zapsány v osmičkové (oktalové) číselné soustavě, začínají totiž nulou.

V jazyku RPG pak konstanty musíme definovat jako proměnné se stejnými počátečními hodnotami, ale v šestnáctkové (hexadecimální) soustavě.

```
      *   File Access Modes
     D O_RDONLY        S             10I 0 Inz(X'01')
     D O_WRONLY        S             10I 0 Inz(x'02')
     D O_RDWR          S             10I 0 Inz(x'04')
     D O_CREAT         S             10I 0 Inz(x'08')
     D O_EXCL          S             10I 0 Inz(x'10')
     D O_TRUNC         S             10I 0 INZ(x'40')
      *   File Status Flags
     D O_CCSID         S             10I 0 INZ(x'20')
     D O_TEXTDATA      S             10I 0 INZ(x'01000000')
     D O_TEXT_CREAT    S             10I 0 INZ(x'02000000')
```

Prototyp funkce open() převedeme do jazyka RPG tak, že definujeme typy všech pěti parametrů a vracené hodnoty.

```
D Open            Pr            10I 0 ExtProc('open')
D  PathP                          *   Value Options(*String)
D  Oflag                        10I 0 Value
D  Mode                         10I 0 Value Options(*Nopass)
D  CCSID1                       10I 0 Value Options(*Nopass)
D  CCSID2                       10I 0 Value Options(*Nopass)
```

Vracená hodnota znamená tzv. file descriptor a je definována jako celé číslo (`10I 0`) v prvním řádku prototypu.

Parametr *PathP* je ukazatel (pointer) na cestu k souboru. Cesta k souboru je znakový řetězec (např. '/Applications/S870') ukončený nulovým bajtem, tzv. null terminated string. Údaj `Options(*String)` zajistí doplnění koncového znaku "nulového znaku" `X'00'` (v jazyku C označovaným `\0` nebo `NULL`), když parametr bude zadán jako normální znaková proměnná.

Parametr *Oflag* obsahuje bitové příznaky, které se získají v jazyku C bitovým součtem OR vyjádřeným svislou čarou, například

```
O_CREAT | O_RDWR, | O_TRUNC | O_CCSID, | O_TEXTDATA
```

Převod do jazyka RPG by pak vypadal takto:

```
oflag = O_CREAT + O_RDWR + O_TRUNC + // vytvořit, čtení i zápis, smazat data
         O_CCSID + O_TEXTDATA;         // kódová sránka, textová data
```

Parametry *Mode*, *CCSID1*, *CCSID2* jsou nepovinné. Tři tečky v C prototypu představují nepovinné parametry. Nepovinné parametry v RPG musí být v prototypu všechny zapsány a označeny `Options(*Nopass)`.

Parametr *Mode* určuje přístupová práva k souboru, je-li v parametru Oflag zadán příznak O_CREAT (vytváření souboru).

```
// přístupová práva R=čtení, W=zápisu a X="provedení" souboru (execution)
// U=pro uživatele, G=pro skupinu, O=pro ostatní
mode  = S_IRWXU;
```

Parametr *CCSID1* představuje číslo CCSID pro nově vytvářený soubor, je-li zadán příznak O_CCSID v parametru Oflag. Jestliže výstupní soubor již existuje, parametr se ignoruje.

```
// v proměnné outCCSID je čislo unsigned integer, např. 870
fdOut = open(%trimr(Path): oflag: mode: outCCSID);
```

Vrácená proměnná *fdOut* je celé číslo (file descriptor), které identifikuje soubor při dalších operacích s ním.

Parametr *CCSID2* musí být zadán, jsou-li v parametru Oflag zadán příznak O_TEXT_CREAT a s ním související příznaky O_CCSID, O_TEXTDATA, O_CREAT. Představuje kódovou stránku dat, která budou poskytována při zápisu nebo dat, která budou získána při čtení.

Následující úryvek programu ukazuje, jak můžeme vytvořit nový (dosud neexistující) soubor s kódovou stránkou zadnou v proměnné *outCCSID* (např. 1250), jemuž pak budeme posílat data kódovaná stránkou zadanou v proměnné *inCCSID* (např. 870).

```
d Path            s            256a   inz('/Applications/S870')
d PathP           s               *   inz(%addr(Path))
d oflag           s             10i 0 inz(0)
d mode            s             10i 0 inz(0)
d outCCSID        s             10i 0 inz(1250)
d inCCSID         s             10i 0 inz(870)
```

```
...
  mode  = S_IRWXU + S_IRWXG + S_IRWXO; // všechna přístupová práva
  oflag = O_RDWR +                    // čtení i zápis
          O_TEXTDATA + O_CCSID +      // text, ccsid
          O_CREAT + O_TEXT_CREAT +    // vytvořit, překódovat
          O_EXCL;                     // soubor nesmí existovat
  fdOut = open(outPath: oflag: mode: outCCSID: inCCSID);
```

Příklady na překódování pomocí funkce open()

Kompletní příklady jsou k dispozici v příloze zároveň se zdrojovým členem PROTOTYPY, v němž jsou uvedeny prototypy funkcí a konstanty. V příkladech jsou použity rozmanité unixové funkce k práci se soubory, ale také funkce k obsluze chybových stavů:

- write - zápis dat do souboru
- read - čtení dat ze souboru
- close - uzavření souboru
- getcwd - získání cesty k běžnému adresáři (get current directory)
- stat - získání informací o souboru
- GetErrNo - získání čísla chybové zprávy
- StrError - získání textu chybové zprávy

Tyto funkce zde nevysvětlujeme, předpokládáme, že jejich použití bude dostatečně zřejmé ze zdrojových programů a prototypů uvedených v příloze.

Program CRTIFSF – vytvoření IFS souboru se zadanou kódovou stránkou

Bude vytvořen prázdný soubor v běžném adresáři (zde /home/vzupka), proto stačí zadat jen prosté jméno souboru. Program přijímá dva parametry – jméno souboru (zvolíme S870) a číslo kódové stránky (zvolíme 870). Pro usnadnění zápisu parametrů je k dispozici CL příkaz CRTIFSF (stejného jména jako program):

```
CRTIFSF FILENAME('S870') CCSID(870)
```

```
 ****************************************************************************
 *
 *   CRTIFSF
 *   -------
 *   Program, který vytvoří prázdný IFS soubor v běžném adresáři
 *   se zadanou kódovou stránkou (CCSID)
 *
 ****************************************************************************

h dftActGrp(*no) actGrp(*new) bndDir('QC2LE')

 *   Prototypy a data pro funkce IFS
 /copy QRPGLESRC,PROTOTYPY

 // Prototyp programu
d CRTIFSF1        pr
d  fileName                   255a
d  CCSID                       10u 0

 // Rozhraní programu
d CRTIFSF1        pi
d  fileName                   255a
d  CCSID                       10u 0

 // absolutní cesta k běžnému adresáři a pak k souboru
d Path            s             256a
d PathP           s               *   inz(%addr(Path))

 // příznaky pro funkci open
d oflag           s              10i 0 inz(0)
d mode            s              10i 0 inz(0)

 // deskriptor souboru (vznikne ve funkci open)
d fdOut           s              10i 0 inz(0)

 /free
  // získám absolutní cestu k běžnému adresáři s koncovou nulou
  PathP = getcwd (PathP: 256);

  // do cesty k běžnému adresáři vložím
  // před koncový znak x'00' lomítko a jméno souboru
  Path = %replace('/'+ %trimr(fileName):
                  Path: %scan(x'00': Path): 0);

  // určím přístupová práva čtení, zápisu, provedení
  mode  = S_IRWXU + S_IRWXG + S_IRWXO;  // pro uživatele, skupinu a ostatní

  // určím příznaky pro výstupní soubor
  oflag = O_CREATE + O_RDWR + O_TRUNC + // vytvořit, čtení i zápis, smazat data
          O_CCSID + O_TEXTDATA;         // kódová sránka, textová data

  // pokusím se otevřít výstupní soubor
  fdOut = open(%trimr(Path): oflag: mode: CCSID);

  // nepodaří-li se soubor otevřít, zjistím chybu
  if fdOut = -1;
     ErrNoP = GetErrNo;
     ErrMsgP = StrError(ErrNo);
     dump(a) 'chyba CRTIFSF';
```

```
      *inlr = *on;
      return;
    endif;

    // v případě úspěchu uzavřu výstupní soubor
    callp     close(fdOut);
    *inlr = *on;
    return;  // návrat

   /end-free
```

Do takto vytvořeného souboru je třeba zapsat nějaká data, aby následující příklad měl co překódovat.

Program CVTIFSF – překódování IFS souboru do zadané kódové stránky

Kopírováním dat ze vstupního souboru vytvoříme výstupní soubor se zadanou kódovou stránkou (outCCSID). To znamená, že budeme číst vstupní soubor, jehož kódovou stránku si zjistí program a vstupní buffer bude zároveň výstupním. Při zápisu dat z bufferu do výstupního souboru proběhne překódování do outCCSID. Program přijímá tři parametry:

- cestu ke vstupnímu souboru (inPath)
- cestu k vytvářenému souboru (outPath)
- kódovou stránku vytvářeného souboru (outCCSID)

Ke zjištění kódové stránky existujícího souboru použije program funkci *stat()*, která uloží informace o souboru do datové struktury *StatBuf*. Jedna z jejích položek se jmenuje *st_ccsid* a představuje kódovou stránku. Struktura je definovaná ve zdrojovém členu PROTOTYPY.

Pro snadnější zadávání parametrů je k dispozici CL příkaz CVTIFSF (stejného jména jako program).

```
     ****************************************************************
     *
     *   CVTIFSF - Překódování IFS souboru do zadaného CCSID.
     *             CCSID vstupního souboru si zjistí program.
     *             Výstupní soubor nesmí existovat, vytvoří se nový.
     *
     ****************************************************************

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

     // Prototyp programu
    d CVTIFSF         pr
    d  inPath                       256a
    d  outPath                      256a
    d  outCCSID                       5u 0

     // Rozhraní programu
    d CVTIFSF         pi
    d  inPath                       256a
    d  outPath                      256a
    d  outCCSID                       5u 0

     // pracovní proměnné
    d inBuf           s             300a
    d inBufP          s                *   inz(%addr(inBuf))
    d outBuf          s             300a
    d outBufP         s                *   inz(%addr(outBuf))
```

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

 // návratový kód
d rc              s             10i 0

 // deskriptory souborů
d fdIn            s             10i 0
d fdOut           s             10i 0

 // příznaky do parametrů funkce open
d oflag           s             10i 0
d mode            s             10i 0

 // délky přečtených a zapsaných dat
d rsize           s             10i 0
d wsize           s             10i 0

 /free

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

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

  // otevřu vstupní soubor
  mode  = S_IRWXU + S_IRWXG + S_IRWXO;     // všechna přístupová práva
  oflag = O_RDONLY + O_TEXTDATA + O_CCSID; // jen čtení, text, ccsid
  fdIn = open(inPath: oflag: mode: st_ccsid);
  if fdIn = -1;
     ErrNoP = GetErrNo;
     ErrMsgP = StrError(ErrNo);
     dump(a) 'chyba otevření vstupu';
     *inlr = *on;
     return;
  endif;

  // otevřu výstupní soubor
  mode  = S_IRWXU + S_IRWXG + S_IRWXO; // všechna přístupová práva
  oflag = O_RDWR +                  // čtení i zápis
          O_TEXTDATA + O_CCSID +    // text, ccsid
          O_CREAT + O_TEXT_CREAT +  // vytvořit, překódovat
          O_EXCL;                   // soubor nesmí existovat
  fdOut = open(outPath: oflag: mode: outCCSID: st_ccsid);
  if fdOut = -1;
     ErrNoP = GetErrNo;
     ErrMsgP = StrError(ErrNo);
     dump(a) 'chyba otevření výstupu';
     callp close (fdIn); // musím zavřít vstupní soubor!
     *inlr = *on;
```

```
         return;
       endif;

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

       // zpracuji soubory
       dow rsize > 0; // dokud je něco přečteno
          // zapíšu přečtená data, přičemž proběhne konverze do outCCSID
          wsize = write(fdOut: inBufP: rsize);
          // čtu další data
          rsize = read(fdIn: inBufP: inBufLen);
       enddo;

       rc = close (fdIn);      // zavřu vstupní soubor
       rc = close (fdOut);     // zavřu výstupní soubor

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

Dejme tomu, že jsme programem EDTF do souboru S870 zapsali následující text, který bude zakódován v CCSID 870. Musíme přitom dát pozor, aby v úloze–jobu bylo zadáno také CCSID 870.

```
ě  š  č  ř  ž  ý  á  í  é  CRLF
DF 9C 47 8E B6 8D 45 55 51 0D25
```

Vyvoláme příkaz CVTIFSF s třemi parametry, z nichž cesty k souborům jsou tentokrát zadány absolutně.

```
CVTIFSF INPATH('/home/vzupka/S870') OUTPATH('/home/vzupka/S1208') OUTCCSID(1208)
```

Výstupní soubor nesmí předtím existovat, zadali jsme totiž v programu příznak O_EXCL. Jinak program skončí chybou s výpisem paměti, kde najdeme ERRNO = 3457 a ERRMSG = File exists.

Výsledkem volání bude nově vytvořený soubor S1208 s údaji překódovanými do CCSID 1208 neboli UTF-8:

```
ě    š    č    ř    ž    ý    á    í    é    CRLF
C49B C5A1 C48D C599 C5BE C3BD C3A1 C3AD C3A9 0D0A
```