FS_FINDFIRST

Purpose

Find first occurrence of a file name in a directory.

Calling Sequence

```
int far pascal FS_FINDFIRST(pcdfsi, pcdfsd, pName, iCurDirEnd, attr, pfsfsi,
                            pfsfsd, pData, cbData, pcMatch, level, flags)

struct cdfsi far * pcdfsi;
struct cdfsd far * pcdfsd;
char far * pName;
unsigned short iCurDirEnd;
unsigned short attr;
struct fsfsi far * pfsfsi;
struct fsfsd far * pfsfsd;
char far * pData;
unsigned short cbData;
unsigned short far * pcMatch;
unsigned short level;
unsigned short flags;
```

Where

pcdfsi is a pointer to the file-system-independent working directory structure.

pcdfsd is a pointer to the file-system-dependent working directory structure.

pName is a pointer to the ASCIIZ name of the file or directory.

Wildcard characters are allowed only in the last component. The FSD does not need to validate this pointer.

iCurDirEnd is the index of the end of the current directory in *pName*.

This is used to optimize FSD path processing. If *iCurDirEnd* == -1 there is no current directory relevant to the name text, that is, a device.

attr is a bit field that governs the match.

Any directory entry whose attribute bit mask is a subset of *attr* and whose name matches that in *pName* should be returned. For example, an attribute of system and hidden is passed in. A file with the same name and an attribute of system is found. This file is returned. A file with the same name and no attributes (a regular file) is also returned. The attributes read-only and file-archive will not be passed in and should be ignored when comparing directory attributes.

The value of *attr* passed to the FSD will be valid. The bit 0x0040 indicates a non-8.3 filename format. It should be treated the same way as system and hidden attributes are.

pfsfsi is a pointer to the file-system-independent file-search structure.

The FSD should not update this structure.

pfsfsd is a pointer to the file-system-dependent file-search structure.

The FSD may use this to store information about continuation of the search.

pData is the address of the application data area.

Addressing of this data area is not validated by the kernel (see *FSH_PROBEBUF*). The FSD will fill in this area with a set of packed, variable- length structures that contain the requested data and matching file name.

cbData is the length of the application data area in bytes.

pcMatch is a pointer to the number of matching entries.

The FSD returns, at most, this number of entries; the FSD returns in this parameter the number of entries actually placed in the data area.

The FSD does not need to validate this pointer.

level is the information level to be returned.

Level selects among a series of data structures to be returned. The level passed to the FSD is valid.

flags	**indicates whether to return file-position information.**
flags == 0	indicates that file-position information should not be returned and the information format described under *DosFindFirst* should be used.
flags == 1	indicates that file-position information should be returned and the information format described below should be used.

The *flag* passed to the FSD has a valid value.

Remarks

For *flags* == 1, the FSD must store in the first DWORD of the per-file attributes structure adequate information to allow the search to be resumed from the file by calling *FS_FINDFROMNAME*. For example, an ordinal representing the file's position in the directory could be stored. If the filename must be used to restart the search, the DWORD may be left blank.

For *level* 0x0001 and *flags* == 0, directory information for *FS_FINDFIRST* is returned in the following format:

```
struct FileFindBuf {
    unsigned short dateCreate;
    unsigned short timeCreate;
    unsigned short dateAccess;
    unsigned short timeAccess;
    unsigned short dateWrite;
    unsigned short timeWrite;
    long           cbEOF;
    long           cbAlloc;
    unsigned short attr;
    unsigned char  cbName;
    unsigned char  szName[];
}
```

For *level* 0x0001 and *flags* == 1, directory information for *FS_FINDFIRST* is returned in the following format:

```
struct FileFromFindBuf {
    long           position;    /* position given to FSD on following */
                                /* FS_FINDFROMNAME call               */
    unsigned short dateCreate;
    unsigned short timeCreate;
    unsigned short dateAccess;
    unsigned short timeAccess;
    unsigned short dateWrite;
    unsigned short timeWrite;
    long           cbEOF;
    long           cbAlloc;
    unsigned short attr;
    unsigned char  cbName;
    unsigned char  szName[];
}
```

The other information levels have similar format, with the position the first field in the structure for *flags* == 1.

If the non-8.3 filename format bit is set in the attributes of a file found by *FS_FINDFIRST/NEXT/FROMNAME*, it must be turned off in the copy of the attributes returned in *pData*.

If *FS_FINDFIRST* for a particular search returns an error, an *FS_FINDCLOSE* for that search will not be issued.

Sufficient information to find the next matching directory entry must be saved in the fsfsd data structure.

In the case where directory entry information overflows the *pData* area, the FSD should be able to continue the search from the entry which caused the overflow on the next *FS_FINDNEXT* or *FS_FINDFROMNAME*.

In the case of a global search in a directory, the first two entries in that directory as reported by the FSD should be '.' and '..' (current and the parent directories.

The example above just shows the effect of *flags* == 1 on a *level* 1 filefind record; *level* 2 and *level* 3 filefind records are similarly affected.

Note: The FSD will be called with the *FINDFIRST/FINDFROMNAME* interface when the 32-bit *DosFindFirst/DosFindNext* APIs are called. THIS IS A CHANGE FROM 1.X IFS interface for redirector FSDs. The kernel will also be massaging the find records so that they appear the way the caller expects. Redirectors who have to resume searches should take this information into account. (i.e. You might want to reduce the size of the buffer sent to the server, so that the position fields can be added to the beginning of all the find records).