FS_OPENPAGEFILE

Purpose

Creates/opens the paging file for the Pager.

Calling Sequence

```
int far pascal FS_OPENPAGEFILE(pFlags, pcMaxReq, pName, psffsi, psffsd,
                               usOpenMode, usOpenFlag, usAttr, Reserved)

unsigned long far * pFlag;
unsigned long far * pcMaxReq;
char far * pName;
struct sffsi far * psffsi;
struct sffsd far * psffsi;
unsigned short usOpenMode;
unsigned short usOpenFlag;
unsigned short usAttr;
unsigned long Reserved;
```

Where

pFlag is a pointer to a flag double word for passing of information between the pager and the file system.

pFlag == 0x00000001	indicates first open of the page file.
pFlag == 0x00004000	indicates physical addresses are required in the page list.
pFlag == 0x00008000	indicates 16:16 virtual addresses are required in the page list.

All other values are reserved.

pcMaxReq is a pointer to a unsigned long where the FSD places the maximum request list length that can be managed by an enhanced strategy device driver.

pName is a pointer to the ASCIIZ path and filename of the paging file.

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.

usOpenMode indicates the desired sharing mode and access mode for the file handle.

See OS/2 Version 2.0 Control Program Programming Reference for a description of the *OpenMode* parameter for *DosOpen*.

usOpenFlag indicates the action taken when the file is present or absent.

See OS/2 Version 2.0 Control Program Programming Reference for a description of the *usOpenFlag* parameter for *DosOpen*.

usAttr are the OS/2 file attributes.

Reserved is a double word parameter reserved for use in the future.

Remarks

Enough information is provided for the FSD to perform a “normal” open/create call.

Since a page file has special requirements about contiguity of its allocations, *FS_OPENPAGEFILE* must assure that any data sectors allocated are returned (Create call only). *FS_ALLOCATEPAGESPACE* will be called to handle file allocation.

If the FSD cannot support the *FS_DOPAGEIO* (usually due to an disk device driver which does not support the Extended strategy entry point), the FSD can return zero (0) for **pcMaxReq*. This tells the kernel file system that it must emulate *FS_DOPAGEIO*.

The FSD can require either physical or virtual (16:16) addresses for subsequent calls to *FS_DOPAGEIO*. This allows an FSD to emulate *FS_DOPAGEIO* without having to worry about dealing with physical addresses.

For a detailed description of the Extended Strategy request interface please see the OS/2 Version 2.0 Physical Device Driver Reference.