FS_FSCTL

Purpose

Allow an extended standard interface between an application and a file system driver.

Calling Sequence

```
int far pascal FS_FSCTL(pArgdat, iArgType, func, pParm, lenParm, plenParmIO,
                        pData, lenData, plenDataIO)

union argdat far * pArgDat;
unsigned short iArgType;
unsigned short func;
char far * pParm;
unsigned short lenParm;
unsigned short far * plenParmIO;
char far * pData;
unsigned short lenData;
unsigned short far * plenDataIO;
```

Where

pArgDat is a pointer to the union whose contents depend on *iArgType*. The union is defined as follows:

```
union argdat {

    /* pArgType = 1, FileHandle directed case */

    struct sf {
        struct sffsi far * psffsi;
        struct sffsd far * psffsd;
    };

    /* pArgType = 2, Pathname directed case */

    struct cd {
        struct cdfsi far * pcdfsi;
        struct cdfsd far * pcdfsd;
        char far *         pPath;
        unsigned short     iCurDirEnd;
    };

    /* pArgType = 3, FSD Name directed case */
    /* pArgDat is Null                      */
};
```

iArgType	**indicates the argument type.**
iArgType = 1	means that *pArgDat→sf.psffsi* and *pArgDat→sf.psffsd* point to an *sffsi* and *sffsd*, respectively.

iArgType	**indicates the argument type.**
iArgType = 2	means that *pArgDat→cd.pcdfsi* and *pArgDat→cd.pcdfsd* point to a *cdfsi* and *cdfsd*, *pArgDat→cd.pPath* points to a canonical pathname, and *pArgDat→cd.iCurDirEnd* gives the index of the end of the current directory in *pPath*. The FSD does not need to verify the *pPath* pointer.
iArgType = 3	means that the call was FSD name routed, and *pArgDat* is a NULL pointer.

func	**indicates the function to perform.**
func == 1	indicates a request for new error code information.
func == 2	indicates a request for the maximum EA size and EA list size supported by the FSD.

pParm is the address of the application input parameter area.

Addressing of this data area has not been validated by the kernel (see *FSH_PROBEBUF*).

lenParm is the maximum length of the application parameter area (*pParm*).

plenParmIO On input, contains the length in bytes of the parameters being passed in to the FSD in *pParm*. On return, contains the length in bytes of data returned in *pParm* by the FSD. The length of the data returned by the FSD in *pParm* must not exceed the length in *lenParm*. Addressing of this area is not validated by the kernel (see *FSH_PROBEBUF*).

pData is the address of the application output data area.

Addressing of this data area is not validated by the kernel (see *FSH_PROBEBUF*).

lenData is the maximum length of the application output data area (*pData*).

plenDataIO On input, contains the length in bytes of the data being passed in to the FSD in *pData*. On return, contains the length in bytes of data returned in *pData* by the FSD. The length of the data returned by the FSD in *pData* must not exceed the length in *lenData*. Addressing of this area is not validated by the kernel (see *FSH_PROBEBUF*).

Remarks

The accessibility of the parameter and data buffers has not been validated by the kernel. *FS_PROBEBUF* must be used.

All FSDs must support *func* == 1 to return new error code information and *func* == 2 to return the limits of the EA sizes.

For *func* == 1, the error code is passed to the FSD in the first WORD of the parameter area. On return, the first word of the data area contains the length of the ASCIIZ string containing an explanation of the error code. The data area contains the ASCIIZ string beginning at the second WORD.

For *func* == 2, the maximum EA and EA list sizes supported by the FSD are returned in the buffer pointed to by *pData* in the following format:

```
struct EASizeBufStruc {
    unsigned short easb_MaxEASize;     /* Max size of an individual EA */
    unsigned long easb_MaxEAListSize;  /* Max full EA list size        */
}
```