

Win16 Local Heap Functions

Overview

In 16-bit versions of Windows (1.x, 2.x, 3.x), each module (application or DLL) typically has its own data segment (DGROUP) limited to 64 KB. This segment contains the stack, the local heap, and the atom table. Local heap functions manage memory within this segment using near pointers (offsets). They are exported by the kernel (KRNL386.EXE). Internally, the local heap is managed through a set of data structures that reside inside the segment itself.

A key part of this management is the Instance Data (also called the NULL segment) located at the very beginning of DGROUP, which holds pointers to the heap, atom table, and stack information. The field at offset 6 (pLocalHeap) contains a near pointer to the HeapInfo structure that heads the local heap. This pointer is set by LocalInit() and must be validated by checking the signature word (li_sig) at offset 22h (Standard mode) or 28h (KRNL386) within the presumed heap.

The local heap itself is organized as a series of arenas (headers) that precede each block. The two low bits of the la_prev field in an arena are used as flags: bit 0 indicates whether the block is in use (1) or free (0); bit 1 indicates whether the block is MOVEABLE (1) or FIXED (0). Free blocks are linked via la_free_prev and la_free_next. For MOVEABLE blocks, a separate handle table (pointed to by hi_htable in HeapInfo) stores the actual address, lock count, and flags; each handle is an offset into this table.

Internal Structures

Instance Data (NULL Segment)

The first 16 (10h) bytes of the default data segment (DGROUP) are reserved for system use and are collectively referred to as the Instance Data or NULL segment. This area is present only if the first WORD at offset 0 is zero; otherwise, the structure is not present. The layout is as follows:

| Offset | Type | Field | Description |
|--------|-------|--------------|---|
| 00h | WORD | wMustBeZero | Must be zero for the NULL segment structure to be considered present. |
| 02h | DWORD | dwOldSSSP | When SwitchStackTo() is called, the current SS:SP is stored here. At other times, may contain the value 5 (from C compiler's _rsrvptrs). |
| 06h | WORD | pLocalHeap | Near pointer to the Local Heap information structure (i.e., the HeapInfo structure). This field is set by LocalInit() and points to the beginning of the local heap management structures. If no local heap exists, this field may be stale (non-zero but invalid). Always verify the heap by checking the signature at offset 22h (Standard mode) or 28h (KRNL386) - see li_sig below. |
| 08h | WORD | pAtomTable | Near pointer to the atom table, set by InitAtomTable(). Zero until atoms are used. |
| 0Ah | WORD | pStackTop | Near pointer to the end (top) of the stack. For DLLs, this is zero. |
| 0Ch | WORD | pStackMin | High-water mark of stack usage. For DLLs, zero. |
| 0Eh | WORD | pStackBottom | Near pointer to the beginning (bottom) of the stack. For DLLs, zero. |

Important Notes:

- The field at offset 6 (pLocalHeap) is the primary way to locate the local heap structures given only the DGROUP selector.
- When LocalInit() is called on a globally allocated block (non-DGROUP), the WORD at offset 6 of that block is also set to point to the local heap information structure for that block.
- Similarly, if InitAtomTable() is called on a global block, offset 8 points to the atom table, and offset 6 will point to the associated local heap (since atoms are stored in the local heap).

HeapInfo and LocalInfo

Every local heap begins with an instance of the HeapInfo structure, which is identical to the one used by the global heap and is defined in WINKERN.INC. Its location is given by the pLocalHeap field at offset 6 of the Instance Data. Immediately following the HeapInfo structure are additional fields that, together with HeapInfo, form the LocalInfo structure.

HeapInfo Structure (386)

In the enhanced mode (krnl386), the HeapInfo structure occupies **1Eh** bytes and has the following format (according to “Windows Internals” documentation, Table 2-2):

| Offset | Type | Field | Description |
|--------|-------|-------------|--|
| 00h | WORD | hi_check | If this value is nonzero, the debug version of KERNEL verifies the heap. This field appears to be used only for the local heap, not for the global heap. |
| 02h | WORD | hi_freeze | If this is nonzero, KERNEL should not compact the heap. For the global heap, this value appears to be set only while inside the INT 24h handler. The local heap is frozen during LocalAlloc() and LocalRealloc(). |
| 04h | WORD | hi_count | The total number of blocks in the heap. |
| 06h | DWORD | hi_first | A far pointer to the arena header for the first block in the heap. The first block is always a sentinel and points to itself. |
| 0Ah | DWORD | hi_last | A far pointer to the arena header for the last block in the heap. The last block is always a sentinel and points to itself. |
| 0Eh | BYTE | hi_ncompact | The number of compactations that have been performed on the heap to try and free up memory for a particular allocation. Some code appears to use this field as a count, while other code seems to use it as bitfields. |
| 0Fh | BYTE | hi_dislevel | According to WINKERN.INC, it is the current discard level. Both the local and global heaps use it. The global heap treats the value as a bitfield, using it with flags such as GA_NODISCARD. |
| 10h | DWORD | hi_distotal | Only used by the global heap. When discarding begins, this field contains the number of bytes that need to be discarded. As the global heap discards blocks, it subtracts their sizes from this field. When the field reaches zero or below, discarding stops. |
| 14h | WORD | hi_hstable | This field contains a near pointer to a handle table for the heap. Only the local heap uses this field. |
| 16h | WORD | hi_hfree | Near pointer to the free handle table. Only local heap uses it. |

| Offset | Type | Field | Description |
|--------|------|------------|--|
| 18h | WORD | hi_hdelta | When the local heap needs to increase the number of handles it has, it allocates the number of handles specified in this field. The default value is 20h. |
| 1Ah | WORD | hi_hexpand | A near pointer to the function that KERNEL uses to increase the number of handles for the local heap. Because it's a near pointer, the function must reside in the KERNEL code segment. Thus, there's no way to hook this function. The Global heap does not use it. |
| 1Ch | WORD | hi_pstats | A near pointer to a LocalStats structure which the local heap uses in the debug KERNEL. As the local heap does various things, such as search for free blocks, it increments fields in the structure. The structure is defined in WINKERN.INC. |

HeapInfo Structure (286)

In standard mode (krnl286), the HeapInfo structure occupies 18h bytes. Due to the 16-bit segmented architecture, near pointers are used instead of far pointers, and all size fields are limited to 64 KB. The layout is as follows:

| Offset | Type | Field | Description |
|--------|------|-------------|--|
| 00h | WORD | hi_check | If nonzero, debug kernel verifies the heap. Used only for the local heap. |
| 02h | WORD | hi_freeze | If nonzero, heap compaction is disabled. |
| 04h | WORD | hi_count | Total number of blocks in the heap. |
| 06h | WORD | hi_first | Near pointer to the arena header of the first block (always a sentinel pointing to itself). |
| 08h | WORD | hi_last | Near pointer to the arena header of the last block (always a sentinel pointing to itself). |
| 0Ah | BYTE | hi_ncompact | Number of compactions performed (or bitfield). |
| 0Bh | BYTE | hi_dislevel | Current discard level. |
| 0Ch | WORD | hi_distotal | When discarding begins, this field contains the number of bytes that need to be discarded. As discarding proceeds, the sizes of discarded blocks are subtracted until the value reaches zero or below. (In krnl286 this is a WORD because the heap cannot exceed 64 KB.) |
| 0Eh | WORD | hi_htable | Near pointer to the handle table for moveable blocks. Used only by the local heap. |
| 10h | WORD | hi_hfree | Near pointer to the free handle table entry list. |
| 12h | WORD | hi_hdelta | Number of handle table entries allocated when the table needs to grow. Default is 20h. |
| 14h | WORD | hi_hexpand | Near pointer to the function that expands the handle table. This function must reside in the KERNEL code segment. |
| 16h | WORD | hi_pstats | Near pointer to a LocalStats structure (used in debug kernel). |

Note: In krnl286, all pointer fields are near (16-bit offsets) because the local heap resides in a single 64-KB segment.

LocalInfo Structure (386)

For **krnl386** (HeapInfo size of 1Eh), the LocalInfo structure has the following layout:

| Offset | Type | Field | Description |
|--------|-------|------------|---|
| 00h | - | HeapInfo | 1Eh-byte HeapInfo structure as described above. |
| 1Eh | DWORD | li_notify | Far pointer to a routine called either when a heap block is about to be moved or discarded, or when the heap is out of memory. Initialized to point at LocalNotifyDefault(). |
| 22h | WORD | li_lock | Lock count of the local heap. A non-zero value prevents blocks from moving or being discarded. |
| 24h | WORD | li_extra | Minimum amount by which the local heap should be grown when expanded. Default is 200h. |
| 26h | WORD | li_minsize | Minimum size of the local heap, as specified by the HEAPSIZELINE in the .DEF file. |
| 28h | WORD | li_sig | Signature word set to 484Ch ('LH' in a hex dump). Used by various Windows routines to verify heap integrity. This signature should be checked when validating a potential heap pointer from offset 6. |

LocalInfo Structure (286)

For krnl286, the LocalInfo structure immediately follows the HeapInfo structure and contains additional fields that manage heap notifications, locking, sizing, and a signature word. Its layout begins at offset 18h from the start of the local heap information block:

| Offset | Type | Field | Description |
|--------|-------|------------|--|
| 00h | - | HeapInfo | 16h-byte HeapInfo structure as described above. |
| 18h | DWORD | li_notify | Far pointer to a routine called when a heap block is about to be moved/discarded or when the heap is out of memory. Initialized to point to LocalNotifyDefault(). |
| 1Ch | WORD | li_lock | Lock count of the local heap. A non-zero value prevents blocks from being moved or discarded. |
| 1Eh | WORD | li_extra | Minimum amount (in bytes) by which the local heap should be grown when expanded. Default is 200h. |
| 20h | WORD | li_minsize | Minimum size of the local heap, as specified by the HEAPSIZELINE in the .DEF file. |
| 22h | WORD | li_sig | Signature word set to 484Ch ('LH' in a hex dump). Used by various Windows routines to verify heap integrity. This signature should be checked when validating a potential heap pointer from offset 6 of the Instance Data. |

Important: In krnl286, the li_sig field is located at offset 22h from the beginning of the combined HeapInfo + LocalInfo structure.

The total size of the local heap information block (HeapInfo + LocalInfo) for krnl286 is therefore 24h bytes (18h + 0Ch), though only the first 22h bytes are used up to the signature. The signature itself occupies bytes 22h-23h.

Arena Formats

Every block in the local heap is preceded by an arena (header) that contains management information. Arenas always start on a 4-byte boundary, so the two low bits of every arena address are

zero. These bits are reused as flags in the `la_prev` field of each arena. The two low bits of `la_prev` have the following meaning:

- Bit 0 (least significant): Set if the block is in use (FIXED or MOVEABLE); cleared if the block is free.
- Bit 1: Set if the block is MOVEABLE; cleared if the block is FIXED (only meaningful when bit 0 is set).

Thus, to obtain the real address of the previous arena, the two low bits must be masked off.

There are three arena formats, corresponding to the three possible block states:

FIXED Block Arena

| Offset | Type | Field | Description |
|--------|------|----------------------|--|
| 00h | WORD | <code>la_prev</code> | Near pointer to the preceding arena, with flags in the low two bits. |
| 02h | WORD | <code>la_next</code> | Near pointer to the next arena. |

For a FIXED block, the handle is the address of the block itself. The arena can be found by subtracting 4 from the block address.

MOVEABLE Block Arena

| Offset | Type | Field | Description |
|--------|------|------------------------|--|
| 00h | WORD | <code>la_prev</code> | Near pointer to the preceding arena, with flags. |
| 02h | WORD | <code>la_next</code> | Near pointer to the next arena. |
| 04h | WORD | <code>la_handle</code> | Offset of the handle table entry for this block. |

The `la_handle` field provides two-way mapping between a MOVEABLE block and its handle table entry. Given the block address, subtract 6 to get the arena; the `la_handle` field gives the offset of the handle entry. Given a handle entry, the first field (`lhe_address`) gives the block address.

Free Block Arena

| Offset | Type | Field | Description |
|--------|------|---------------------------|--|
| 00h | WORD | <code>la_prev</code> | Near pointer to the preceding arena, with flags. |
| 02h | WORD | <code>la_next</code> | Near pointer to the next arena. |
| 04h | WORD | <code>la_size</code> | Size of the block, including the arena. |
| 06h | WORD | <code>la_free_prev</code> | Offset of the previous free arena. |
| 08h | WORD | <code>la_free_next</code> | Offset of the next free arena. |

Free blocks are threaded in a doubly linked list using `la_free_prev` and `la_free_next`. The start of this free list is stored in the `la_free_next` field of the first block in the heap (see below).

The First Local Heap Block

The first block in the local heap is special. It resides in memory before the LocalInfo structure. Although its `la_prev` field has the low bit set (indicating a FIXED, in-use arena), the local heap routines treat this arena as if it were free. The `la_free_next` field of this first block points to the first truly free block in the heap.

Handle Table

For MOVEABLE blocks, handles are offsets into a handle table that resides in its own local heap blocks. The first handle table is located via the `hi_htable` field in the HeapInfo structure. Each handle table begins with a WORD specifying how many handle entries follow. After all entries, the last WORD is the offset of the next handle table (or 0 if none). Free handle entries are linked together for quick allocation.

Format of Handle Table Entries (in-use):

| Offset | Type | Field | Description |
|--------|------|--------------------------|--|
| 00h | WORD | <code>lhe_address</code> | Address of the memory block referenced by the handle. |
| 02h | BYTE | <code>lhe_flags</code> | Flags: 0Fh = LHE_DISCARDABLE (discard level), 1Fh = LHE_USERFLAGS (reserved for programmer), 40h = LHE_DISCARDED (block has been discarded). |
| 03h | BYTE | <code>lhe_count</code> | Lock count of the block. Non-zero prevents moving or discarding. |

Format of Handle Table Entries (free):

| Offset | Type | Field | Description |
|--------|------|-----------------------|---|
| 00h | WORD | <code>lhe_link</code> | Offset of the next free handle table entry. |
| 02h | WORD | <code>lhe_free</code> | Always FFFFh to indicate a free entry. |

Heap Operations

- **Allocation (LocalAlloc)** walks the free list, splitting blocks if necessary, and sets up the appropriate arena. For MOVEABLE blocks, it also allocates a handle table entry.
- **Compaction (LocalCompact)** coalesces adjacent free blocks and may move or discard unlocked MOVEABLE blocks. When a block is moved, its `lhe_address` is updated.
- **Locking (LocalLock/LocalUnlock)** manipulates the `lhe_count` field of the handle entry for MOVEABLE blocks; for FIXED blocks, no count is maintained.
- **Discarding (LocalDiscard)** frees the memory of a MOVEABLE block but keeps the handle entry alive with the LHE_DISCARDED flag set.

From:
<http://osfree.ru/doku/> - osFree wiki

Permanent link:
http://osfree.ru/doku/doku.php?id=en:docs:win16:modules:local_heap&rev=1771903132

Last update: 2026/02/24 03:18

