FS_FILEIO

Purpose

Perform multiple lock, unlock, seek, read, and write I/O.

Calling Sequence

```
int far pascal FS_FILEIO (psffsi, psffsd, pCmdList, cbCmdList, poError,
                          IOflag)

struct sffsi far * psffsi;
struct sffsd far * psffsd;
char far * pCmdList;
unsigned short cbCmdList;
unsigned short far * poError;
unsigned short IOflag;
```

Where

psffsi is a pointer to the file-system-independent portion of an open file instance.

psffsd is a pointer to the file-system-dependent portion of an open file instance.

pCmdList is a pointer to a command list that contains entries indicating what commands will be performed.

Each individual operation (*CmdLock*, *CmdUnlock*, *CmdSeek*, *CmdIO*) is performed as atomic operations until all are complete or until one fails. *CmdLock* executes a multiple range lock as an atomic operation. *CmdUnlock* executes a multiple range unlock as an atomic operation. Unlike *CmdLock*, *CmdUnlock* cannot fail as long as the parameters to it are correct, and the calling application had done a Lock earlier, so it can be viewed as atomic.

The validity of the user address is not verified (see *FSH_PROBEBUF*).

For *CmdLock*, the command format is:

```
struct CmdLock {
    unsigned short Cmd = 0;   /* 0 for lock operations          */
    unsigned short LockCnt;   /* number of locks that follow    */
    unsigned long  TimeOut;   /* ms time-out for lock success   */
}
```

which is followed by a series of records of the following format:

```
struct Lock {
    unsigned short Share = 0;   /* 0 for exclusive, 1 for read-only  */
    long           Start;       /* start of lock region              */
    long           Length;      /* length of lock region             */
}
```

If a lock within a *CmdLock* causes a time-out, none of the other locks within the scope of *CmdLock* are in force, because the lock operation is viewed as atomic.

CmdLock. *TimeOut* is the count in milliseconds, until the requesting process is to resume execution if the requested locks are not available. If *CmdLock*. *TimeOut* == 0, there will be no wait. If *CmdLock*. *TimeOut* < 0xFFFFFFFF it is the number of milliseconds to wait until the requested locks become available. If *CmdLock*. *TimeOut* == 0xFFFFFFFF then the thread will wait indefinitely until the requested locks become available.

Lock.Share defines the type of access other processes may have to the file-range being locked. If its value == 0, other processes have No-Access to the locked range. If its value == 1, other process have Read-Only access to the locked range.

For *CmdUnlock*, the command format is:

```
struct CmdUnlock {
    unsigned short Cmd = 1;     /* 1 for unlock operations        */
    unsigned short UnlockCnt;   /* Number of unlocks that follow  */
}
```

which is followed by a series of records of the following format:

```
struct UnLock {
    long Start;                 /* start of locked region         */
    long Length;                /* length of locked region        */
}
```

For *CmdSeek*, the command format is:

```
struct CmdSeek {
    unsigned short Cmd = 2;   /* 2 for seek operation         */
    unsigned short Method;    /* 0 for absolute               */
                              /* 1 for relative to current    */
                              /* 2 for relative to EOF        */
    long           Position;  /* file seek position or delta  */
    long           Actual;    /* actual position seeked to    */
}
```

For *CmdIO*, the command format is:

```
struct CmdIO {
    unsigned short Cmd;        /* 3 for read, 4 for write      */
    void far * Buffer;         /* pointer to the data buffer   */
    unsigned short BufferLen;  /* number of bytes requested    */
    unsigned short Actual;     /* number of bytes transferred  */
}
```

cbCmdList is the length in bytes of the command list.

poError is the offset within the command list of the command that caused the error.

This field has a value only when an error occurs.

The validity of the user address has not been verified (see *FSH_PROBEBUF*).

IOflag	**indicates information about the operation on the handle.**
IOflag == 0x0010	indicates write-through.
IOflag == 0x0020	indicates no-cache.

Remarks

This function provides a simple mechanism for combining the file I/O operations into a single request and providing improved performance, particularly in a networking environment.

File systems that do not have the *FileIO* bit in their attribute field do not see this call: The command list is parsed by the IFS router. The FSD sees only *FS_CHGFILEPTR*, *FS_READ*, *FS_WRITE* calls.

File systems that have the *FileIO* bit in their attribute field see this call in its entirety. The atomicity guarantee applies only to the commands themselves and not to the list as a whole.

Of the information passed in *IOflag*, the write-through bit is a mandatory bit in that any data written to the block device must be put out on the medium before the device driver returns. The no-cache bit, on the other hand, is an advisory bit that says whether the data being transferred is worth caching or not.