```
/************************************************************************

    FnIO.cp

************************************************************************/

/*
    These functions allow you to read, write, and print text files in
    the form of a TE Record.

    Functions Include:

      FnAE_OpenTextFile      Puts file into TE Record
      FnIO_SaveTextFile      Save current file
      FnIO_SaveAsTextFile    You get a chance to rename file
      FnIO_PageSetup         Change default print setup
      FnIO_PrintTERecord     Print contents of TE Record

      // Semi-private
      FnIO_CreateFile
      FnIO_ReadFile
      FnIO_WriteFile
      FnIO_pStrCopy
      FnIO_PrintText
      FnIO_PrintDoc
      FnIO_DrawText
*/

#include <Printing.h>

#define TAB_CHAR        ((char)'\t')
#define WATCH           4
#define MAX_CHAR        32000

// Prototypes

int   FnIO_OpenTextFile   ( TEHandle *te,           // Follow with a call
                            /* results */          // to the function
                            Str255    fileName,    // FnTE_DetSBarIncr
                            short     *vRef,
                            short     *refNum );
int   FnIO_SaveTextFile   ( TEHandle *te,
                            Str255    fileName,
                            short     *vRef,
                            short     *refNum );
int   FnIO_SaveAsTextFile( TEHandle *te,
                            Str255    fileName,
                            /* results */
                            short     *vRef,
                            short     *refNum );
void FnIO_PageSetup       ( THPrint *pPrintH );
void FnIO_PrintTERecord ( TEHandle *te, THPrint *pPrintH );

int   FnIO_CreateFile( Str255     fn,
                       short      *vRef,
                       short      *theRef );
int   FnIO_ReadFile   ( short      refNum,
                       TEHandle   *te );
int   FnIO_WriteFile ( short      refNum,
                       char       *p,
                       long       num );
void FnIO_pStrCopy   ( StringPtr p1,
```

```
                        StringPtr p2 );
void FnIO_PrintText ( THPrint *pPrintH,
                        char      **hText,
                        long      length,
                        int       topMargin,
                        int       bottomMargin,
                        int       leftMargin,
                        int       font,
                        int       size,
                        int       tabPixels);
void FnIO_PrintDoc   ( THPrint *pPrintH,
                        char      **hText,
                        long      count,
                        int       topMargin,
                        int       bottomMargin,
                        int       leftMargin,
                        int       font,
                        int       size,
                        int       tabPixels );
void FnIO_DrawText   ( char      *p,
                        int       count,
                        int       topMargin,
                        int       bottomMargin,
                        int       leftMargin,
                        int       tabPixels );

extern void FnErr_DisplayStr(     Str255 s1,
                                  Str255 s2,
                                  Str255 s3,
                                  Str255 s4,
                                  int      quitFlag );


/********** OpenTextFile */

int FnIO_OpenTextFile( TEHandle *te,
                        /* results */
                        Str255     fileName,
                        short      *vRef,
                        short      *refNum )
/*
     Takes TE handle and fills it in with text from a file that the user
     selects.   Modifies fileName, vRef, and refNum based on user input
     and system calls.   Uses FnIO_pStrCopy, FnIO_CreateFile,
     FnErr_DisplayStr, and FnIO_ReadFile functions.
*/

     int            ok;
     Point          SFGwhere =  90, 82 ;
     SFReply        reply;
     SFTypeList fileTypes;
     Str255         fn;
     int            volRef;
     short          referenceNum;
     int            dialogHeight = 136;
     int            dialogWidth = 348;

     ok = true;
     fileTypes[0] = 'TEXT';
     SFGwhere.v = (qd.screenBits.bounds.bottom/3) - (dialogHeight/2);
     SFGwhere.h = (qd.screenBits.bounds.right/2) - (dialogWidth/2);
     SFGetFile( SFGwhere, "\p", 0L, 1, fileTypes, 0L, &reply );
```

```
    if( reply.good )

        FnIO_pStrCopy( reply.fName, fn );
        volRef = reply.vRefNum;
        if( FSOpen( fn, volRef, &referenceNum ) != noErr )

            FnErr_DisplayStr( "\pError opening file ", fileName,
                "\p", "\p", false );
            ok = false;

        else

            FnIO_ReadFile( referenceNum, te );
            FnIO_pStrCopy( fn, fileName );
            *vRef = volRef;
            *refNum = referenceNum;
            FSClose( referenceNum );

            /*
                Should do the following:
                FnTE_DetSBarIncr
                SetWTitle( window, fileName )
                dirty = 0
            */


    else
        ok = FALSE;

    return( ok );



/********** SaveTextFile */

int FnIO_SaveTextFile( TEHandle *te,
                       Str255    fileName,
                       short     *vRef,
                       short     *refNum )
/*
    Takes TE handle and saves it out to existing file.   Uses
    FnErr_DisplayStr, and FnIO_WriteFile functions.
*/

    int       ok;
    SFReply reply;

    ok = TRUE;
    if( FSOpen( fileName, *vRef, refNum ) != noErr )

        FnErr_DisplayStr( "\pError opening file ", fileName,
            "\p", "\p", FALSE );
        ok = FALSE;

    else

        FnIO_WriteFile(
            *refNum,
            (*(***te).hText),
            (long)(***te).teLength );
        FSClose( *refNum );
```

```
        /*
            Should do the following:
            dirty = 0
        */


    return( ok );



/********** SaveAsTextFile */

int FnIO_SaveAsTextFile( TEHandle *te,
                         Str255     fileName,
                         /* results */
                         short      *vRef,
                         short      *refNum )
/*
    Takes TE handle and initial guess of fileName, and saves it out
    to a text file.   Modifies fileName, vRef, and refNum based on user
    input and system calls.   Uses FnIO_pStrCopy, FnIO_CreateFile,
    FnErr_DisplayStr, and FnIO_WriteFile functions.
*/

    int        ok;
    Point      SFPwhere =  106, 104 ;
    SFReply reply;
    int        dialogHeight = 104;
    int        dialogWidth = 304;

    ok = TRUE;
    SFPwhere.v = (qd.screenBits.bounds.bottom/3) - (dialogHeight/2);
    SFPwhere.h = (qd.screenBits.bounds.right/2) - (dialogWidth/2);
    SFPutFile( SFPwhere, "\pSave file as", fileName, 0L, &reply );
    if( reply.good )

        FnIO_pStrCopy( reply.fName, fileName );
        *vRef = reply.vRefNum;
        if( !FnIO_CreateFile( fileName, vRef, refNum ) )

            FnErr_DisplayStr( "\pError creating file ", fileName,
                "\p", "\p", FALSE );
            ok = FALSE;

        else

            FnIO_WriteFile(
                *refNum,
                (*(***te).hText),
                (long)(***te).teLength );
            FSClose( *refNum );

            /*
                Should do the following:
                SetWTitle( window, fileName )
                dirty = 0
            */


    else
        ok = FALSE;
```

```
    return( ok );



/********** PageSetup */

void FnIO_PageSetup( THPrint *pPrintH )
/*
    Takes a pointer to a THPrint handle, initialized to NULL, and
    uses it to display standard page setup dialog.   First time
    through, the Record is initialized to point to a default set of
    data.
*/

    PrOpen();
    if( *pPrintH == NULL )

        *pPrintH = (TPrint **)NewHandle( sizeof( TPrint ) );
        PrintDefault( *pPrintH );

    PrStlDialog( *pPrintH );
    PrClose();



/********** PrintTERecord */

void FnIO_PrintTERecord( TEHandle *te, THPrint *pPrintH )
/*
    Takes pointers to a TE Handle and Print Record, and prints text
    using default font, size, and margins.
*/

    int topMargin      = 36;
    int bottomMargin = 36;
    int leftMargin     = 54;
    int font             = monaco;
    int size             = 10;
    int tabChars       = 4;

    int tabPixels;
    long teLength;

    tabPixels = StringWidth("\pmmmm");
    teLength = (long)(***te).teLength;
    FnIO_PrintText( pPrintH,
                    (***te).hText,
                    teLength,
                    topMargin,
                    bottomMargin,
                    leftMargin,
                    font,
                    size,
                    tabPixels );



/********** CreateFile */

int   FnIO_CreateFile( Str255      fn,
                       short       *vRef,
                       short       *theRef )
```

```
/*
    Creates a file.
*/

    OSErr io;
    OSType creator;
    OSType fileType;
    char *c;

    c = (char *)&creator;
    c[0] = '?';
    c[1] = '?';
    c[2] = '?';
    c[3] = '?';

    c = (char *)&fileType;
    c[0] = 'T';
    c[1] = 'E';
    c[2] = 'X';
    c[3] = 'T';

    io = Create( fn, *vRef, creator, fileType );
    if( (io == noErr) || (io == dupFNErr) )
        io = FSOpen( fn, *vRef, theRef );

    return( (io == noErr) || (io == dupFNErr) );



/********** ReadFile */

int FnIO_ReadFile( short refNum, TEHandle *te )
/*
    Reads text file into referenced TE handle.
*/

    char buffer[256];
    long count;
    int   io;

    TESetSelect( 0, (***te).teLength, *te );
    TEDelete( *te );
    do

        count = 256;
        if( ((***te).teLength + count) >= MAX_CHAR )

            FnErr_DisplayStr(
                "\pFile too large to read, ",
                "\pit has been truncated to fit ",
                "\psize limitations.",
                "\p",
                false );   // don't quit
            io = eofErr;

        else

            io = FSRead( refNum, &count, &buffer );
            TEInsert( &buffer, count, *te );

     while( io == noErr );
```

```
    return( io == eofErr );



/********** WriteFile */

int   FnIO_WriteFile ( short       refNum,
                       char        *p,
                       long        num )
/*
    Writes text to a file.
*/

    int io;

    io = FSWrite( refNum, &num, p );
    if( io == noErr )
        io = SetEOF( refNum, num );
    return( io );



/********** pStrCopy */

void FnIO_pStrCopy   ( StringPtr p1,
                       StringPtr p2 )
/*
    Copies Pascal string from p1 to p2.
*/

    register int len;

    len = *p2++ = *p1++;
    while( --len >= 0 ) *p2++ = *p1++;



/********** PrintText */

void FnIO_PrintText( THPrint *pPrintH,
                     char      **hText,
                     long      length,
                     int       topMargin,
                     int       bottomMargin,
                     int       leftMargin,
                     int       font,
                     int       size,
                     int       tabPixels)
/*
    Prints a TE record.
*/

    TPPrPort    printPort;
    GrafPtr     oldPort;
    TPrStatus prStatus;
    int           copies;

    int           howMany;
    CursHandle hCurs;
    Cursor        waitCursor;

    PrOpen();
```

```
    if( *pPrintH == NULL )

        *pPrintH = (TPrint **)NewHandle( sizeof( TPrint ) );
        PrintDefault( *pPrintH );

    if( tabPixels <= 0 ) tabPixels = StringWidth("\pm");
    SetCursor( &qd.arrow );
    if( PrJobDialog( *pPrintH ) != 0 )

        hCurs = GetCursor( WATCH );
        waitCursor = **hCurs;
        SetCursor( &waitCursor );
        GetPort( &oldPort );
        if( (***pPrintH).prJob.bJDocLoop == bDraftLoop )
            howMany = (***pPrintH).prJob.iCopies;
        else
            howMany = 1;
        for( copies = howMany; copies > 0; copies-- )

            FnIO_PrintDoc( pPrintH,
                           hText,
                           length,
                           topMargin,
                           bottomMargin,
                           leftMargin,
                           font,
                           size,
                           tabPixels );
            if((***pPrintH).prJob.bJDocLoop == bSpoolLoop &&
                PrError() == noErr)
                PrPicFile( *pPrintH, 0L, 0L, 0L, &prStatus );

        SetPort( oldPort );

    PrClose();
    SetCursor( &qd.arrow );



/********** PrintDoc */

void FnIO_PrintDoc ( THPrint *pPrintH,
                     char     **hText,
                     long      count,
                     int       topMargin,
                     int       bottomMargin,
                     int       leftMargin,
                     int       font,
                     int       size,
                     int       tabPixels )
/*
    Function called by FnIO_PrintText.   Calls function FnIO_DrawText.
*/

    register int   line = 0;
    register int   lastLineOnPage = 0;
    int            length;
    Rect           printRect;
    int            linesPerPage;
    int            lineBase;
    int            lineHeight;
    register char *ptr, *p1;
```

```
FontInfo        info;
TPPrPort        printPort;

int             rightMargin = 36;   // don't go off page!
int             max_length;

printPort = PrOpenDoc( *pPrintH, 0L, 0L );
SetPort( (GrafPtr)printPort );
TextFont( font );
TextSize( size );
printRect = (***pPrintH).prInfo.rPage;
GetFontInfo( &info );
lineHeight = info.leading + info.ascent + info.descent;
linesPerPage =
     (printRect.bottom - printRect.top - topMargin - bottomMargin)
     / lineHeight;

// determine max characters per line (assumes proportional font!)
max_length = ( (printRect.right - printRect.left) -
                (leftMargin + rightMargin) ) / (info.widMax-2);

HLock( hText );
ptr = p1 = (*hText);
do

     PrOpenPage( printPort, 0L );
     lastLineOnPage += linesPerPage;
     MoveTo( printRect.left + leftMargin,
          (lineBase = printRect.top + lineHeight + topMargin) );
     do
        // print line
          while((ptr <= (*hText)+count) && (*ptr++ != (char)'\r')) ;
          if((length = (int)(ptr-p1) - 1) > 0)

               if( length > max_length )              // wrap-around text

                    length = max_length;
                    ptr = (char*)(p1 + length);
                    while( *ptr != (char)' ' )

                         *ptr--;
                         length--;

                    *ptr++;
                    length++;
                    if( length <= 0 )

                         length = max_length;
                         ptr = (char*)(p1 + length);


               FnIO_DrawText( p1,
                              length,
                              topMargin,
                              bottomMargin,
                              leftMargin,
                              tabPixels );

          MoveTo( printRect.left + leftMargin,
               (lineBase += lineHeight) );
          p1 = ptr;
      while((++line != lastLineOnPage) && (ptr < (*hText) + count));
```

```
            PrClosePage( printPort );
        while( ptr < (*hText) + count );
        HUnlock( hText );
        PrCloseDoc( printPort );



/********** DrawText */

void FnIO_DrawText ( char      *p,
                     int       count,
                     int       topMargin,
                     int       bottomMargin,
                     int       leftMargin,
                     int       tabPixels )
/*
    Function called by FnIO_PrintDoc.
*/

    register char *p1, *p2;
    int           len;
    Point         pt;

    p1 = p;
    p2 = p + count;
    while( p < p2 )

        while( (p1 < p2) && (*p1 != TAB_CHAR) ) *p1++;
        if( (len = p1 - p) > 0 ) DrawText( p, 0, (p1-p) );
        if( *p1 == TAB_CHAR )

            GetPen( &pt );
            Move( (tabPixels - (pt.h - leftMargin)%tabPixels), 0 );
            *p1++;

        p = p1;



// End of File
```