RepFunc=[[DosOpen]] |Remarks=This is part of the [[Family API]]. This call opens a new file, an existing file, a replacement for an existing file, a named pipe, or a device.

Syntax

DosOpen (FileName, FileHandle, ActionTaken,

```
        FileSize, FileAttribute, OpenFlag, OpenMode, Reserved)
```

Parameters

; FileName (PSZ) - input : Address of the ASCIIZ path name of the file, named pipe, or device to be opened. ; FileHandle (PHFILE) - output : Address of the handle for the file, named pipe, or device. ; ActionTaken (PUSHORT) - output : Address of the action taken as a result of DosOpen. 'Value Definition' 0001H File exists 0002H File created 0003H File replaced. ; FileSize (ULONG) - input : File's new logical size (EOD), in bytes. This parameter is significant only when creating a new file or replacing an existing file. Otherwise, it is ignored. ; FileAttribute (USHORT) - input : File attribute bits. Defined below: 'Bit Description' 15-6 Reserved and must be zero. 5 File archive 4 Subdirectory 3 Reserved and must be zero. 2 System file 1 Hidden file 0 Read only file

These bits may be set individually or in combination. For example, an attribute value of 0021H (bits 5 and 0 set to 1) indicates a read-only file that should be archived.

; OpenFlag (USHORT) - input : One-word field indicates the action to be taken if the file exists or does not exist.

'Bit Description' 15-8 Reserved and must be zero. 7-4 0000 = Fail if file does not exist.

```
          0001 = Create file if file does not exist.
```

3-0 0000 = Fail if the file already exists.

```
          0001 = Open the file if it already exists.
          0010 = Replace the file if it already exists.
```

; OpenMode (USHORT) - input : The OpenMode parameter contains the following bit flags:

'Bit Description' 15 DASD Open flag:

```
           0 = FileName represents a file to be opened in the normal way.
           1 = FileName is "Drive:" and represents a mounted disk or
diskette volume
               to be opened for direct access.
```

14 Write-Through flag:

```
           0 = Writes to the file may be run through the file system buffer
cache.
           1 = Writes to the file may go through the file system buffer
```

```
cache but
               the sectors are written (actual file I/O completed) before a
synchronous
               write call returns. This state of the file defines it as a
synchronous file.
               For synchronous files, this is a mandatory bit in that the
data must be
               written out to the medium for synchronous write operations.
```

This bit is not inherited by child processes.

13 Fail-Errors flag. Media I/O errors are handled as follows:

```
           0 = Reported through the system critical error handler.
           1 = Reported directly to the caller by way of return code.
```

```
          Media I/O errors generated through an IOCTL Category 8 function
always get
          reported directly to the caller by way of return code. The Fail-
Errors function
          applies only to non-IOCTL handle-based file I/O calls.
```

This bit is not inherited by child processes.

12 No-Cache/Cache flag:

```
           0 = It is advisable for the disk driver to cache the data in I/O
operations
               on this file.
           1 = I/O to the file need not be done through the disk driver
cache.
```

This bit advises FSDs and device drivers whether it is worth caching the data.

```
          Like the write-through bit, this is a per-handle bit and is not
inherited by
          child processes.
```

11 Reserved and must be zero.

10-8

```
          The locality of reference flags contain information about how the
application
          is to access the file.
          Value      Definition
          000        No locality known.
          001        Mainly sequential access.
          010        Mainly random access.
          011        Random with some locality.
```

7 Inheritance flag:

```
           0 = File handle is inherited by a spawned process resulting from
a
               DosExecPgm call.
           1 = File handle is private to the current process.
               This bit is not inherited by child processes.
```

6-4 Sharing Mode flags.

```
          This field defines any restrictions to file access placed by the
caller on
          other processes:
          '''Value       Definition'''
          001          Deny Read/Write access
          010          Deny Write access
          011          Deny Read access
          100          Deny neither Read or Write access (Deny None). Any
other value is invalid.
```

3 Reserved and must be zero.

2-0 Access Mode flags. This field defines file access required by the caller:

```
          Value           Definition
          000             Read-Only access
          001             Write-Only access
          010             Read/Write access.
```

Any other value is invalid.

Any other combinations are invalid.

File sharing requires the cooperation of sharing processes. This cooperation is communicated through sharing and access modes. Any sharing restrictions placed on a file opened by a process are removed when the process closes the file with a DosClose request.

;Sharing Mode

Specify the type of access other processes may have to the file (sharing mode). For example, if it is permissible for other processes to continue reading the file while your process is operating on it, specify Deny Write. This sharing mode prevents other processes from writing to the file but still allows them to read it.

;Access Mode

Specify the type of access to the file needed by your process (access mode).

For example, if your process requires Read/Write access, and another process has already opened the file with a sharing mode of Deny None, your DosOpen request succeeds. However, if the file is open with a sharing mode of Deny Write, your process is denied access.

If the file is inherited by a child process, all sharing and access restrictions are also inherited.

If an open file handle is duplicated by a call to DosDupHandle, all sharing and access restrictions are also duplicated.

; Reserved (ULONG) - input : Reserved and must be set to zero.

Return Code

rc (USHORT) - return

Return code descriptions are: * 0 NO_ERROR * 2 ERROR_FILE_NOT_FOUND * 3 ERROR_PATH_NOT_FOUND * 4 ERROR_TOO_MANY_OPEN_FILES * 5 ERROR_ACCESS_DENIED * 12 ERROR_INVALID_ACCESS * 26 ERROR_NOT_DOS_DISK * 32 ERROR_SHARING_VIOLATION * 36 ERROR_SHARING_BUFFER_EXCEEDED * 82 ERROR_CANNOT_MAKE * 87 ERROR_INVALID_PARAMETER * 108 ERROR_DRIVE_LOCKED * 110 ERROR_OPEN_FAILED * 112 ERROR_DISK_FULL * 206 ERROR_FILENAME_EXCED_RANGE * 231 ERROR_PIPE_BUSY * 99 ERROR_DEVICE_IN_USE

Remarks

A successful DosOpen request for a file returns a handle to access the file. The read/write pointer is set at the first byte of the file. The pointer's position may be changed by a DosChgFilePtr request or by read and write operations on the file.

The file's date and time can be queried by calling DosQFileInfo, and set by calling DosSetFileInfo.

FileAttribute sets attribute bits for the file object. Attributes of an existing file can be queried and set by DosQFileMode and DosSetFileMode. A file's read-only attribute may also be set with the OS/2 ATTRIB command.

FileAttribute cannot be set to Volume Label. Volume labels cannot be opened. DosSetFSInfo may be issued with a logical drive number to set volume label information.

The FileSize parameter affects the size of the file only when the file is a new file or a replacement for an existing one. If an existing file is simply opened, FileSize is ignored. DosNewSize may be called to change the existing file's size.

The value in FileSize is a recommended size for the file. If allocation of the full size fails, the open may still succeed. The file system makes a reasonable attempt to allocate the new size in as nearly contiguous an area as possible on the medium. When the file size is extended, the value of the new bytes is undefined.

The DASD Open bit provides direct access to an entire disk or diskette volume, independent of the file system. This mode of opening the volume currently mounted on the drive returns a handle to the caller, which represents the logical volume as a single file. The caller specifies this handle with a DosDevIOCtl Category 8 Function 0 request to block other processes from accessing the logical volume.

The file handle state bits can be set by the DosOpen and DosSetFHandState requests. An application can query the file handle state bits as well as the rest of the Open Mode field, by calling

DosQFHandState. If a program running with the NEWFILES bit set tries to create or replace a file with blanks immediately preceding the dot on a FAT drive, the system rejects the name. For example, if c: is a FAT drive, the name “file .txt” is rejected and the name “file.txt” is accepted.

Family API Considerations

Some options operate differently in the DOS mode than in the OS/2 mode. Therefore, the following restrictions apply to DosOpen when coding for the DOS mode:

OpenMode restrictions: * Handles returned in response to DASD open are valid only for DosDevIOCtl. * Inheritance flag is not supported in DOS 2.X. * Write-Through flag must be set to zero. * Fail-errors flag must be set to zero. * Sharing mode field has meaning only if Sharing is loaded in DOS 3.X, ignored if Sharing is not loaded. Sharing mode is not supported in DOS 2.X. * Access mode field has meaning only if Sharing is loaded in DOS 3.X, ignored if Sharing is not loaded. Access mode field is not supported in DOS 2.X. * Access mode is valid only if Sharing is loaded.

Named Pipe Considerations

DosOpen opens the client end of a pipe by name and returns a handle. The open succeeds only if the pipe is in a listening state; otherwise, the open returns with ERROR_PIPE_BUSY. The pipe can be busy because of the following reasons: * All instances of the pipe are already open. * The pipe is closed but is not yet disconnected by the serving end. * No DosConnectNmPipe is issued against the pipe after it is disconnected.

Once a given instance has been opened by a client, that same instance cannot be opened by another client at the same time. Pipes can only be two-ended; however, the opening process can duplicate the open handle as many times as desired.

Pipes are always opened with the pipe-specific states set to B = 0 (to block reads/writes) and RR = 00 (read the pipe as a byte stream). The client can change these modes by calling DosSetNmPHandState if desired.

The access and sharing modes specified on the open must be consistent with those specified on the DosMakeNmPipe request.

Example Code

C Binding

<PRE> #define INCL_DOSFILEMGR

USHORT rc = DosOpen(FileName, FileHandle, ActionTaken, FileSize,

```
                                FileAttribute, OpenFlag, OpenMode, Reserved);
```

PSZ FileName; /* File path name string */ PHFILE FileHandle; /* File handle (returned) */ PUSHORT ActionTaken; /* Action taken (returned) */ ULONG FileSize; /* File primary allocation */ USHORT

FileAttribute; /* File Attribute */ USHORT OpenFlag; /* Open function type */ USHORT OpenMode; /* Open mode of the file */ ULONG 0; /* Reserved (must be zero) */

USHORT rc; /* return code */ </PRE>

This example opens a file. <PRE> #define INCL_DOSFILEMGR

#define OPEN_FILE 0x01 #define CREATE_FILE 0x10 #define FILE_ARCHIVE 0x20 #define FILE_EXISTS OPEN_FILE #define FILE_NOEXISTS CREATE_FILE #define DASD_FLAG 0 #define INHERIT 0x80 #define WRITE_THRU 0 #define FAIL_FLAG 0 #define SHARE_FLAG 0x10 #define ACCESS_FLAG 0x02

#define FILE_NAME “test.dat” #define FILE_SIZE 800L #define FILE_ATTRIBUTE FILE_ARCHIVE #define RESERVED 0L

HFILE FileHandle; USHORT Wrote; USHORT Action; PSZ FileData[100]; USHORT rc;

```
Action = 2;
strcpy(FileData, "Data...");
rc = DosOpen(FILE_NAME,              /* File path name */
             &FileHandle,            /* File handle */
             &Action,                /* Action taken */
             FILE_SIZE,              /* File primary allocation */
             FILE_ATTRIBUTE,         /* File attribute */
             FILE_EXISTS | FILE_NOEXISTS,     /* Open Function type */
             DASD_FLAG | INHERIT |            /* Open mode of the file */
             WRITE_THRU | FAIL_FLAG |
             SHARE_FLAG | ACCESS_FLAG,
             RESERVED);              /* Reserved (must be zero) */
```

</PRE>

MASM Binding

<PRE> EXTRN DosOpen:FAR INCL_DOSFILEMGR EQU 1

PUSH@ ASCIIZ FileName ;File path name string PUSH@ WORD FileHandle ;File handle (returned) PUSH@ WORD ActionTaken ;Action taken (returned) PUSH DWORD FileSize ;File primary allocation PUSH WORD FileAttribute ;File Attribute PUSH WORD OpenFlag ;Open function type PUSH WORD OpenMode ;Open mode of the file PUSH DWORD 0 ;Reserved (must be zero) CALL DosOpen

Returns WORD </PRE>

Note

Text based on http://www.edm2.com/index.php/DosOpen_(FAPI)

Family API		
DOS	Process Manager	DosBeep DosExit DosSleep DosExecPgm
	File Manager	DosChDir DosChgFilePtr DosClose DosDelete DosDupHandle DosMkDir DosMove DosQCurDir DosQCurDisk DosSetFileMode DosOpen DosQFileInfo DosRead DosQFileMode DosQFSInfo DosQVerify DosRmDir DosSelectDisk DosFindClose DosFindFirst DosFindNext DosSetFileInfo DosSetVerify DosWrite DosFileLocks DosSetFHandState DosNewSize DosBufReset DosQFHandState DosSetFSinfo DosShutdown
	Memory Manager	DosFreeSeg DosSubAlloc DosSubFree DosSubSet DosAllocHuge DosAllocSeg DosReallocHuge DosReallocSeg DosGetHugeShift DosCreateCSAlias
	NLS	DosCaseMap DosGetCtryInfo DosGetDBCSEv DosSetCtryCode DosGetCollate DosGetMessage DosInsMessage DosPutMessage
	Date and Time	DosSetDateTime DosGetDateTime
	Devices	DosDevConfig DosDevIOCtl DosDevIOCtl2
	Signals	DosHoldSignal DosSetSigHandler
	Misc	BadDynLink DosGetEnv DosGetMachineMode DosGetVersion DosError DosErrClass DosSetVec
KBD		KbdCharIn KbdFlushBuffer KbdGetStatus KbdSetStatus KbdStringIn KbdPeek
VIO		VioGetBuf VioGetConfig VioGetCurPos VioGetCurType VioGetPhysBuf VioReadCellStr VioReadCharStr VioScrollUp VioScrollDn VioScrollLf VioScrollRt VioScrUnLock VioSetCurPos VioSetCurType VioSetMode VioGetMode VioShowBuf VioWrtCellStr VioWrtCharStr VioWrtCharStrAtt VioWrtNAttr VioWrtNCell VioWrtNChar VioWrtTTY VioScrLock VioPopUp
Tools		BIND
Modules		DOSCALLS.DLL VIOCALLS.DLL KBDCALLS.DLL MSG.DLL
Libraries		API.LIB OS2386.LIB FAPI.LIB DOSCALLS.LIB SUBCALLS.LIB

2018/08/25 15:05 · prokushev · 0 Comments