 This is part of **Family API** which allow to create dual-os version of program runs under OS/2 and DOS

Note: This is legacy API call. It is recommended to use 32-bit equivalent

2021/09/17 04:47 · prokushev · 0 Comments
2021/08/20 03:18 · prokushev · 0 Comments

DosAllocHuge

This call allocates multiple segments as a huge block of memory.

Syntax

```
DosAllocHuge (NumSeg, Size, Selector, MaxNumSeg, AllocFlags)
```

Parameters

- NumSeg (USHORT) - input : Number of 65536-byte segments to be allocated.
- Size (USHORT) - input : Number of bytes to be allocated in the last (non-65536-byte) segment. A value of zero indicates none.
- Selector (PSEL) - output : Address where the selector of the first segment allocated is returned.
- MaxNumSeg (USHORT) - input : Maximum number of 65536-byte segments this object occupies as a result of any subsequent DosReallocHuge. If MaxNumSeg is 0, OS/2 assumes this segment will never be increased by DosReallocHuge beyond its original size, though it may be decreased. This value is ignored in the DOS mode.
- AllocFlags (USHORT) - input : Bit indicators describing the characteristics of the segment allocated. The bits that can be set and their meanings are:

Bit	Description
15-4	Reserved and must be set to zero.
3	If segment is shared, it can be decreased in size by DosReallocHuge.
2	Segment may be discarded by the system in low memory situations.
1	Segment is shareable through DosGetSeg.
0	Segment is shareable through DosGiveSeg.

```
#define SEG_NONSHARED    0
#define SEG_GIVEABLE     1
#define SEG_GETTABLE     2
#define SEG_DISCARDABLE  4
```

Return Code

rc (USHORT) - return

Return code descriptions are:

- 0 NO_ERROR
- 8 ERROR_NOT_ENOUGH_MEMORY
- 87 ERROR_INVALID_PARAMETER
- 212 ERROR_LOCKED

Remarks

DosAllocHuge allows a process to allocate a large amount of memory by telling the system how many 64KB segments it needs and whether it requires an additional partial segment. The system allocates the memory, which is movable and swappable, and returns a selector to the first segment. When this selector is used with a call, the requested function is performed for the entire block of memory.

Each segment of a huge memory allocation has a unique selector. To determine the remaining selectors of a huge memory allocation, issue DosGetHugeShift, which returns a shift count. To compute the next sequential selector, take the value 1 and shift it left by the number of bits specified in shift count. Use the resulting value as an increment to add to the previous selector, using the selector returned by DosAllocHuge as the first selector. For example:

- Assume DosAllocHuge is issued with NumSeg equal to 3, and that the number 63 is returned for the selector of the first segment.
- If DosGetHugeShift returns a shift count of 4, shifting the value “1” by this amount results in an increment of 16.
- Adding this increment to selector number 63 produces 79 for the second selector. Adding the same increment to selector number 79 yields 95 for the third selector.

Like single segment memory allocated with DosAllocSeg, huge memory can be designated as shareable by other processes and discardable by the system when no longer needed. Allocating a huge block of memory as discardable automatically locks the memory for use by the caller. When one segment of a huge allocation is discarded by the system, this forces the discard of all the other segments. See DosAllocSeg for more information relating to discardable and shared segments.

Applications should be discretionary in claiming large memory when doing so can impair system performance. To test system memory availability, issue DosMemAvail. This call returns the size of the largest block of unallocated memory. Although this value can change at any time because of system activity, it can provide a good indication of the system memory state.

Memory allocated by DosAllocHuge is freed by DosFreeSeg. One call to DosFreeSeg, passing the selector returned from a DosAllocHuge, frees all of the memory allocated.

Note: This request may be issued from privilege level 2. However, the segment is allocated as a privilege level 3 segment.

Family API Considerations

Some options operate differently in the DOS mode than in the OS/2 mode. Therefore, the following considerations apply to DosAllocHuge when coding for the DOS mode:

- Requested size value is rounded up to the next paragraph (16-byte)
- Selector is the actual segment address allocated.

Example Code

C Binding

```
#define INCL_DOSMEMMGR

USHORT  rc = DosAllocHuge(NumSeg, Size, Selector, MaxNumSeg, AllocFlags);

USHORT          NumSeg;         /* Number of 65536-byte segments */
USHORT          Size;           /* Number of bytes in last segment */
PSEL            Selector;       /* The first Selector
                                    allocated (returned) */
USHORT          MaxNumSeg;      /* Max number of 65536-byte segments */
USHORT          AllocFlags;     /* Allocation flags */

USHORT          rc;             /* return code */
```

Example

This example requests a block of memory with 4 segments, the last segment having 1,040 bytes. The block of memory will never be larger than 8 segments. The memory can be shared with DosGiveSeg API calls. The system can discard the memory if it needs too.

```
#define INCL_DOSMEMMGR

#define NUMBER_OF_SEGMENTS 4
#define BYTES_IN_LAST_SEGMENT 1040
#define MAXIMUM_SEG_SIZE 8
#define ALLOC_FLAG SEG_GIVEABLE | SEG_DISCARDABLE

SEL    Selector;
USHORT rc;

rc = DosAllocHuge(NUMBER_OF_SEGMENTS,     /* # of 65536-byte segments */
                  BYTES_IN_LAST_SEGMENT,  /* # of bytes in last segment */
                  &Selector,              /* The 1st selector allocated */
                  MAXIMUM_SEG_SIZE,       /* Max number of segments */
                  ALLOC_FLAG);            /* Allocation flags */
```

MASM Binding

```
EXTRN  DosAllocHuge:FAR
INCL_DOSMEMMGR      EQU 1

PUSH   WORD    NumSeg        ;Number of 65536-byte segments
PUSH   WORD    Size          ;Number of bytes in last segment
PUSH@  WORD    Selector      ;The first Selector allocated (returned)
PUSH   WORD    MaxNumSeg     ;Max number of 65536-byte segments
PUSH   WORD    AllocFlags    ;Allocation flags
CALL   DosAllocHuge

Returns WORD
```

Note

Text based on http://www.edm2.com/index.php/DosAllocHuge

Family API		
DOS	Process Manager	DosBeep DosExit DosSleep DosExecPgm
	File Manager	DosChDir DosChgFilePtr DosClose DosDelete DosDupHandle DosMkDir DosMove DosQCurDir DosQCurDisk DosSetFileMode DosOpen DosQFileInfo DosRead DosQFileMode DosQFSInfo DosQVerify DosRmDir DosSelectDisk DosFindClose DosFindFirst DosFindNext DosSetFileInfo DosSetVerify DosWrite DosFileLocks DosSetFHandState DosNewSize DosBufReset DosQFHandState DosSetFSinfo DosShutdown
	Memory Manager	DosFreeSeg DosSubAlloc DosSubFree DosSubSet DosAllocHuge DosAllocSeg DosReallocHuge DosReallocSeg DosGetHugeShift DosCreateCSAlias
	NLS	DosCaseMap DosGetCtryInfo DosGetDBCSEv DosSetCtryCode DosGetCollate DosGetMessage DosInsMessage DosPutMessage
	Date and Time	DosSetDateTime DosGetDateTime
	Devices	DosDevConfig DosDevIOCtl DosDevIOCtl2
	Signals	DosHoldSignal DosSetSigHandler
	Misc	BadDynLink DosGetEnv DosGetMachineMode DosGetVersion DosError DosErrClass DosSetVec
KBD		KbdCharIn KbdFlushBuffer KbdGetStatus KbdSetStatus KbdStringIn KbdPeek
VIO		VioGetBuf VioGetConfig VioGetCurPos VioGetCurType VioGetPhysBuf VioReadCellStr VioReadCharStr VioScrollUp VioScrollDn VioScrollLf VioScrollRt VioScrUnLock VioSetCurPos VioSetCurType VioSetMode VioGetMode VioShowBuf VioWrtCellStr VioWrtCharStr VioWrtCharStrAtt VioWrtNAttr VioWrtNCell VioWrtNChar VioWrtTTY VioScrLock VioPopUp
Tools		BIND
Modules		DOSCALLS.DLL VIOCALLS.DLL KBDCALLS.DLL MSG.DLL
Libraries		API.LIB OS2386.LIB FAPI.LIB DOSCALLS.LIB SUBCALLS.LIB

2018/08/25 15:05 · prokushev · 0 Comments