API-Based File Operations

Commented Demo

-
[DennisMcK]

```
'Example api code snippets to open a file for input, output,
'and append with error trapping.
'Also api code for 'line input', 'print #file', and getting the
'whole file at once.
'Numeric data must be converted to string before writing it
'to a file.

'chr$(13) + chr$(10) is a carriage return + linefeed
'LB strips the carriage return/linefeed with "line input #file"
'and adds it with "print #file". That behavior is also used here.
'LB does not add a crlf if the print statement is followed by
'a semicolon. That behavior is not used here, although it
'can be duplicated with slightly different code. In fact, all
'LB file operations can be done using api calls. Most, if not all, of
'the api calls needed are used here. Having said that, the main reason
'for using api file operations instead of LB's is for the error
'trapping and recovery that can be performed. Maybe LB4 will include
'error trapping and make api file operations unnecessary.

'******************************************************************
'                           snippets
'******************************************************************

eoFile(0) = 0   'global "end of file" indicator
hFile = 0       'file handle

INVALID.HANDLE.VALUE = -1
'Do not use hexdec("&Hffffffff") or _INVALID_HANDLE_VALUE

'******************* Open a file for input ********************

filedialog "Open text file", "*.txt", fileName$
if fileName$ = "" then end

'open a file for input
hFile = OpenForInput(fileName$)

if hFile <> INVALID.HANDLE.VALUE then 'file opened successfully
    'Get file contents one line at a time.
    while eoFile(0) = 0
        content$ = LineInput$(hFile)
        print content$
    wend
    notice "DONE"
    print

    'OR: Get the file contents all at once.
  '  content$ = GetFileContent$(hFile)
  '  print content$
else
    notice "File Access Error" + chr$(13) +
"File could not be opened."
end if

call CloseFile hFile
wait

'******************* Open a file for append *******************

filedialog "Open a file for append", "*.txt", fileName$
if fileName$ = "" then end

'open a file for append
hFile = OpenForAppend(fileName$)

if hFile <> INVALID.HANDLE.VALUE then 'file opened successfully
    'Append some text to the file.
    r = writeFile(hFile,"First appended line")
    r = writeFile(hFile,"Second appended line")
    r = writeFile(hFile,"Third appended line")
    'For additional error checking, r could be tested for 0.
    '0 would indicate that the data was not written to the file.
else
    notice "File Access Error" + chr$(13) +
"File could not be opened."
end if

call CloseFile hFile
wait


'******************* Open a file for output *******************

filedialog "Open a file for output", "*.txt", fileName$
if fileName$ = "" then end
```

```
'open a file for output
hFile = OpenForOutput(fileName$)

if hFile <> INVALID.HANDLE.VALUE then 'file opened successfully
    'Print some text to the file.
    r = writeFile(hFile,"First line of text")
    r = writeFile(hFile,"Second line of text")
    r = writeFile(hFile,"Third line of text")
    'For additional error checking, r could be tested for 0.
    '0 would indicate that the data was not written to the file.
else
    notice "File Access Error" + chr$(13) +
"File could not be opened."
end if

call CloseFile hFile

wait

'*****************************************************************
'                api file handling: subs / functions
'*****************************************************************

function OpenForInput(fileName$)
    'open a file for input.
    calldll #kernel32, "CreateFileA", fileName$ as ptr,
 _GENERIC_READ as ulong, _
        0 as ulong, 0 as long, _OPEN_EXISTING as ulong, _
        _FILE_ATTRIBUTE_NORMAL as ulong, 0 as long,
 OpenForInput as long
end function

function OpenForOutput(fileName$)
    'Open a file for output.
    'If file doesn't exist it will be created.
    'If it does exist, it will be opened and the contents erased.

    calldll #kernel32, "CreateFileA", fileName$ as ptr,
 _GENERIC_WRITE as ulong, _
        0 as ulong, 0 as long, _CREATE_ALWAYS as ulong, _
        _FILE_ATTRIBUTE_NORMAL as ulong, 0 as long,
 OpenForOutput as long
end function

function OpenForAppend(fileName$)
```

```
    'Open a file for append.
    'If the file does not exist it will be created.
    'If it does exist, it will be opened and the file
    'pointer set to the end of the file.

    calldll #kernel32, "CreateFileA", fileName$ as ptr,
 _GENERIC_WRITE as ulong, _
        0 as ulong, 0 as long, _OPEN_ALWAYS as ulong, _
        _FILE_ATTRIBUTE_NORMAL as ulong, 0 as long, hFile as long

    'set the file pointer to the end of the file
    calldll #kernel32,"SetFilePointer", hFile as long, 0 as long, 0
as long, _
        _FILE_END as ulong, r as ulong

    OpenForAppend = hFile
end function

sub CloseFile hFile
    calldll #kernel32, "CloseHandle", hFile as long, r as long
end sub

function LineInput$(hFile)
    'get 1 line from a file opened with OpenForInput
    'without the carriage return/linefeed.

    struct BytesRead, pcb as ulong
    eoFile(0) = 0 'reset "end of file" indicator
    'i(0) is used as a static file pointer variable

    while x = 0 'depend on exit while to leave loop
        pbBuff$ = space$(512)
        calldll #kernel32,"ReadFile", hFile as long, pbBuff$ as ptr, _
        512 as long, BytesRead as struct, 0 as long, r as long

        if BytesRead.pcb.struct > 0 then
            i = instr(pbBuff$, chr$(13))
            if i > 0 then
                LineInput$ = LineInput$ + left$(pbBuff$,i-1)
                fptr = i + 1 + i(0)
                i(0) = fptr
                calldll #kernel32,"SetFilePointer", hFile as long,
 fptr as long, 0 as long, _
                    _FILE_BEGIN as ulong, r as ulong
                exit while
            else
```

```
                LineInput$ = LineInput$ + pbBuff$
                i(0) = i(0) + len(pbBuff$)
            end if
        else
            eoFile(0) = -1
            i(0) = 0
            exit while
        end if
    wend
end function

function GetFileContent$(hFile)
    'get the entire contents of a file opened with OpenForInput
    struct BytesRead, pcb as ulong

    'set the file pointer to the beginning of the file
    calldll #kernel32,"SetFilePointer", hFile as long, 0 as long, 0
as long, _
        _FILE_BEGIN as ulong, r as ulong


'Get the file size. As used here, the max file size this will accurate
ly return
    'is 4,294,967,295 bytes.
    calldll #kernel32,"GetFileSize", hFile as long, 0 as ulong,
 size as ulong

    'read the whole file into pbBuff$
    if size > 0 then
        pbBuff$ = space$(size)
        calldll #kernel32,"ReadFile", hFile as long, pbBuff$ as ptr, _
        size as long, BytesRead as struct, 0 as long, r as long
        GetFileContent$ = pbBuff$
        pbBuff$ = ""
    end if
end function

function writeFile(hFile,data$)

'print a line of text to a file opened with OpenForOutput or OpenForAp
pend
    d$ = data$ + chr$(13) + chr$(10)
    struct BytesWritten, pcb as ulong
    BytesToWrite = len(d$)
    calldll #kernel32,"WriteFile", hFile as long, d$ as ptr, _
    BytesToWrite as long, BytesWritten as struct, 0 as long, r as long
    writeFile = BytesWritten.pcb.struct
end function
```