U++ - Feature #287
Support safe Value inspectors in GDB

05/17/2012 11:53 PM - Massimo Del Fedele

Status:	Approved	**Start date:**	05/17/2012
Priority:	High	**Due date:**	
Assignee:		**% Done:**	100%
Category:	Debugger	**Estimated time:**	0.00 hour
Target version:		**Spent time:**	0.00 hour

Description

Changing Core/Value.h from line 77 as following :

```
#ifdef _DEBUG
struct ValueMagic
{
   uint32 MAGIC;
   ValueMagic() { MAGIC = 0xdeadbeef; }
};
class Value : Moveable<Value>, public ValueMagic {
#else
class Value : Moveable<Value> {
#endif
public:
   class Void {
   protected:
      Atomic  refcount;
```

will allow safe value inspection in Gdb_MI2 module even for uninitialized value, showing correctly "<value not initialized>" instead of crashing the debugger.
Please don't change MAGIC member name nor value (or tell me !) because python inspectors rely on those.

Max

History

#1 - 05/20/2012 10:32 AM - Miroslav Fidler

- Assignee changed from Miroslav Fidler to Massimo Del Fedele

I do not believe this is a good solution - there will be situations where this does not work (e.g. repeated entry on stack frame or in Value constructor).

If this is to be done properly, I believe you should put magic into Value, then set at the END of Value constructor to magic value, then CLEAR at the beginning of Value destructor (to cleanup stack).

It still will be error-prone, with crash probability 1/2^32. Using longer and more random magic value would help there... (hate to say that, but perhaps 16bytes magic would be best, even if it will double the memory consumption in debug mode).

I also hope that you are somehow able to process Value even without 'magic'.

#2 - 05/20/2012 11:19 AM - Massimo Del Fedele

- Assignee changed from Massimo Del Fedele to Miroslav Fidler

Mhhh... I looked at Value, it has a bunch of constructors, most inlined, that's why I used the "simple" solution.
I could add the MAGIC initialization at the end of each, but it seems to me overkilling, just to enable the debugger feature, but if you mean it could be done.
Other solutions, doing it all from debugger, it's not feasible, imho, because Value is too complicated and because of custom data types.
Maybe an expert python coder could try it, but I think it's really not worth the effort, and if you change some Value's internal all work would be useless.

Anyways, GDB accesses value ONLY when stopped and, if you don't stop inside Value constructors (very unlikely...) you won't have problems.
Yes, the MAGIC not being reset could be a problem when re-entering stack frames.

The best would be to find another way to detect if Value is initialized fom GDB, but I guess it's not possible.

About MAGIC size.... well, it could be increased, but IMHO 1/2^32 is quite enough, you can get a crash over some millions of debugging sessions, I think we can accept it :-)

Please tell me if I should apply your solution, embedding MAGIC initialization on each constructor and inside destructor... and if you'd like to have MAGIC size increased to uint64.

#3 - 05/20/2012 01:45 PM - Miroslav Fidler

Massimo Del Fedele wrote:

> *Mhhh... I looked at Value, it has a bunch of constructors, most inlined, that's why I used the "simple" solution.*
> *I could add the MAGIC initialization at the end of each, but it seems to me overkilling, just to enable the debugger feature, but if you mean it could be done.*

Now this I do NOT understand. IMO, it should be exactly at single place - at the end of Value constructor. Otherwise you run into problems with partially constructed Values (when base class is constructed and Value itself not yet).

> *Anyways, GDB accesses value ONLY when stopped and, if you don't stop inside Value constructors (very unlikely...) you won't have problems.*

Sometimes you do.

> *About MAGIC size.... well, it could be increased, but IMHO 1/2^32 is quite enough, you can get a crash over some millions of debugging sessions, I think we can accept it :-)*

Thing is that sizeof(Value) in such case is likely to be 32 anyway... So 16 bytes magic is not a problem IMO.

> *Please tell me if I should apply your solution, embedding MAGIC initialization on each constructor and inside destructor...*

What you mean by "each"? IMO there is only one constructor for Value...

#4 - 05/20/2012 01:45 PM - Miroslav Fidler
- Assignee changed from Miroslav Fidler to Massimo Del Fedele

#5 - 05/20/2012 04:18 PM - Massimo Del Fedele

- Assignee changed from Massimo Del Fedele to Miroslav Fidler

> *What you mean by "each"? IMO there is only one constructor for Value...*

Hemmm... what are those ?

```
template <class T> Value(const T& value, VSMALL);
Value(const String& s) : data(s) {}
Value(const WString& s);
Value(const char *s) : data(s)   {}
Value(int i)                : data(i, INT_V, String::SPECIAL) {}
Value(int64 i)              : data(i, INT64_V, String::SPECIAL) {}
Value(double d)             : data(d, DOUBLE_V, String::SPECIAL) {}
Value(bool b)               : data(b, BOOL_V, String::SPECIAL) {}
Value(Date d)               : data(d, DATE_V, String::SPECIAL) {}
Value(Time t)               : data(t, TIME_V, String::SPECIAL) {}
Value(const Nuller&)        : data((int)Null, INT_V, String::SPECIAL) {}
Value(const Value& v);
```

#6 - 05/20/2012 07:09 PM - Massimo Del Fedele

- Status changed from New to Approved

- Assignee deleted (Miroslav Fidler)

- % Done changed from 0 to 100

Patched python code, tests ok inspecting various Value types.