Technical Note TN1118

Unlocking GDHandles Considered Harmful

CONTENTS

Recently, Apple became aware of problems in Mac OS and in third-party code which can cause a crash.

The problem occurs when a program inadvertently unlocks one or more QuickDraw data structures. The crash typically occurs elsewhere, when the system accesses one of these data structures under the assumption that it is locked.

This Technote is addressed to two audiences: developers whose products make any use of the `GDHandle` data structure, and users who seek more information on certain intermittent crashes reported in other media.

Updated: [Jun 15 1998]

The Symptoms

The problem manifests most often as a crash in `StdText` (MacsBug reports the crash in `NQDStdText`, which is the PowerPC version of `StdText`). The crash occurs when a half-dereferenced `GDHandle` is accessed after a call which can (and does) move memory.

It should be noted that a crash could theoretically appear in a number of places as a result of the problem. The problem generally does not occur near the site of the crash. One of the more insidious potential examples of this is in the cursor drawing routines, which are executed at interrupt time. An unlocked `GDHandle` could be accessed while it is being moved by the Memory Manager, although this is considered a rare, if at all extant, symptom.

Back to top

The Problem

`NewGDevice` locks a `GDHandle` immediately after creating it, and the rest of QuickDraw assumes it will stay locked. The problem is that some programs, including numerous third-party applications and some versions of Apple's Monitors & Sound control panel (including the version in Mac OS 8) contain code like this:

```
// begin problematic code
GDHandle gdh = GetMainDevice ( );
HLock ((Handle)gdh);
// do something with or to gdh
// which moves memory (requires gdh to be locked)
HUnlock ((Handle)gdh);
// this is the problem line
// end problematic code
```

The above code assumes that the GDHandle is unlocked before the code executes, and unlocks it as the code ends. (Ironically, there was never any need to lock this GDHandle.)

Note:
Versions of Monitors & Sound which shipped in Mac OS 8.1 and later do not have this problem.

There has always been a requirement that a GDHandle in the system's graphic device list remain locked. The requirement is present in the first-released implementation of Color QuickDraw (the Mac II ROMs) and the original source code contains comments to the effect that a GDHandle in the system's graphic device list must never be unlocked.

Thus, the potential for the problem has always been present, though the symptoms have only recently been isolated. Because of the complex way in which relocatable Memory Manager blocks move over time, programs which cause the problem have not necessarily also caused the symptoms, and when they have caused the symptoms, the symptoms have not always been easily reproducible. (The offending programs have been "getting away with it.")

An exhaustive search of Apple's previous documentation turned up no instances of explicit prohibitions against unlocking a GDHandle in the system's graphic device list. Informally communicated Mac OS programming lore has always held that you should be careful when changing the state of a handle not created by your own code. We are now making this rule formal and explicit. (See the Developer Solution section, below, for details.)

Back to top

The Solutions

The previous version of this Technote claimed there would be a solution which users could apply. In fact, there is only a developer solution.

Developer Solution

You should make sure none of your code unlocks a GDHandle in the system's graphic device list. Use code like the following if you need to make sure a GDHandle is locked while you access it:

```
GDHandle gdh = GetMainDevice ( );
SInt8 hState = HGetState ((Handle) gdh);
   HLock ((Handle)gdh);
    // do something with or to gdh
    // which moves memory (requires gdh to be locked)
   HSetState ((Handle)gdh,hState);
```

This code saves the handle's state, locks the handle, and restores the handle's previous state. This means that if the handle is locked before this code runs, it will stay locked afterward, and if the handle is unlocked before this code runs, it will be unlocked afterward.

In a perfect world, you would not need to lock or unlock any GDHandle, because it would always be locked before your program ever saw it. However, the possibility that other programs may erroneously unlock a GDHandle in the system's graphic device list is real. The above code shows one proper way to protect yourself from that possibility.

Another technique would be to avoid locking a GDHandle entirely and, instead, copy its fields into local variables as necessary. Most fields of the GDHandle are small and should not be inconvenient to copy.

Apple has not found any cases in which you need to keep any GDHandle which is **not** in the system's graphic device list locked. For example, a GDHandle created by NewGWorld is unlocked.

If your program needs to lock any GDHandle, the easiest and safest thing to do is save and restore the state of the GDHandle in order to work properly, regardless of whether the GDHandle is in the system's graphic device list.

Note:
This solution does not solve the problem completely with respect to binaries that have already been deployed. If a GDHandle is ever unlocked by an existing binary program, the potential for a crash begins immediately because system code, such as NQDStdText, will not be altered to lock a GDHandle before accessing it. This solution only keeps new programs from causing the problem and keeps them from crashing if they ever encounter an unexpectedly unlocked GDHandle in the system's graphic device list. The system, however, may still crash when it encounters such a GDHandle.

An exhaustive search of Apple's previous documentation produced no clear directions for when to use HUnlock and when to use HGetState and HSetState. As a result, DTS has produced a separate Technote treating this issue in detail.

User Solution

The previous version of this Technote claimed Apple would be releasing a system extension that masks this problem. Since this Technote's publication, however, Apple has chosen not to release this extension.

Back to top

Summary

No GDHandle in the system's graphic device list should ever be unlocked. Some programs, both from Apple and from third parties, unlock at least one GDHandle in the system's graphic device list. This can cause a crash. Developers should preserve the state of any GDHandle in the system's graphic device list, which needs to be locked by calling HGetState and HSetState. There is no solution users can apply.

Back to top

References

The Memory Manager chapter in *Inside Macintosh: Memory*

The Graphics Devices chapter in *Inside Macintosh: Imaging with QuickDraw*

Technote 1122: "Locking and Unlocking Handles"

Understanding the Mercutio-GDevice Problem

Back to top

Change History

06-February-1998	First published.
01-June-1998	Updated to remove the description of the user-level solution.

Back to top

Downloadables

Acrobat version of this Note (K). Download