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