Note: This API calls are shared between DOS and Win16 personality.

DPMI is a shared interface for DOS applications to access Intel 80286+ CPUs services. DOS DMPI host provides core services for protected mode applications. Multitasking OS with DOS support also provides DMPI in most cases. Windows standard and extended mode kernel is a DPMI client app. Standard and extended mode kernel differs minimally and shares common codebase. Standard Windows kernel works under DOSX extender. DOSX is a specialized version of 16-bit DPMI Extender (but it is standard DPMI host). Standard mode is just DPMI client, exnhanced mode is DPMI client running under Virtual Machime Manager (really, multitasker which allow to run many DOS sessions). Both modes shares DPMI interface for kernel communication. The OS/2 virtual DOS Protected Mode Interface (VDPMI) device driver provides Version 0.9 DPMI support for virtual DOS machines. Win16 (up to Windows ME) provides Version 0.9 DPMI support. Windows in Standard Mode provides DPMI services only for Windows Applications, not DOS sessions.

DPMI host often merged with DPMI extender. Usually DPMI extender provide DPMI host standard services and DOS translation or True DPMI services.

2021/08/05 10:15 · prokushev · 0 Comments

Int 31H, AH=0DH, AL=00H

Version

1.0

Brief

Allocate Shared Memory

Input

```
AX = 0D00H
ES:(E)DI = selector:offset of shared memory allocation request structure in
the following format:
```

Offset	Length	Contents
00H	4	Requested length of shared memory block (set by client, may be zero)
04H	4	Length actually allocated (set by host)
08H	4	Shared memory handle (set by host)
0CH	4	Linear address of shared memory block (set by host)

Offset	Length	Contents
10H	6	offset32:selector of ASCIIZ (null-terminated ASCII) name for shared memory block (set by client)
16H	2	Reserved
18H	4	Reserved, must be zero

```
if function successful
Carry flag = clear
and the request structure fields at offsets 04H, 08H, and 0CH updated by
host

if function unsuccessful
Carry flag = set
AX = error code
8012H   linear memory unavailable
8013H   physical memory unavailable
8014H   backing store unavailable
8016H   handle unavailable
8021H   invalid value (name for the memory block is too long)
and the request structure fields at offsets 04H, 08H and 0CH unmodified by
host
```

Notes

Allocates a memory block that may be shared by DPMI clients.

For 16-bit programs, the high word of the offset32 for the ASCIIZ name must be zero.

The maximum length of the shared memory block name is 128 characters, including the terminal null character.The linear address provided by the host is guaranteed to be the same for all clients in all virtual machines using a shared memory block. The client must establish addressability for the block by allocating and initializing a descriptor with separate function calls.

No assumptions should be made about handle values. Successive allocations of the same shared memory block by the same client may return distinct handles; the client is responsible for tracking and individually deallocating each handle.

The first client that allocates a shared memory block determines its size; the length requested and the length actually allocated will always be equal, if the allocation succeeds at all. Subsequent allocations by the same or different clients that specify the same or a different size will succeed, but the size of the block will remain unchanged. The actual size of the block is always returned to the client at offset 4 in the shared memory allocation request structure.

Allocation of zero-length shared memory blocks is explicitly allowed. The handle of a zero-length block can be used with the serialization functions (Int 31H Functions 0D02H and 0D03H) as a semaphore for inter-client communication. The linear address that is returned at offset 0CH in the data structure for zero-length blocks is undefined, and any reference to it may produce a page fault.

The first paragraph (16 bytes) of the shared memory block (or the entire shared block, if smaller than 16 bytes) will always be initialized to zero on the first allocation and can be used by clients as an

“area initialized” indicator. For example, a shared memory block might be used by a suite of cooperating client programs to hold a table of static data or a subroutine library. The first client to allocate the shared memory block can obtain exclusive ownership of the block with Int 31H Function 0D02H, load the necessary data or code into the block from disk, set the first 16 bytes of the block to a nonzero value, and finally release its ownership of the block with Int 31H Function 0D03H. Other clients that allocate the shared memory block can check the “area initialized” indicator and know that the desired code or data is already present in memory.

Shared memory block allocations and serializations are tracked by the host on a per client basis. All shared memory allocations for a client are freed by the host when the client terminates.

See also

Note

Text based on http://www.delorie.com/djgpp/doc/dpmi/

DPMI	
Process manager	**INT 2FH** 1680H, 1687H
Signals	
Memory manager	
Misc	**INT 2FH** 1686H, 168AH
Devices	

2021/08/13 14:23 · prokushev · 0 Comments