Subject: Strange behavior of Point in watches
Posted by dolik.rce on Tue, 25 Aug 2009 07:42:27 GMT
View Forum Message <> Reply to Message

Hello,
I've encountered strange problem while debugging my code... Here is simplest possible testcase:

```
#include <CtrlLib/CtrlLib.h>
using namespace Upp;

class win : public TopWindow{
public:
 typedef win CLASSNAME;
 virtual void LeftDown(Point p,dword flags){
  DUMP(p);
 }
};

GUI_APP_MAIN{
 win().Run();
}
```

I setup breakpoint in LeftDown() and run in debug mode. When the execution stops at the breakpoint, opening Quick watch window and entering "p" yields:

```
{
  <Upp::Moveable<Upp::Point_<int>, Upp::EmptyClass>> = {
    <Upp::EmptyClass> = {<No data fields>}, <No data fields>},
  members of Upp::Point_<int>:
  x = -1078378568,
  y = 8388608
}
```

Same values are shown if I add watch in the bottom panel of theide. But the most interesting (or puzzling) thing is, that the output in log file is correct, i.e. something like "p = [59, 138]".

At first, I thought that it is a problem in watches, but for something like "Point P(10,20);" watches show correct result. It can be very confusing, I was looking for bug almost an hour on absolutely wrong place, because of this...

Just in case this is compiler/platform specific: I use gcc 4.3 on ubuntu.

Regards,
Honza

PS: I'm not sure if this belongs here. If not, feel free to move this topic to some better place.

Subject: Re: Strange behavior of Point in watches
Posted by mirek on Tue, 25 Aug 2009 08:07:24 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Tue, 25 August 2009 03:42Hello, I've encountered strange problem while debugging my code... Here is simplest possible testcase:#include <CtrlLib/CtrlLib.h>

```
using namespace Upp;

class win : public TopWindow{
public:
 typedef win CLASSNAME;
 virtual void LeftDown(Point p,dword flags){
  DUMP(p);
 }
};

GUI_APP_MAIN{
 win().Run();
}
```

I setup breakpoint in LeftDown() and run in debug mode. When the execution stops at the breakpoint, opening Quick watch window and entering "p" yields:

```
{
  <Upp::Moveable<Upp::Point_<int>, Upp::EmptyClass>> = {
    <Upp::EmptyClass> = {<No data fields>}, <No data fields>},
  members of Upp::Point_<int>:
  x = -1078378568,
  y = 8388608
}
```

Same values are shown if I add watch in the bottom panel of theide. But the most interesting (or puzzling) thing is, that the output in log file is correct, i.e. something like "p = [59, 138]".

At first, I thought that it is a problem in watches, but for something like "Point P(10,20);" watches show correct result. It can be very confusing, I was looking for bug almost an hour on absolutely wrong place, because of this...

Just in case this is compiler/platform specific: I use gcc 4.3 on ubuntu.

Regards,
Honza

PS: I'm not sure if this belongs here. If not, feel free to move this topic to some better place.

It is because correct stack frame for the function is not yet established - gdb reads incorrect values.

If you step inside, you they get corrected.

This is dbg feature, there is nothing we can do about it...

Mirek

Subject: Re: Strange behavior of Point in watches
Posted by dolik.rce on Tue, 25 Aug 2009 18:44:49 GMT
View Forum Message <> Reply to Message

luzr wrote on Tue, 25 August 2009 10:07It is because correct stack frame for the function is not yet established - gdb reads incorrect values.

If you step inside, you they get corrected.
Thanks for reply. I think I understand what you mean - basically that I asked for the value before entering the function (or it's stack frame, more precisly). BUT: The same problem happens even when stepping inside. Let's take longer function, like: virtual void LeftDown(Point p,dword flags){

```
  DUMP(p);
  p.Offset(10,10);
  Point pt=p;
  DUMP(pt);
  p=pt;
 }
```

Then at any point in the function p watches yield nonsense. Watches for pt are correct (after it's assignment of course). Does that mean that watching any parameter passed to the function might be incorrect? That would be really confusing, especially for people who doesn't know about it, like me

Honza

Subject: Re: Strange behavior of Point in watches
Posted by mirek on Wed, 26 Aug 2009 07:00:12 GMT
View Forum Message <> Reply to Message

Well, what we get from gdb, we display...

Maybe you could try in gdb alone to see if anything is wrong in theide?

Mirek

Subject: Re: Strange behavior of Point in watches
Posted by dolik.rce on Thu, 27 Aug 2009 03:06:15 GMT
View Forum Message <> Reply to Message

Hi Mirek,
Good news, theide is innocent It's all gdb's fault. It's fine to know, I just wish I knew before and did not learning about it "the hard way".

Anyway, I ran following minimalistic program in gdb:#include <Core/Core.h>

```
using namespace Upp;

void somefunction(Point p){
 Point pt=p;
 p=pt;
}

CONSOLE_APP_MAIN{
 Point p(10,20);
 somefunction(p);
}
```
And here is the gdb session:Quote:
```
GNU gdb 6.8-debian
Copyright (C) 2008 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu"...
(gdb) break ConsoleMainFn_()
Breakpoint 1 at 0x804a662: file /media/other/opt/uppsvn/MyApps/test/main.cpp, line 10.
(gdb) break somefunction(Upp::Point_<int>)
Breakpoint 2 at 0x804a63a: file /media/other/opt/uppsvn/MyApps/test/main.cpp, line 5.
(gdb) run
Starting program: /tmp/ptest
[Thread debugging using libthread_db enabled]
[New Thread 0xb7d776d0 (LWP 31862)]
[Switching to Thread 0xb7d776d0 (LWP 31862)]

Breakpoint 1, ConsoleMainFn_ () at /media/other/opt/uppsvn/MyApps/test/main.cpp:10
10  Point p(10,20);
(gdb) next
11  somefunction(p);
(gdb) print p
$1 = {<Upp::Moveable<Upp::Point_<int>, Upp::EmptyClass>> = {<Upp::EmptyClass> = {<No data
fields>}, <No data fields>}, x = 10, y = 20}
(gdb) print &p
$2 = (Point *) 0xbf85e0e0
(gdb) c
Continuing.

Breakpoint 2, somefunction (p=
     {<Upp::Moveable<Upp::Point_<int>, Upp::EmptyClass>> = {<Upp::EmptyClass> = {<No data
fields>}, <No data fields>}, x = -1081745192, y = -1081745184})
    at /media/other/opt/uppsvn/MyApps/test/main.cpp:5
5  Point pt=p;
(gdb) print p
$3 = {<Upp::Moveable<Upp::Point_<int>, Upp::EmptyClass>> = {<Upp::EmptyClass> = {<No data
fields>}, <No data fields>}, x = -1081745192, y = -1081745184}
(gdb) print &p
$4 = (Point *) 0xbf85e0c0
(gdb) print pt
$5 = {<Upp::Moveable<Upp::Point_<int>, Upp::EmptyClass>> = {<Upp::EmptyClass> = {<No data
fields>}, <No data fields>}, x = -1081745192, y = 3683889}
(gdb) print &pt
$6 = (Point *) 0xbf85e0b0
(gdb) next
6  p=pt;
(gdb) next
7 }
(gdb) print p
$7 = {<Upp::Moveable<Upp::Point_<int>, Upp::EmptyClass>> = {<Upp::EmptyClass> = {<No data fields>}, <No data fields>}, x = -1081745192, y = -1081745184}
(gdb) print &p
$8 = (Point *) 0xbf85e0c0
(gdb) print pt
$9 = {<Upp::Moveable<Upp::Point_<int>, Upp::EmptyClass>> = {<Upp::EmptyClass> = {<No data fields>}, <No data fields>}, x = 10, y = 20}
(gdb) print &pt
$10 = (Point *) 0xbf85e0b0
(gdb) c
Continuing.

Program exited normally.
(gdb) quit
```
As you can see, gdb probably has no idea where is Point p stored in memory. One interesting thing is, that every time I tried, the offset between addresses of p in ConsoleMainFn_ and somefunction were always 0x20. That's probably not a coincdence, unfortunately I couldn't pinpoint yet, why is it happening. For simple types and simple structs it works correctly. I'll try to investigate this bit more and if I find something I post it here.

Regards,
Honza

Subject: Re: Strange behavior of Point in watches
Posted by dolik.rce on Wed, 02 Sep 2009 02:47:29 GMT

View Forum Message <> Reply to Message

Helo Mirek!

Sorry it took me so long, but I was busy at work.

I did a little investigation and finally found exact piece of code that causes problems in gdb. I'm not sure what is the real reason, but all the troubles are caused by following
constructors:
```
Point_(const Point_<int>& pt) : x((T)pt.x), y((T)pt.y) {}
 Point_(const Point_<short>& pt) : x((T)pt.x), y((T)pt.y) {}
 Point_(const Point_<double>& pt) : x((T)pt.x), y((T)pt.y) {}
 Point_(const Point_<int64>& pt) : x((T)pt.x), y((T)pt.y) {}
```
If you replace those four lines by
```
template <class U>
 Point_(const Point_<U>& pt) : x((T)pt.x), y((T)pt.y) {}
```

watches show correct values. Do you think that this workaround is safe? To my best knowledge it should produce absolutely same results as original code and nobody should try passing there any non-scalar type, so it should not cause any harm... but I'm no expert when it comes to templates

Also, during the investigation I found that same problems apply to Size_ and Rect_. Since their anatomy is very similar, it should be possible to apply the same workaround.

Honza

Subject: Re: Strange behavior of Point in watches
Posted by mirek on Wed, 02 Sep 2009 15:59:14 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Tue, 01 September 2009 22:47Helo Mirek!

Sorry it took me so long, but I was busy at work.

I did a little investigation and finally found exact piece of code that causes problems in gdb. I'm not sure what is the real reason, but all the troubles are caused by following
constructors: Point_(const Point_<int>& pt) : x((T)pt.x), y((T)pt.y) {}
 Point_(const Point_<short>& pt) : x((T)pt.x), y((T)pt.y) {}
 Point_(const Point_<double>& pt) : x((T)pt.x), y((T)pt.y) {}
 Point_(const Point_<int64>& pt) : x((T)pt.x), y((T)pt.y) {}
If you replace those four lines by template <class U>
 Point_(const Point_<U>& pt) : x((T)pt.x), y((T)pt.y) {}
 watches show correct values. Do you think that this workaround is safe? To my best knowledge it should produce absolutely same results as original code and nobody should try passing there any non-scalar type, so it should not cause any harm... but I'm no expert when it comes to templates

Also, during the investigation I found that same problems apply to Size_ and Rect_. Since their anatomy is very similar, it should be possible to apply the same workaround.

Honza

I do not know... I really hate such workarounds for debugger problems...

Mirek

Subject: Re: Strange behavior of Point in watches
Posted by dolik.rce on Wed, 02 Sep 2009 17:17:22 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 02 September 2009 17:59I do not know... I really hate such workarounds for debugger problems...

Mirek

I fully understand that. The only proper solution is to fix it in gdb. I'll try to file it in their bugzilla and we'll see if they can fix it in few months/years

Honza

Subject: Re: Strange behavior of Point in watches
Posted by mirek on Wed, 02 Sep 2009 18:43:44 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Wed, 02 September 2009 13:17luzr wrote on Wed, 02 September 2009 17:59I do not know... I really hate such workarounds for debugger problems...

Mirek

I fully understand that. The only proper solution is to fix it in gdb. I'll try to file it in their bugzilla and we'll see if they can fix it in few months/years

Honza

Actually, the only proper solution is to develop own debugger. It is not THAT hard, once you have symbolic and line info - MSC++ in theide does just that. And there are libraries available to provide symbol info.... And, in fact, all we would need is to make generic platform independent debugging interface, rework MSC++ debugger into using it, then implement this for Posix..

But it quite a lot of work, unfortunately...

Subject: Re: Strange behavior of Point in watches
Posted by dolik.rce on Thu, 03 Sep 2009 08:19:05 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 02 September 2009 20:43Actually, the only proper solution is to develop own debugger. It is not THAT hard, once you have symbolic and line info - MSC++ in theide does just that. And there are libraries available to provide symbol info.... And, in fact, all we would need is to make generic platform independent debugging interface, rework MSC++ debugger into using it, then implement this for Posix..

But it quite a lot of work, unfortunately...

Well, it sounds like a good idea, but that would need some volunteers to do it, which will be hard to find, because it's not anything critical...

Anyway, I've got (pretty quick) response to the bug report:
Quote:This has been fixed somewhere since 6.8 (which is VERY old). If you desperately need to do this, I suggest grabbing/using a snapshot.

Just for your information, "VERY old" means released on March 27, 2008. That is quite long, but it is still listed everywhere on their website as current release. Version 7.0, which will probably behave correctly, should be released soon, but I think their schedule is bit slipping...

I believe that closes this topic (at least until someone decides to rewrite the debugging from scratch). As this is not critical problem (and I'm aware of it), I'll probably just wait till the new version propagates into ubuntu repositories.

Thanks for your patience,
Honza

Subject: Re: Strange behavior of Point in watches
Posted by mr_ped on Thu, 03 Sep 2009 09:11:14 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Thu, 03 September 2009 10:19... I'll probably just wait till the new version propagates into ubuntu repositories.

So looking forward for Ubuntu 10.04 or 10.10?

Subject: Re: Strange behavior of Point in watches
Posted by dolik.rce on Thu, 03 Sep 2009 17:00:28 GMT
View Forum Message <> Reply to Message

mr_ped wrote on Thu, 03 September 2009 11:11dolik.rce wrote on Thu, 03 September 2009 10:19... I'll probably just wait till the new version propagates into ubuntu repositories.

So looking forward for Ubuntu 10.04 or 10.10?

That's not very optimistic
If it takes so long to get into ubuntu, than I can always "steal" it from Debian unstable

---

Subject: Re: Strange behavior of Point in watches
Posted by mr_ped on Fri, 04 Sep 2009 06:32:57 GMT
View Forum Message <> Reply to Message

Well, the 9.10 is already quite frozen. Although this package will very likely reside in universe repository, so it may be frozen later, but I doubt gdb will release 7.0 soon enough.

At this point it's either 10.04 (in case they release it in next 1-2 months) or somebody putting it into PPA (personal repository at launchpad) for older Ubuntu and you finding it there + adding that PPA. Or 10.10 if they miss 10.04 freeze milestone.

Ubuntu is running ahead quite fast actually, so it's easy to miss the release with your SW, if you don't care and don't follow a good release schedule. Yet even at this frantic speed, when you *need* some SW, even fresh Ubuntu feels suddenly outdated. Weird, but that's how it works.