

FREAD() Function

Returns a specified number of bytes from a file opened with a low-level function.

Syntax

FREAD(*nFileHandle*, *nBytes*)

Returns

Character

Arguments

nFileHandle

Specifies the file handle number for the file from which FREAD() returns data.

nBytes

Specifies the number of bytes returned by FREAD(). FREAD() returns data starting from the current file pointer position and continues until it returns *nBytes* bytes or until it encounters the end of the file.

FREAD() Function Example

The following example uses FREAD() to display the contents of a file. If the file is empty, a message is displayed.

* TEST.TXT must exist -- you can create this file
* using Notepad.

```
Local gnFileHandle, nSize, cString
gnFileHandle = FOPEN("test.txt")
* Seek to end of file to determine the number of bytes in the file
nSize = FSEEK(gnFileHandle, 0, 2)      && Move pointer to EOF
IF nSize <= 0
  * If the file is empty, display an error message
  WAIT WINDOW "This file is empty!" NOWAIT
ELSE
  * If file is not empty, the program stores its contents
  * in memory, then displays the text on the main Visual FoxPro window
  = FSEEK(gnFileHandle, 0, 0)          && Move pointer to BOF
  cString = FREAD(gnFileHandle, nSize)
  ? cString
ENDIF
= FCLOSE(gnFileHandle)                && Close the file
```

FOPEN() Function

Opens a file for use with low-level file functions.

Syntax : FOPEN(*cFileName* [, *nAttribute*])

Returns : Numeric

Arguments

cFileName

Specifies the name of the file to open. *cFileName* can include a path to open files in directories, folders, drives, or volumes not in the current Microsoft Visual FoxPro search path. If a path isn't included, Visual FoxPro searches for the file in the following locations:

- Default directory
- Path established with SET PATH

Note Visual FoxPro will not recognize a path name properly if a disk or directory name contains an exclamation point (!).

nAttribute

Specifies read-write privileges or a buffering scheme for the file you open. The following table lists each number you can include in *nAttribute*, and the read-write file privileges and buffering scheme it establishes.

<i>nAttribute</i>	Read-Write privileges	Buffered/unbuffered
0	(Default) Read Only	Buffered
1	Write-Only	Buffered
2	Read and Write	Buffered
10	Read-Only	Unbuffered
11	Write-Only	Unbuffered
12	Read and Write	Unbuffered

If *nAttribute* isn't included, or if *nAttribute* evaluates to 0, the file is opened as read-only and is buffered.

Note Visual FoxPro will not recognize a path name properly if a disk or directory name contains an exclamation point (!).

Remarks

If FOPEN() successfully opens the file, the file handle number of the file is returned. FOPEN() returns -1 if the file cannot be opened.

Tip Assign the file handle number to a memory variable so you can access the file by the memory variable in other low-level file functions.

The following information about files opened with FOPEN() can be displayed or sent to a printer with DISPLAY STATUS or LIST STATUS.

- Drive and directory or volume and folder, and file name
- File handle number
- File pointer position
- Read-write attributes

FOPEN() Function Example

```
IF FILE('errors.txt')  && Does file exist?
    gnErrFile = FOPEN('errors.txt',12)  && If so, open read-write
ELSE
    gnErrFile = FCREATE('errors.txt')  && If not, create it
ENDIF
IF gnErrFile < 0  && Check for error opening file
    WAIT 'Cannot open or create output file' WINDOW NOWAIT
ELSE  && If no error, write to file
    =FWRITE(gnErrFile, 'Error information to be written here')
ENDIF
=FCLOSE(gnErrFile)  && Close file
MODIFY FILE errors.txt NOWAIT  && Open file in edit window
```

FGETS() Function

Returns a series of bytes from a file or a communication port opened with a low-level file function until it encounters a carriage return.

Syntax

FGETS(*nFileHandle* [, *nBytes*])

Returns

Character

Arguments

nFileHandle

Specifies the numeric file handle of the file or communication port from which FGETS() returns data.

nBytes

Specifies the number of bytes FGETS() returns. FGETS() returns *nBytes* bytes unless a carriage return is encountered first. FGETS() returns data between the starting file-pointer position and the carriage return if a carriage return is encountered within *nBytes* bytes.

If you omit *nBytes*, FGETS() returns a maximum of 254 bytes by default.

Remarks

You can read a file line by line by issuing a series of FGETS().

FGETS() returns a series of bytes as a character string. Data is returned starting from the current file's pointer position and continuing until a carriage return is encountered. The file pointer is then positioned on the byte immediately following the carriage return. The carriage return isn't returned as part of the string, and line feeds are discarded.

FGETS() Function Example

```
*** TEST.TXT must exist ***
STORE FOPEN('test.txt') TO gnFileHandle    && Open the file
STORE FSEEK(gnFileHandle, 0, 2) TO gnEnd    && Move pointer to EOF
STORE FSEEK(gnFileHandle, 0) TO gnTop      && Move pointer to BOF
IF gnEnd <= 0    && Is file empty?
    WAIT WINDOW 'This file is empty!' NOWAIT
ELSE    && If not
    gcString = FGETS (gnFileHandle, gnEnd)    && Store contents
    ? gcString
ENDIF
= FCLOSE(gnFileHandle)    && Close the file
```

FWRITE() Function

Writes a character string to a file opened with a low-level file function.

Syntax

`FWRITE(nFileHandle, cExpression [, nCharactersWritten])`

Returns

Numeric

Arguments

nFileHandle

Specifies the file handle number for the file to which FWRITE() writes.

cExpression

Specifies the character expression that FWRITE() writes to the file specified with *nFileHandle*.

nCharactersWritten

FWRITE() writes the entire character expression to the file unless you include *nCharactersWritten*. When you include *nCharactersWritten*, *nCharactersWritten* characters are written to the file. If *nCharactersWritten* is less than the number of characters in *cExpression*, only *nCharactersWritten* characters are written to the file. All of the characters in *cExpression* are written to the file if *nCharactersWritten* is equal to or greater than the number of characters in *cExpression*.

Remarks

Unlike FPUTS(), FWRITE() doesn't place a carriage return and a line feed at the end of the character string.

FWRITE() returns the number of bytes written to the file. If FWRITE() can't write to the file for any reason, 0 is returned.

FPUTS() Function

Writes a character string, carriage return, and line feed to a file opened with a low-level file function.

Syntax

`FPUTS(nFileHandle, cExpression [, nCharactersWritten])`

Returns

Numeric

Arguments

nFileHandle

Specifies the file handle number for the file to which FPUTS() writes data.

cExpression

Specifies the character expression that FPUTS() writes to the file.

nCharactersWritten

Specifies the number of characters in *cExpression* to write to the file.

FPUTS() writes the entire character expression *cExpression* to the file if you omit *nCharactersWritten*. If you include *nCharactersWritten*, *nCharactersWritten* characters are written to the file. If *nCharactersWritten* is less than the number of characters in *cExpression*, only *nCharactersWritten* characters are written to the file. All of *cExpression* is written to the file if *nCharactersWritten* is equal to or greater than the number of characters in *cExpression*.

Remarks

FPUTS() returns the number of bytes written to the file. Zero is returned if FPUTS() can't write to the file for any reason.

FCREATE() Function

Creates and opens a low-level file.

Syntax : FCREATE(*cFileName* [, *nFileAttribute*])

Returns : Numeric

Arguments : *cFileName*

Specifies the name of the file to create. You can include a drive designator and path with the file name. If a drive designator or path isn't included, the file is created in the default directory.

Note Visual FoxPro will not recognize a path name properly if a disk or directory name contains an exclamation point (!).

nFileAttribute

Specifies the attributes of the file created. The following table lists the file attributes you can specify.

<i>nFileAttribute</i>	File attributes
0	(Default) Read-Write
1	Read-Only
2	Hidden
3	Read-Only/Hidden
4	System
5	Read-Only/System
6	System/Hidden
7	Read-Only/Hidden/System

Note that a file created with *nFileAttribute* other than 0 cannot be written to with FPUTS() or FWRITE() until the file is closed and opened again.

Use DISPLAY STATUS or LIST STATUS to display or print information about files created and opened with FCREATE(). DISPLAY STATUS and LIST STATUS give the following information about each file opened or created with a low-level file function:

- The drive, directory, and file name
- The file handle number
- The file pointer position
- The read-write attributes

Remarks

If a file with the name you specify already exists, it is overwritten without warning.

FCREATE() assigns a file handle number to the file, which you can use to identify the file in other Visual FoxPro low-level file functions. FCREATE() returns the file handle number when a file is created, or returns -1 if the file cannot be created.

Tip Assign the file handle number to a memory variable so you can access the file by the variable in other low-level file functions.

You cannot open a communication port with FCREATE(). Use FOPEN() to open a communication port.

FCREATE() Function Example

```
IF FILE('errors.txt')    && Does file exist?
    gnErrFile = FOPEN('errors.txt',12)    && If so, open read-write
ELSE
    gnErrFile = FCREATE('errors.txt')    && If not create it
ENDIF
IF gnErrFile < 0        && Check for error opening file
    WAIT 'Cannot open or create output file' WINDOW NOWAIT
ELSE && If no error, write to file
    =FWRITE(gnErrFile , 'Error information to be written here')
ENDIF
=FCLOSE(gnErrFile )    && Close file
IF gnErrFile > 0
MODIFY FILE errors.txt NOWAIT    && Open file in edit window
ENDIF
```

FSEEK() Function

Moves the file pointer in a file opened with a low-level file function.

Syntax

FSEEK(*nFileHandle*, *nBytesMoved* [, *nRelativePosition*])

Returns

Numeric

Arguments

nFileHandle

Specifies the file handle for the file in which FSEEK() moves the file pointer. A file handle number is returned by FCREATE() or FOPEN() when the file is created or opened.

nBytesMoved

Specifies the number of bytes to move the file pointer. The file pointer is moved toward the end of the file if *nBytesMoved* is positive. The file pointer is moved toward the beginning of the file if *nBytesMoved* is negative.

nRelativePosition

Moves the file pointer to a relative position in the file. By default, the file pointer is moved relative to the beginning of the file. You can also move the file pointer relative to the current file pointer or the end of the file by including *nRelativePosition*. The following table lists values for *nRelativePosition* and from where the file pointer is moved.

<i>nRelativePosition</i>	Moves the file pointer relative to
0	(Default) The beginning of the file.
1	The current file pointer position.
2	The end of the file.

Remarks

After moving the file pointer, FSEEK() returns the number of bytes the file pointer is positioned from the beginning of the file. The file pointer can also be moved with FREAD() and FWRITE().

FSEEK() Function Example

The following user-defined function uses FSEEK() to return the size of a file. If you don't pass parameters to the user-defined function, it returns -2. If the file cannot be found, the user-defined function returns -1.

```

FUNCTION fsize2
PARAMETERS gcFileName  && File to be checked
PRIVATE pnHandle, pnSize
IF PARAMETERS( ) = 0
    RETURN -2  && Return -2 if no parameter passed
ELSE
    IF !FILE(gcFileName)
        RETURN -1  && Return -1 if file does not exist
    ENDIF
ENDIF
pnHandle = FOPEN(gcFileName)  && Open file
pnSize = FSEEK(pnHandle, 0, 2)  && Determine file size, assign to pnSize
=FCLOSE(pnHandle)  && Close file
RETURN pnSize  && Return value
    
```

FFLUSH() Function

Flushes to disk a file opened with a low-level function.

Syntax : FFLUSH(*nFileHandle*)

Returns : Logical

Arguments : *nFileHandle*

Specifies the file handle of the file to flush to disk.

Remarks : FFLUSH() also releases the memory used by the file's buffer.

FLUSH is different from the FFLUSH() function. FLUSH doesn't operate on low-level files but rather on tables and indexes.

FFLUSH() Function Example

The following example opens and writes to a file named Input.dat. After writing the first two strings, the program flushes the buffers to ensure that the strings are written to disk. It then writes the next two strings, flushes the buffers again and closes the file.

```
IF FILE('input.dat')
    gnTestFile = FOPEN('input.dat',2)
ELSE
    gnTestFile = FCREATE('input.dat')
ENDIF
gnIOBytes = FWRITE(gnTestFile,'Test output')
gnIOBytes = FWRITE(gnTestFile,' for low-level file I/O')
glFlushOk = FFLUSH(gnTestFile)
gnIOBytes = FWRITE(gnTestFile,'Test output2')
gnIOBytes = FWRITE(gnTestFile,' for low-level file I/O')
glFlushOk = FFLUSH(gnTestFile)
glCloseOk = FCLOSE(gnTestFile)
MODIFY FILE input.dat NOWAIT NOEDIT
```

FEOF() Function

Determines whether the file pointer is positioned at the end of a file.

Syntax : FEOF(*nFileHandle*)

Returns : Logical

Arguments : *nFileHandle*

Specifies the file handle number of the file to check for the end-of-file condition. FEOF() always returns true (.T.) if you specify a file handle number of a communication port opened with FOPEN().

Remarks

This low-level file function returns true (.T.) if the file pointer is positioned at the end of a file opened with a low-level file function. FEOF() returns false (.F.) if the file pointer isn't at the end of the file.

FEOF() Function Example

```
*** Open the file test.txt ***
gnFileHandle = FOPEN('test.txt')
*** Move the file pointer to BOF ***
gnPosition = FSEEK(gnFileHandle, 0)
*** If file pointer is at BOF and EOF, the file is empty ***
*** Otherwise the file must have something in it ***
IF FEOF(gnFileHandle)
    WAIT WINDOW 'This file is empty!' NOWAIT
ELSE
    WAIT WINDOW 'This file has something in it!' NOWAIT
ENDIF
= FCLOSE(gnFileHandle)
```