Subject: Heap Leaks on Linux when migrating to new distro and newer UPP
Posted by jfranks on Thu, 27 Aug 2015 19:06:09 GMT
View Forum Message <> Reply to Message

We have a mature application that has been developed over the years and running on upp-2007.1 and Ubuntu distro 9.10

We've decided to migrate to a new Linux distribution, i.e., Linux Mint 17.2
Also, we migrated to the UPP snap shot upp-x11-src-8760.

We have been successful at compiling and linking the application after making some minor code changes to satisfy the new compile environment and UPP.

Everything works great, except when the application exits.
We get a dialog "Fatal error" with the message "Heap leaks detected!"
This goes away if we use the project flag USEMALLOC, but this is not the solution since it only hides a memory usage issue.

Ordinarily, development of a new program would have caught this problem early on. The problem is that we are now looking for a needle in the haystack because of the size of the program. Where do we look?

We've tried various memory diagnostic tools to no avail.
They give false positives, and provided no useful information.
We've tried valgrind and a few others with no relief.

On the other hand, we believe there are memory handling issues in our application that belong to us.
These have only been discovered (or caused by) the new environment: new compiler, linker, libraries, and UPP library.

We've tried compiling the UPP examples and tutorials and they behave properly on exit, as does the UPP theide.

What we are faced with now is dismembering the application piece at a time and re-compile/re-link/re-run until we discover the offending code.

What we are asking is that someone may have a better solution or alternative method for finding the real problem to our heap leak issue. Maybe there is something we can do to obtain more information that we don't know about.
We would welcome any help or suggestions on this matter.
Please let us know what you think.

regards,
-jlf

Subject: Re: Heap Leaks on Linux when migrating to new distro and newer UPP
Posted by dolik.rce on Thu, 27 Aug 2015 19:52:26 GMT
View Forum Message <> Reply to Message

Hi jfranks,

See this message: http://www.ultimatepp.org/forums/index.php?t=msg&th=3285&goto=14822&#msg_14822

First, if you get consistent values for --memory-breakpoint when running the application multiple times, then you can just add that as a parameter when running in debugger and it will show you where the offending piece of memmory was allocated.

Second method is to add MemmoryIgnoreLeaksBegin/End pairs around various parts of your application, until the warning stops. Then you know you found it... It is only slightly better than what you probably had in mind when you wrote about dismembering your application piece at a time, but it could save you some time and manual work

Best regards,
Honza

Subject: Re: Heap Leaks on Linux when migrating to new distro and newer UPP
Posted by jfranks on Sat, 24 Oct 2015 15:45:32 GMT
View Forum Message <> Reply to Message

We have made good progress at identifying and correcting the heap leaks in our application.
Additional tools were used in order to accomplish this, i.e., compile
with CLANG, and use asan.

The U++ snapshot that we are now using is upp-x11-src-9004.tar.gz

We are left with two heap leaks that we have yet to resolve.
Memory breakpoint 22748 is in our own application and we take ownership of that issue.

However, memory breakpoint 22235 is a real puzzle because it is almost exclusively is in U++ library.
Here is the log-file followed by a snapshot of the debugger call-stack:
==========
 /home/svn/P101/trunk/Source/Build/upp.out/P101/CLANG.Debug.Debug_Full.Gui.Mt.Newerupp.Noblitz.Shared.Stacktrace.X11/p101 21.10.2015 15:27:03, user: administrator

Heap leaks detected:

--memory-breakpoint__ 22748 : Memory at 0x0x7fd0c21bd9a0, size 0x58 = 88
   +0 0x00007FD0C21BD9A0 60 D6 48 01 00 00 00 00 00 00 00 00 00 00 00 00 `.H.............
  +16 0x00007FD0C21BD9B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....

```
   +32 0x00007FD0C21BD9C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
   +48 0x00007FD0C21BD9D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................

--memory-breakpoint__ 22235 : Memory at 0x0x7fd0ce6b7720, size 0x68 = 104
    +0 0x00007FD0CE6B7720 10 72 48 01 00 00 00 00 01 65 65 46 72 65 65    .rH.......eeFree
   +16 0x00007FD0CE6B7730 00 87 07 C2 D0 7F 00 00 14 00 00 00 01 00 00 00    .....•.........
   +32 0x00007FD0CE6B7740 60 00 00 00 00 00 00 00 00 10 00 00 00 00 00 00    `...............
   +48 0x00007FD0CE6B7750 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
****************** PANIC: Heap leaks detected!
====================
```

Can someone help on this particular issue?

File Attachments

1) memory-breakpoint-22235.jpg, downloaded 1181 times

Thread	Frame	Function	Arguments	Location
2	0	Upp::MemoryAlloc	(size = 96)	/home/administrator/upp/uppsrc/Core/he
1	1	operator new	(size = 96)	/home/administrator/upp/uppsrc/Core/he
	2	Upp::Image::Set	(this = 0x7fffffffde10, b =)	/home/administrator/upp/uppsrc/Draw/I
	3	Upp::Image::Image	(this = 0x7fffffffde10, b =)	/home/administrator/upp/uppsrc/Draw/I
	4	Upp::AdjustColors	(simg =)	/home/administrator/upp/uppsrc/Draw/C
	5	Upp::Override	(target = , source = , colored = true)	/home/administrator/upp/uppsrc/Draw/C
	6	Upp::ColoredOverride	(target = , source =)	/home/administrator/upp/uppsrc/Draw/C
	7	Upp::ChStdSkin	()	/home/administrator/upp/uppsrc/CtrlLib/
	8	Upp::Ctrl::ReSkin	()	/home/administrator/upp/uppsrc/CtrlCor
	9	Upp::Ctrl::InitX11	(display = 0x0)	/home/administrator/upp/uppsrc/CtrlCor
	10	main	(argc = 3, argv = 0x7fffffffe088, envptr = 0x1ac8570)	/home/svn/P101/trunk/Source/Applicatio

---

Subject: Re: Heap Leaks on Linux when migrating to new distro and newer UPP
Posted by mirek on Sun, 25 Oct 2015 07:22:36 GMT
View Forum Message <> Reply to Message

The fact that it was allocated from U++ does not in this case mean that the leak is caused by U++.

This looks like the memory block is allocated when loading skin. ColoredOverride is called on CtrlsImg Images.

If you have an Image in your code that is on the heap and gets assing CtrlsImg:: image, and is leaked, you are about to get the leak you are getting. E.g.:

```
GUI_APP_MAIN {
    Image *m = new Image;
    *m = CtrlsImg::SizeGrip();
    // m is not deleted and is a leak
}
```
