Subject: Fatal Upp Core memory management (heap/malloc) interventions in AppKit/Cocoa?
Posted by fudadmin on Sun, 16 Jan 2011 19:53:07 GMT
View Forum Message <> Reply to Message

While working with various upp Cocoa things I became very annoyed with my apps randomly/constantly crashing. Especially when custom menu is enabled. With a message:

which according to some sources:
Quote: is most likely to occur as a result of over-releasing or premature release of an object.

showing any of your code) is mostly useless. The debugger is equally useless as it has no way of identifying the dealloced object.
I've spent many hours during last 2 weeks trying to figure that out. At first I thought it was my Objective-C code at fault. But then, following my instincts, I tried the same apps without Upp::Core and they ran smoothly!
Because, I guess, without those top 4 lines intervention:

#0 0x1000110af in Upp::Heap::DivideBlock --crash
#1 0x100010eb4 in Upp::Heap::LAlloc
#2 0x100011bf3 in Upp::Heap::Alloc
#3 0x10000e5b7 in operator new
#4 0x7fff829a48d7 in HIMenuBarView::Construct
#5 0x7fff829a221f in HIView::EventHandler
#6 0x7fff8298d22b in HIObject::Construct
#7 0x7fff8298cdd9 in HIObject::Create
#8 0x7fff8298ccc4 in HIObjectCreate
#9 0x7fff829a469a in HIMenuBarView::Create
#10 0x7fff829a3f48 in HIMenuBarFrameView::Initialize
#11 0x7fff8299204b in HIObject::HandleClassHIObjectEvent
#12 0x7fff82991f19 in HIObject::EventHook
#13 0x7fff82991997 in DispatchEventToHandlers
#14 0x7fff82990ee6 in SendEventToEventTargetInternal
#15 0x7fff82990d57 in SendEventToEventTargetWithOptions
#16 0x7fff8298cdfa in HIObject::Create
#17 0x7fff8298ccc4 in HIObjectCreate
#18 0x7fff829a1d09 in NewWindowCommon
#19 0x7fff829a181f in _HIWindowCreateWithCGWindow
#20 0x7fff829a0aec in _GetMenuBarWindow
#21 0x7fff829a094c in GetMenuBarView
#22 0x7fff829a15d4 in MBarMenuRgn
#23 0x7fff829a1406 in ShowBar
#24 0x7fff829a10d0 in SetSystemUIMode
#25 0x7fff863d57d5 in -[NSApplication finishLaunching]
#26 0x7fff863d5350 in -[NSApplication run]
#27 0x10000104f in main at main.mm:51

The question is: how to eliminate/prevent or to work THAT (mostly genious) memory intervention to our/upp advantage?

Possible solutions:

1. Compiling conditions:

a) I tried to change in Core
#if defined(UPP_HEAP) && !defined(__OBJC__)
#include <new>
....

but HIObject and other things in Cocoa might be __cplusplus__, I guess. Hence,
no help.

b) make #include cocoa
and then #include Core/core.h
nut we need something from AppKit inside Core.h? ...

2. Upp flag USE_MALLOC - also crash...

3. ...?

Any ideas, corrections for my thinking?

Subject: Re: Fatal Upp Core memory management (heap/malloc) interventions in AppKit/Cocoa?
Posted by mirek on Sun, 16 Jan 2011 22:33:44 GMT
View Forum Message <> Reply to Message

fudadmin wrote on Sun, 16 January 2011 14:53

2. Upp flag USE_MALLOC - also crash...

It is USEMALLOC....

Anyway, I am not quite sure what is the core of the problem.

I have thought that cocoa in fact is not C++ so it should not use 'new'?!

What is 'HIObject' ? C++ class?

(Sorry for my ignorance).

Subject: Re: Fatal Upp Core memory management (heap/malloc) interventions in AppKit/Cocoa?
Posted by fudadmin on Sun, 16 Jan 2011 23:53:54 GMT
View Forum Message <> Reply to Message

thanks for quick response!
mirek wrote on Sun, 16 January 2011 22:33
What is 'HIObject' ? C++ class?

C++ (more likely) or just C (OO?) ? But.. from wikipedia:

building GUIs. This is available in Mac OS X v10.2 or later, and gives Carbon programmers some of the tools that Cocoa developers have long been familiar with. Starting with Mac OS X v10.2, HIObject is the base class for all GUI elements in Carbon. HIView is supported by Interface Builder, part of Apple's developer tools. Traditionally GUI architectures of this sort have been left to third-party application frameworks to provide. Starting with Mac OS X v10.4, HIObjects are NSObjects and inherit the ability to be serialized into data streams for transport or saving to disk.

screen. HITheme was introduced in Mac OS X v10.3, and Appearance Manager is a compatibility layer on top of HITheme since that version.

Mac OS X v10.2, all controls are HIViews. In Mac OS X v10.4, the Control Manager was renamed HIView Manager.

Since Mac OS X v10.2, windows have a root HIView.

v10.2, menus are HIObjects. Since Mac OS X v10.3, menu content may be drawn using HIViews, and all standard menus use HIViews to draw.

My strong suspicion is that HiObject and HIToolbox etc is a bridge for Cocoa to communicate with old but still in use Carbon (edit: or whatever is below Cocoa) functions.

Subject: Re: Fatal Upp Core memory management (heap/malloc) interventions in

AppKit/Cocoa?
Posted by fudadmin on Mon, 17 Jan 2011 01:52:56 GMT
View Forum Message <> Reply to Message

More info:
Quote:Some Managers, such as the HIView Manager (a superset of the Control Manager), are implemented in C++, but Carbon remains a C API.

Also, some crazy ideas...
1. to put #ifdef checking if class/object had derived from HIObject and neutralize Upp new overriding.. ?
maybe less crazy - 2. just block Upp code "travelling" into apple side? But how to separate? ... see 1 - above.

Subject: Re: Fatal Upp Core memory management (heap/malloc) interventions in AppKit/Cocoa?
Posted by fudadmin on Mon, 17 Jan 2011 03:05:36 GMT
View Forum Message <> Reply to Message

maybe this link could help (HIObject.h)
http://www.koders.com/c/fidFB6D22D4A9C7802739050F99BFAD01A63C717085.aspx?s=%22alang%22

So, I just tried quickly #if defined(UPP_HEAP) && !defined(__HIOBJECT__) but...

Subject: Re: Fatal Upp Core memory management (heap/malloc) interventions in AppKit/Cocoa?
Posted by mirek on Mon, 17 Jan 2011 09:32:52 GMT
View Forum Message <> Reply to Message

I guess, start with "USEMALLOC"....

This certainly will be problem in the future. Well, sort of - U++ will not be as fast as on other platforms... But whatever.

Subject: Re: Fatal Upp Core memory management (heap/malloc) interventions in AppKit/Cocoa?
Posted by fudadmin on Mon, 17 Jan 2011 15:27:15 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 17 January 2011 09:32I guess, start with "USEMALLOC"....

This certainly will be problem in the future. Well, sort of - U++ will not be as fast as on other platforms... But whatever.

I don't like slow software.... but for time being I will try to live with that.
The situation was that I had tried to use "USEMALLOC". But... Now I have found the reason - my Xcode configs were recompiling Upp Core correctly but linking from the wrong location.

On the other hand, with multithreading "USEMALLOC" in used anyway?

Regarding Upp::Heap I sniff a slim possibility to use custom allocation with CFAllocator but that's for the future.

ok, back to porting work.

Subject: Re: Fatal Upp Core memory management (heap/malloc) interventions in AppKit/Cocoa?
Posted by mirek on Mon, 17 Jan 2011 15:57:52 GMT
View Forum Message <> Reply to Message

fudadmin wrote on Mon, 17 January 2011 10:27
On the other hand, with multithreading "USEMALLOC" in used anyway?

USEMALLOC has nothing to do with MT.

It just tells U++ not to override new/delete and to use malloc/free for allocations.

Subject: Re: Fatal Upp Core memory management (heap/malloc) interventions in AppKit/Cocoa?
Posted by fudadmin on Mon, 17 Jan 2011 16:16:11 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 17 January 2011 15:57fudadmin wrote on Mon, 17 January 2011 10:27
On the other hand, with multithreading "USEMALLOC" in used anyway?

USEMALLOC has nothing to do with MT.

It just tells U++ not to override new/delete and to use malloc/free for allocations.

sorry, I had mistaken "&&" for "||"
```
#if defined(flagMT)
 #if defined(PLATFORM_WIN32) && defined(COMPILER_GCC)
  #define flagUSEMALLOC //MINGW does not support
 #endif
#endif
```

Subject: Re: Fatal Upp Core memory management (heap/malloc) interventions in AppKit/Cocoa?
Posted by daveremba on Sun, 19 Jun 2011 22:50:55 GMT
View Forum Message <> Reply to Message

I encountered a similar problem.
Not a crash, but a leak detected in UPP/heapdbg.cpp
I get "Heap leaks detected!" on exit.

Here is what I found:

a stack trace showed a MacOSX carbon function
from Xft... calling
UPP Core/operator new()!
maybe the order of constructors in UPP
is calling Xft before it is initialized?

Here is a temp fix: in Draw/Font.cpp:
(telling UPP heap debugger to ignore this leak)

```
const CommonFontInfo& Font::Fi() const
{
// add:
 MemoryIgnoreLeaksBlock __;

 if(lastStdFont != AStdFont.AsInt64()) {
  lastFiFont = INT_MIN;
  lastStdFont = AStdFont.AsInt64();
 }
 if(AsInt64() == lastFiFont)
  return lastFontInfo;
// known leak on MacOSX here: getAllCarbonLazyValues2000 calls Core.h op new()
// should not call UPP op new()
// from GetFontInfo() ... XftFontOpenPattern() ... getAllCarbonLazyValues2000() -> new()
 lastFontInfo = GetFontInfo(*this);
```

```
 lastFiFont = AsInt64();
 return lastFontInfo;
}
```

A stack trace from gdb is attached to this message.
(stack frames 0-5 are from a temp gets() to force a halt
while gdb is attached to the process)

I haven't tried debugging UPP in UPP yet!

I think the better fix is to include some Xft header
in Font.cpp after Core.h ?
Core/Core.h:201:inline void *operator new(size_t size) throw(std::bad_alloc) { void *ptr =
UPP::MemoryAlloc(size); return ptr; }

Dave

File Attachments

1) stack_trace_at_996.txt, downloaded 619 times

Subject: Re: Fatal Upp Core memory management (heap/malloc) interventions in AppKit/Cocoa?
Posted by mirek on Mon, 20 Jun 2011 03:58:18 GMT
View Forum Message <> Reply to Message

daveremba wrote on Sun, 19 June 2011 18:50I encountered a similar problem.
Not a crash, but a leak detected in UPP/heapdbg.cpp
I get "Heap leaks detected!" on exit.

Here is what I found:

a stack trace showed a MacOSX carbon function
from Xft... calling
UPP Core/operator new()!
maybe the order of constructors in UPP
is calling Xft before it is initialized?

Here is a temp fix: in Draw/Font.cpp:
(telling UPP heap debugger to ignore this leak)

```
const CommonFontInfo& Font::Fi() const
{
```

```
// add:
 MemoryIgnoreLeaksBlock __;

 if(lastStdFont != AStdFont.AsInt64()) {
  lastFiFont = INT_MIN;
  lastStdFont = AStdFont.AsInt64();
 }
 if(AsInt64() == lastFiFont)
  return lastFontInfo;
// known leak on MacOSX here: getAllCarbonLazyValues2000 calls Core.h op new()
// should not call UPP op new()
// from GetFontInfo() ... XftFontOpenPattern() ... getAllCarbonLazyValues2000() -> new()
 lastFontInfo = GetFontInfo(*this);
 lastFiFont = AsInt64();
 return lastFontInfo;
}
```

A stack trace from gdb is attached to this message.
(stack frames 0-5 are from a temp gets() to force a halt
while gdb is attached to the process)

I haven't tried debugging UPP in UPP yet!

Well, it looks like they do not care about releasing memory for global object allocations... In that case, your fix is absolutely correct. Meantime, we have to hope they always call our new/delete If not, there will have to be implicit USEMALLOC for MacOSX....

Quote:
I think the better fix is to include some Xft header
in Font.cpp after Core.h ?
Core/Core.h:201:inline void *operator new(size_t size) throw(std::bad_alloc) { void *ptr =
UPP::MemoryAlloc(size); return ptr; }

Not quite sure how that is going to help?

Mirek