Subject: crush of the program
Posted by forlano on Tue, 27 Feb 2007 20:50:47 GMT
View Forum Message <> Reply to Message

Hello,

this is not related to U++ problem but it is quite general. I got the same problem in the past when programming without U++.

In MSC8 debug mode the program (with a U++ GUI)performs some tasks as expected.
But when I build a release mode the program crush at some moment. I have confined the problem in some part of the code but apparently nothing is wrong.
However if in release Optimal mode I put in the code a line like

Exclamation("I'm here");

something happen and the crush disappear. I wonder how such simple line can modify the code.
Does anybody had a similar experience and knows by what it can depend?

Luigi

Subject: Re: crush of the program
Posted by mr_ped on Tue, 27 Feb 2007 22:09:48 GMT
View Forum Message <> Reply to Message

Yes, it's quite common with larger game projects (maybe even with non-game project, but I have mostly experience with computer games).

The reasons may vary horribly.
- compiler error - very very very unlikely, but I did have seen already some. Search everything else 10 times before you start to search for this one.
- usage of uninitialized/corrupted memory - in debug mode you get all kind of those helpers like 0xCCCCCCCC for allocated memory or 0xFDFDFDFD for deleted one, same allocation addresses, etc... in release you get random garbage and never really knows what to expect. Also allocated memory in debug mode has some guardians space which may catch occasional memory overruns, in release it's much easier to corrupt your memory.
If your bug in code works nicely with for example 0xCC.. value, so you don't notice it, it may go crazy in release with the random values it will hit.
- race conditions in multi-threaded apps. (very hard to debug)
- bad definitions of variables (bad usage of volatile variables)
- and many many more...

Do you have some idea which part of code is crashing?
May you publish it here?
Maybe I will have some idea (and maybe not, those bugs are sometimes extremely difficult to catch, depends on overall quality of code and available resources).

Also if you manage to get stack/memory/code dump of crash, it may be worth to compare it with symbol table to see if it does crash always on the same place, and examine the exact reason of crash.

Subject: Re: crush of the program
Posted by exolon on Wed, 28 Feb 2007 00:48:30 GMT
View Forum Message <> Reply to Message

mr_ped wrote on Tue, 27 February 2007 22:09- race conditions in multi-threaded apps. (very hard to debug)
Yes... I had a very frustrating bug when working on a cryptography assignment done in U++ last week (btw, I got 90%... surely thanks to the GUI, so thank you U++... DrawingDraw and Timer made things really nice).

I noticed that after adding more and more stuff to my main window (TopWindow subclass), the program would crash on startup before the window was displayed.

It got to the stage where adding another Label to the class's member variables caused the program to crash, but commenting out that Label declaration didn't. Which was quite mysterious - even gdb crashed when I tried to debug the error, so I thought maybe the code was running out of stack and maybe the objects created by U++ were huge or something.

Eventually, I realised that the extra class member initialisations were taking long enough that the TopWindow subclass was preempted before it executed its constructor (!) and one of the member variable objects started running instead, which accessed a shared global variable (ack!) that wasn't initialised yet - I intended to increment this shared variable with a timer callback and use it to coordinate graphics operations.

Anyway... long story long, I'd check shared variables and code that might crash if it runs before other code you expect would normally run first. I found the problem by putting PromptOk() almost everywhere, but it's tougher for you since it makes the problem go away! Have you tried moving that line earlier and later to see if it can at least help pinpoint better (ie: here it crashes, here it doesn't crash)...

Subject: Re: crush of the program
Posted by forlano on Wed, 28 Feb 2007 01:53:43 GMT
View Forum Message <> Reply to Message

Thanks for your answers.
After doing some movement of the code the problem disappeared. But I am afraid it will come back again to ruin my dreams.
A very mysterious bug...

Luigi

Subject: Re: crush of the program
Posted by mirek on Wed, 28 Feb 2007 08:25:39 GMT
View Forum Message <> Reply to Message

mr_ped wrote on Tue, 27 February 2007 17:09
- usage of uninitialized/corrupted memory - in debug mode you get all kind of those helpers like 0xCCCCCCCC for allocated memory or 0xFDFDFDFD for deleted one, same allocation addresses, etc... in release you get random garbage and never really knows what to expect. Also allocated memory in debug mode has some guardians space which may catch occasional memory overruns, in release it's much easier to corrupt your memory.

BTW, the most troublesome bug is "read past end of buffer". While there is a huge chance that U++ heap allocator catches writes, reads, especially one byte past end, are impossible to catch. Plus, the chance that you read byte in area that causes exception is very very low. Means it crashes once a week or so.

Once I was dealing with mysterious crashes of one of my application for 6 months, before identifying this.

Quote:
Also if you manage to get stack/memory/code dump of crash, it may be worth to compare it with symbol table to see if it does crash always on the same place, and examine the exact reason of crash.

In U++/Win32 you can call "InstallCrashDump" at the start of program. This will create the core dump that can be later analyzed in "Crash" utility (you should be able to compile it from uppsrc). So if you have any difficulity analyzing, you can try this. Crash will need "*.crash" dump file and map file of executable.

Mirek

Subject: Re: crush of the program
Posted by mirek on Wed, 28 Feb 2007 08:27:26 GMT
View Forum Message <> Reply to Message

forlano wrote on Tue, 27 February 2007 20:53Thanks for your answers.
After doing some movement of the code the problem disappeared. But I am afraid it will come back again to ruin my dreams.
A very mysterious bug...

Well, this is not solution. I strongly recommend to go after it. Return the code to "crashing" status, use InstallCrashDump and resolve the issue.

Mirek

Subject: Re: crush of the program
Posted by mr_ped on Wed, 28 Feb 2007 10:55:57 GMT
View Forum Message <> Reply to Message

@Mirek:
that tip about InstallCrashDump should be somewhere in examples or something.. maybe whole Example about it.
IMHO it's absolute killer feature (no matter how much of it is part of upp and how much it's "standard") if it works like you described.
(didn't try it yet by myself, but surely I will, sounds very promising ... deciphering the crash info by hand is always tedious, so any help is welcome)

Also as I'm right now evaluating development tools under linux in my spare time, I see "valgrind" quite often, haven't checked it yet because I'm busy with other things, but some built-in support in TheIDE with nice GUI would be maybe worth of it (and it would help lazy people like me). (if the tool itself is worth of it, but it looks good on first sight)

Thank you.

Subject: Re: crush of the program
Posted by mirek on Wed, 28 Feb 2007 11:06:56 GMT
View Forum Message <> Reply to Message

mr_ped wrote on Wed, 28 February 2007 05:55@Mirek:
that tip about InstallCrashDump should be somewhere in examples or something.. maybe whole Example about it.
IMHO it's absolute killer feature (no matter how much of it is part of upp and how much it's "standard") if it works like you described.
(didn't try it yet by myself, but surely I will, sounds very promising ... deciphering the crash info by hand is always tedious, so any help is welcome)

Unortunately, right now it is limited to Win32/MSC. As many things in U++, it was created to solve the problem... and needs further refining (adding GCC support etc...) to be commonly useful.

Subject: Re: crush of the program
Posted by forlano on Wed, 28 Feb 2007 14:55:17 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 28 February 2007 09:27forlano wrote on Tue, 27 February 2007 20:53Thanks for your answers.

After doing some movement of the code the problem disappeared. But I am afraid it will come back again to ruin my dreams.
A very mysterious bug...

Well, this is not solution. I strongly recommend to go after it. Return the code to "crashing" status, use InstallCrashDump and resolve the issue.

Mirek

"Unfortunately" the bug disappeared and I have not the previous crashing code
Anyway, as mr_ped said, this important tool should be mentioned somewhere in the documentation even if it exists only for windows.

Thanks,
Luigi

Subject: Re: crush of the program
Posted by forlano on Thu, 22 Mar 2007 13:03:10 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 28 February 2007 09:25

In U++/Win32 you can call "InstallCrashDump" at the start of program. This will create the core dump that can be later analyzed in "Crash" utility (you should be able to compile it from uppsrc). So if you have any difficulity analyzing, you can try this. Crash will need "*.crash" dump file and map file of executable.

Mirek

Hello,

I'm having the same problem for linux (crushes in optimal mode and works in debug mode). Is there some support for linux? I've tried to add "InstallCrashDump" (<Core/Diag.h>) but I get error from gcc. It seems this is very specific Win stuff.

Luigi

Subject: Re: crush of the program
Posted by mirek on Thu, 22 Mar 2007 16:22:56 GMT
View Forum Message <> Reply to Message

Yes, this is win specific (yet?)

Mirek

Subject: Re: crush of the program
Posted by exolon on Wed, 28 Mar 2007 19:58:21 GMT
View Forum Message <> Reply to Message

BTW, a quick note which may be of help to those debugging these kind of problems - on linux, you can use a program called valgrind which will give a report like this:

```
==18964== Conditional jump or move depends on uninitialised value(s)
==18964==    at 0x80947F0: (within /home/oisin/upp/out/GCC32.Gui.Shared/AnimatedHello)
==18964==    by 0x8094DEB: (within /home/oisin/upp/out/GCC32.Gui.Shared/AnimatedHello)
==18964==    by 0x809C7AC: (within /home/oisin/upp/out/GCC32.Gui.Shared/AnimatedHello)
==18964==    by 0x809CCF8: (within /home/oisin/upp/out/GCC32.Gui.Shared/AnimatedHello)
==18964==    by 0x809A004: (within /home/oisin/upp/out/GCC32.Gui.Shared/AnimatedHello)
==18964==    by 0x809A592: (within /home/oisin/upp/out/GCC32.Gui.Shared/AnimatedHello)
==18964==    by 0x8098C19: (within /home/oisin/upp/out/GCC32.Gui.Shared/AnimatedHello)
==18964==    by 0x804CD68: (within /home/oisin/upp/out/GCC32.Gui.Shared/AnimatedHello)
==18964==    by 0x804CDB6: (within /home/oisin/upp/out/GCC32.Gui.Shared/AnimatedHello)
==18964==    by 0x4294EA1: __libc_start_main (in /lib/tls/i686/cmov/libc-2.3.6.so)
```

(many times, then, after a long while... for me, about 3 minutes (!) program appears, I let it run for a moment and quit)

```
==18964==
==18964== ERROR SUMMARY: 100 errors from 19 contexts (suppressed: 39 from 1)
==18964== malloc/free: in use at exit: 2,144,858 bytes in 30,435 blocks.
==18964== malloc/free: 492,498 allocs, 462,063 frees, 51,400,138 bytes allocated.
==18964== For counts of detected errors, rerun with: -v
==18964== searching for pointers to 30,435 not-freed blocks.
==18964== checked 2,146,892 bytes.
==18964==
==18964== LEAK SUMMARY:
==18964==    definitely lost: 697 bytes in 42 blocks.
==18964==      possibly lost: 543,752 bytes in 90 blocks.
==18964==    still reachable: 1,600,409 bytes in 30,303 blocks.
==18964==         suppressed: 0 bytes in 0 blocks.
==18964== Use --leak-check=full to see details of leaked memory.
```

It takes a huge amount of time though - for me running on a not too fast Celeron 2ghz or so in Ubuntu linux, loading AnimatedHello normally in Shared libs mode took about 10 seconds...

Apparently gcc has a memory debugging library called mudflap, but it seems to generate a lot of spurious errors in C++.

Subject: Re: crush of the program

Posted by Novo on Wed, 28 Mar 2007 20:08:19 GMT
View Forum Message <> Reply to Message

exolon wrote on Wed, 28 March 2007 15:58BTW, a quick note which may be of help to those debugging these kind of problems - on linux, you can use a program called valgrind which will give a report like this:

valkyrie (a GUI client for valgrind) makes that report much more readable especially for people who got used to "purify" from Rational Software.

Subject: Re: crush of the program
Posted by exolon on Wed, 28 Mar 2007 20:20:04 GMT
View Forum Message <> Reply to Message

Sounds great... atm I'm using a very rudimentary front-end called Alleyoop which isn't so good. Thanks Novo, I'll check it out.

[edit]Some GUIs listed on the valgrind site.

Subject: Re: crush of the program
Posted by forlano on Wed, 28 Mar 2007 21:15:51 GMT
View Forum Message <> Reply to Message

exolon wrote on Wed, 28 March 2007 21:58BTW, a quick note which may be of help to those debugging these kind of problems - on linux, you can use a program called valgrind which will give a report like this:...

Good to know. Thanks!
Luigi

Subject: Re: crush of the program
Posted by zsolt on Wed, 28 Mar 2007 21:45:05 GMT
View Forum Message <> Reply to Message

Novo wrote on Wed, 28 March 2007 22:08exolon wrote on Wed, 28 March 2007 15:58BTW, a quick note which may be of help to those debugging these kind of problems - on linux, you can use a program called valgrind which will give a report like this:

valkyrie (a GUI client for valgrind) makes that report much more readable especially for people who got used to "purify" from Rational Software.

I found kcachegrind very useful as a valgrind frontend.

Subject: Re: crush of the program
Posted by Novo on Thu, 29 Mar 2007 14:51:35 GMT
View Forum Message <> Reply to Message

zsolt wrote on Wed, 28 March 2007 17:45Novo wrote on Wed, 28 March 2007 22:08exolon wrote on Wed, 28 March 2007 15:58BTW, a quick note which may be of help to those debugging these kind of problems - on linux, you can use a program called valgrind which will give a report like this:

valkyrie (a GUI client for valgrind) makes that report much more readable especially for people who got used to "purify" from Rational Software.

I found kcachegrind very useful as a valgrind frontend.

kcachegrind is a frontend for cachegrind, which is a profiler. In terms of Rational Software it is "quantify".

By default valgrind runs memcheck (valgrind --tool=memcheck).
cachegrind is a part of valgrind starting from version 3.2 I believe.

Subject: Re: crush of the program
Posted by mirek on Thu, 29 Mar 2007 15:49:11 GMT
View Forum Message <> Reply to Message

Novo wrote on Thu, 29 March 2007 10:51
By default valgrind runs memcheck (valgrind --tool=memcheck).
cachegrind is a part of valgrind starting from version 3.2 I believe.

Not sure what "memcheck" does, just wanted to say U++ has quite strong heap checking in debug mode

All blocks have safety sentinels (overwrite is reported), free blocks memory is filled with specific pattern and tested for writes when allocating again and leaks are not tolerated (error at the program exit).

Mirek

Subject: Re: crush of the program
Posted by Novo on Thu, 29 Mar 2007 18:24:54 GMT
View Forum Message <> Reply to Message

luzr wrote on Thu, 29 March 2007 11:49Novo wrote on Thu, 29 March 2007 10:51
By default valgrind runs memcheck (valgrind --tool=memcheck).
cachegrind is a part of valgrind starting from version 3.2 I believe.

Not sure what "memcheck" does, just wanted to say U++ has quite strong heap checking in debug mode

All blocks have safety sentinels (overwrite is reported), free blocks memory is filled with specific pattern and tested for writes when allocating again and leaks are not tolerated (error at the program exit).

Mirek

Quote:
Memcheck can detect if your program:

* Accesses memory it shouldn't (areas not yet allocated, areas that have been freed, areas past the end of heap blocks, inaccessible areas of the stack).
* Uses uninitialised values in dangerous ways.
* Leaks memory.
* Does bad frees of heap blocks (double frees, mismatched frees).
* Passes overlapping source and destination memory blocks to memcpy() and related functions.

That can be done in Debug and Release mode.
I recall somebody's messages about spending dozen of hours looking for a bug.
With valgrind that can be done in few minutes.

Subject: Re: crush of the program
Posted by forlano on Mon, 28 May 2007 16:26:47 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 28 February 2007 09:25
In U++/Win32 you can call "InstallCrashDump" at the start of program. This will create the core dump that can be later analyzed in "Crash" utility (you should be able to compile it from uppsrc). So if you have any difficulity analyzing, you can try this. Crash will need "*.crash" dump file and map file of executable.

Mirek

Hello,

that terrible old bug apperead again. With InstallCrashDump() I got a core dump file of 31 Kb. Now I need to analyze it with the Crush utility. Unfortunately I cannot found it! Where is it? Is it a package? I'm using the stable 2007.1.

Thanks,
Luigi

Subject: Re: crush of the program
Posted by forlano on Tue, 29 May 2007 21:38:14 GMT
View Forum Message <> Reply to Message

forlano wrote on Mon, 28 May 2007 18:26luzr wrote on Wed, 28 February 2007 09:25
In U++/Win32 you can call "InstallCrashDump" at the start of program. This will create the core dump that can be later analyzed in "Crash" utility (you should be able to compile it from uppsrc). So if you have any difficulity analyzing, you can try this. Crash will need "*.crash" dump file and map file of executable.

Mirek

Hello,

that terrible old bug apperead again. With InstallCrashDump() I got a core dump file of 31 Kb. Now I need to analyze it with the Crush utility. Unfortunately I cannot found it! Where is it? Is it a package? I'm using the stable 2007.1.

Thanks,
Luigi

I found it in uvs.
