FSH_PROBEBUF

Purpose

This function provides the mechanism for performing validity checks on arbitrary pointers to data that users may pass in.

Note: FSDs must check on these pointers before using them.

Calling Sequence

```
int far pascal FSH_PROBEBUF(operation, pdata, cbData)

unsigned short operation;
char far * pData;
unsigned short cbData;
```

Where

operation indicates whether read or write access is desired.

operation == 0	indicates read access is to be checked.
operation == 1	indicates write access is to be checked.
All other values are reserved.	

pData is the starting address of user data.

cbData is the length of user data. If cbData is 0, a length of 64K is indicated.

Returns

If no error is detected, a zero error code is returned. If an error is detected, one of the following error codes is returned:

- ERROR_PROTECTION_VIOLATION – indicates access to the indicated memory region is illegal (access to the data is inappropriate or the user transfer region is partially or completely inaccessible).

Remarks

Because users may pass in arbitrary pointers to data, FSDs must perform validity checks on these pointers before using them. Because OS/2 is multi-threaded, the addressability of data returned by *FSH_PROBEBUF* is only valid until the FSD blocks. Blocking, either explicitly or implicitly allows other threads to run, possibly invalidating a user segment. *FSH_PROBEBUF* must, therefore, be reapplied after every block.

FSH_PROBEBUF provides a convenient method to assure a user transfer address is valid and present in memory. Upon successful return, the user address may be treated as a far pointer and accessed up to the specified length without either blocking or faulting. This is guaranteed until the FSD returns or until the next block.

If *FSH_PROBEBUF* detects a protection violation, the process is terminated as soon as possible. The

OS/2 kernel kills the process once it has exited from the FSD.

On 80386 processors, *FSH_PROBEBUF* ensures all touched pages are physically present in memory so the FSD will not suffer an implicit block due to a page fault. However, *FSH_PROBEBUF* does NOT guarantee the pages will be physically contiguous in memory because FSDs are not expected to do DMA.

FSH_PROBEBUF may block.

Note: OS/2 does not validate input parameters. An FSD, therefore, should call *FSH_PROBEBUF* where appropriate.