 This is part of **Family API** which allow to create dual-os version of program runs under OS/2 and DOS

Note: This is legacy API call. It is recommended to use 32-bit equivalent

2021/09/17 04:47 · prokushev · 0 Comments
2021/08/20 03:18 · prokushev · 0 Comments

DosFileLocks

This call locks and unlocks a range in an opened file.

Syntax

```
DosFileLocks (FileHandle, UnLockRange, LockRange)
```

Parameters

- FileHandle (HFILE) - input : File handle.
- UnLockRange (PLONG) - input : Address of the structure containing the offset and length of a range to be unlocked. A doubleword of zero indicates that unlocking is not required.
- FileOffset (ULONG) : The offset to the beginning of the range to be unlocked.
- RangeLength (ULONG) : The length of the range to be unlocked.
- LockRange (PLONG) - input : Address of the structure containing the offset and length of a range to be locked. A doubleword of zero indicates that locking is not required.
- FileOffset (ULONG) : The offset to the beginning of the range to be locked.
- RangeLength (ULONG) : The length of the range to be locked.

Return Code

rc (USHORT) - return

Return code descriptions are:

- 0 NO_ERROR
- 6 ERROR_INVALID_HANDLE
- 33 ERROR_LOCK_VIOLATION
- 36 ERROR_SHARING_BUFFER_EXCEEDED

Remarks

DosFileLocks provides a mechanism that allows a process to lock a region in a file for read/write access. The time a region is locked should be short.

Instead of denying another process read/write access to the entire file by means of access and sharing modes specified with DosOpen or DosOpen2 requests, a process attempts to lock only the range needed for read/write access and examines the error code returned.

A range to be locked must first be cleared of any locked subranges or overlapping ranges. The locked region can be located anywhere in the file, and locking beyond end-of-file is not considered an error.

Once a lock is successful, read/write access by another process to the specified range is denied until the range is unlocked. If both unlocking and locking are specified by a DosFileLocks request, the unlocking operation is performed first. After unlocking is completed, locking is done.

Duplicating the handle duplicates access to any locked regions; however, access to locked regions is not duplicated across the DosExecPgm call.

If a file is closed (either by a DosClose request or by a process terminating) and locks are still in effect, the locks are released in no defined order.

Family API Considerations

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

- If Block = 1 is specified, an “invalid range lock list” or “invalid unlock list” error is returned.
- NewLockIDList is not supported.

Bindings

C Binding

```
#define INCL_DOSFILEMGR

USHORT  rc = DosFileLocks(FileHandle, UnLockRange, LockRange);

HFILE           FileHandle;    /* File handle */
PLONG           UnLockRange;   /* UnLock range */
PLONG           LockRange;     /* Lock range */

USHORT          rc;            /* return code */
```

Example

This example opens a file, writes some data to it, locks a block of the data, and then unlocks it.

```
#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
#define NULL_RANGE 0L

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

struct LockStrc
   {
   long Offset;
   long Range;
   } Area;

int i;

   Action = 2;
   strcpy(FileData, "Data...");
   Area.Offset = 4;
   Area.Range = 100;

   if(!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) */
      {
      for(i=0; i<200; ++i)
         DosWrite(FileHandle,           /* File handle */
                  FileData,             /* User buffer */
                  sizeof(FileData),     /* Buffer length */
                  &Wrote);              /* Bytes written */
      rc = DosFileLocks(FileHandle,           /* File handle */
                        NULL_RANGE,           /* Unlock range */
                        (PLONG) &Area);       /* Lock range */
      rc = DosFileLocks(FileHandle,           /* File handle */
                        (PLONG) &Area,        /* Unlock range */
                        NULL_RANGE);          /* Lock range */
      }
```

MASM Binding

```
EXTRN  DosFileLocks:FAR
INCL_DOSFILEMGR     EQU 1

PUSH   WORD     FileHandle    ;File handle
PUSH@  OTHER    UnLockRange   ;UnLock range
PUSH@  OTHER    LockRange     ;Lock range
CALL   DosFileLocks

Returns WORD
```

Note

Text based on http://www.edm2.com/index.php/DosFileLocks

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