SetGrowZone Install custom heap zone growing procedure

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

```
void            SetGrowZone(growProc );
GrowZoneProcPtr   growProc ;   address of function to handle grow requests
```

SetGrowZone installs a custom procedure that will be called after exhausting all other avenues available for obtaining a requested block of memory. This affects the current heap zone.

growProc is the address of a Pascal-style function in the format described below. It can be as simple as displaying a message indicating "out of memory" or complex, as in swapping data to disk and releasing memory.

Returns: none

Notes: This function does NOT set the grow zone - it sets the address of a function into Zone.gzProc of the current heap zone.

The custom grow procedure should expect a Size (a.k.a. long) type parameter on entry and should return a 32-bit long. It should be in the following format:

```
pascal long MyGrowFn( Size bytesNeeded )
{
    Handle saveHandle;

    saveHandle = GZSaveHnd( );            /* always call this */
        .
        .    Attempt to free or unlock memory . . .
        .    . . . but don't alter saveHandle
        .
    return( 0 );   /* zero if unable to free or unlock any memory */
    return( 1 );   /* non-zero if able to make some progress */
}

/* Early in the program, install this procedure as follows */

SetZone( whateverZone );            /* make desired zone current */
SetGrowZone( MyGrowFn );
```

On entry, *bytesNeeded* is the required amount of memory, including the block header (i.e., the actual amount needed). The custom function should attempt to find places where the application can economize; e.g., it can call **EmptyHandle** to purge transient data or it can write important data to a disk file (for later recovery) and mark the block as purgeable (see **HPurge**). It can also unlock (**HUnlock**) blocks, in the hope that this will unfragment the heap, thus freeing a larger block.

Note: a single handle, the "grow zone root" handle should not be disturbed by your function. See **GZSaveHnd**.

On exit, the grow-zone function should return 0 if it was unable to free or unlock any memory. Before doing so, it should probably display an alert telling the user about the memory shortage. The function should return a non-zero value if it was able to free or unlock any blocks.

Since the grow zone proc is called from inside the Memory Manger, it's possible that the A5 register will not be correct for your application. If you need to use globals or statics, or if you want make intersegment function calls, you must restore the A5 register by using the global CurrentA5. The easiest way to accomplish this is to use the OS utility functions **SetA5** and **SetCurrentA5**.

The Memory Manager will compact the heap, purge purgeable blocks, and attempt normal heap growth (for the application heap). If this fails to yield a block of the requested size, it will call the custom function repeatedly (followed by another attempt to purge and compact) until the function returns a 0, at which point the Memory Manager will give up and fail the system call that requested the allocation.