HLock Lock a handle's data area (keep it from moving)

#include <Memory.h> **Memory Manager**

```
void        HLock(theHandle );
Handle      theHandle ;      handle to lock in place
```

HLock sets the lock attribute of a relocatable block, preventing it from being moved during heap compaction. You should lock a handle before you dereference it in order to access its data.

theHandle is a handle leading to a relocatable allocation block. It is typically a value obtained via **NewHandle**.

Returns: none; the **MemError** function may return one of:

noErr	(0)	No error
nilHandleErr	(-109)	Illegal operation on an empty handle
memWZErr	(-111)	Illegal operation on a free block

Notes: Many Toolbox functions end up compacting the heap, causing data to be moved around. See **About the Moves Memory Icon** for more on determining when memory may move. By default, new handles are not locked and must be explicitly locked if you must prevent the data from being moved. Thus, before accessing a handle's data area, be sure to lock the handle; e.g.:

```
Handle  myHandle;
struct  myStruct *msp;
short       x;
Rect        r;

myHandle = NewHandle( sizeof(myStruct) );
HLock( myHandle );
msp = *myHandle;                      /* msp points to the data, but...*/
FrameRect( &r );                      /* this may move unlocked data...*/
x = msp->myField;                     /* making msp invalid */
HUnlock( myHandle );
```

In the example above, the **FrameRect** function could cause a heap compaction, so the value of msp (a pointer to the myHandle data area) could become invalid. By surrounding the sequence with **HLock** and **HUnlock**, you ensure that all pointers to a handle's data area remain valid.

If you are certain that no heap manipulation will take place, or if you make an effort to adjust for it, you need not lock the handle. For instance, in the following sequence, the values of x and y are assigned via double indirection, so even if **FrameRect** causes the data to move, it will be found by both assignment statements.

```
myHandle = NewHandle( sizeof(myStruct) );
x = (*myHandle)->myX;         /* double indirection is valid even */
FrameRect( &r );
```

```
y = (*myHandle)->myY;          /* . . . if the data area has moved */
```

If you intend to lock a block for any length of time, or during a period in which a nonrelocatoable block will be allocated, you are advised to move the handle high in the heap before locking it. You can do this by calling **MoveHHi** before calling **HLock** or by using the convenient function **HLockHi**, which moves the handle high in the heap before locking it.