Subject: bug in latest svn
Posted by mdelfede on Thu, 01 May 2008 13:35:06 GMT
View Forum Message <> Reply to Message

Opening a package with latest svn219 brings a invalid memory access error. That doesn't happen on svn218 previous release.

Ciao

Max

EDIT : The bad stuff is that bug appears only on optimal build, and not in debug mode....

Max

Subject: Re: bug in latest svn
Posted by Novo on Thu, 01 May 2008 17:52:27 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Thu, 01 May 2008 09:35Opening a package with latest svn219 brings a invalid memory access error. That doesn't happen on svn218 previous release.

In case of TheIDE + NOGTK I couldn't see any problem with opening a package.

Subject: Re: bug in latest svn
Posted by bytefield on Thu, 01 May 2008 18:01:37 GMT
View Forum Message <> Reply to Message

Well, it's not happen always, just some times. I use SVN.219 and meet this problem just once.
Edit: Maybe with NOGTK flag, the build is free of.

Subject: Re: bug in latest svn
Posted by mdelfede on Thu, 01 May 2008 18:45:38 GMT
View Forum Message <> Reply to Message

Well, for me it happens on every package opened...

Max

Subject: Re: bug in latest svn
Posted by bytefield on Thu, 01 May 2008 19:10:21 GMT
View Forum Message <> Reply to Message

Maybe it's a 64 bits issue? Which version have you tried? Both?

Subject: Re: bug in latest svn
Posted by Novo on Thu, 01 May 2008 20:31:43 GMT
View Forum Message <> Reply to Message

bytefield wrote on Thu, 01 May 2008 15:10Maybe it's a 64 bits issue? Which version have you tried? Both?

I'm using x86_64.
I'll try to run valgrind on TheIDE and check for possible problems.

Subject: Re: bug in latest svn
Posted by mdelfede on Thu, 01 May 2008 21:00:59 GMT
View Forum Message <> Reply to Message

I'm using 64 bit too

Subject: Re: bug in latest svn
Posted by Novo on Thu, 01 May 2008 21:28:23 GMT
View Forum Message <> Reply to Message

Novo wrote on Thu, 01 May 2008 16:31bytefield wrote on Thu, 01 May 2008 15:10Maybe it's a 64 bits issue? Which version have you tried? Both?

I'm using x86_64.
I'll try to run valgrind on TheIDE and check for possible problems.

That can be a problem. I can check that with debug build of TheIDE.

Subject: Re: bug in latest svn
Posted by bytefield on Thu, 01 May 2008 21:51:47 GMT
View Forum Message <> Reply to Message

OK, until now seems that just 64 bits build is affected.
Other users of 32 bits build? (Not me , if I remember well I've got such an error but don't know if with SVN.219).
For me it work normal, without errors.

Subject: Re: bug in latest svn
Posted by Novo on Thu, 01 May 2008 22:15:42 GMT
View Forum Message <> Reply to Message

Novo wrote on Thu, 01 May 2008 17:28Novo wrote on Thu, 01 May 2008 16:31bytefield wrote on Thu, 01 May 2008 15:10Maybe it's a 64 bits issue? Which version have you tried? Both?

I'm using x86_64.
I'll try to run valgrind on TheIDE and check for possible problems.

That can be a problem. I can check that with debug build of TheIDE.

A lot of that:

```
==31285== Conditional jump or move depends on uninitialised value(s)
==31285==    at 0x5919DB: Upp::CodeEditor::SyntaxState::ScanSyntax(unsigned short const*, unsigned short co
==31285==    by 0x5964E6: Upp::CodeEditor::ScanSyntax(int) (in /export/home/ssikorsk/dvlp/upp/svn/upp/out/G
==31285==    by 0x598E2C: Upp::CodeEditor::HighlightLine(int, Upp::Vector<Upp::LineEdit::Highlight>&, int)
==31285==    by 0x5E1F34: Upp::LineEdit::Paint0(Upp::Draw&) (in /export/home/ssikorsk/dvlp/upp/svn/upp/out/
==31285==    by 0x5E2A68: Upp::LineEdit::Paint(Upp::Draw&) (in /export/home/ssikorsk/dvlp/upp/svn/upp/out/G
==31285==    by 0x83ECEC: Upp::Ctrl::PaintOpaqueAreas(Upp::Draw&, Upp::Rect_<int> const&, Upp::Rect_<int> c
==31285==    by 0x83E949: Upp::Ctrl::PaintOpaqueAreas(Upp::Draw&, Upp::Rect_<int> const&, Upp::Rect_<int> c
==31285==    by 0x83E949: Upp::Ctrl::PaintOpaqueAreas(Upp::Draw&, Upp::Rect_<int> const&, Upp::Rect_<int> c
==31285==    by 0x83E949: Upp::Ctrl::PaintOpaqueAreas(Upp::Draw&, Upp::Rect_<int> const&, Upp::Rect_<int> c
==31285==    by 0x83E949: Upp::Ctrl::PaintOpaqueAreas(Upp::Draw&, Upp::Rect_<int> const&, Upp::Rect_<int> c
==31285==    by 0x83E949: Upp::Ctrl::PaintOpaqueAreas(Upp::Draw&, Upp::Rect_<int> const&, Upp::Rect_<int> c
```

==31285==   by 0x83E949: Upp::Ctrl::PaintOpaqueAreas(Upp::Draw&, Upp::Rect_<int> const&, Upp::Rect_<int> c
==31285==

==31285== Conditional jump or move depends on uninitialised value(s)
==31285==   at 0x84ABCB: Upp::Ctrl::Refresh() (in /export/home/ssikorsk/dvlp/upp/svn/upp/out/GCC.Debug.Gui
==31285==   by 0x632597: Upp::ArrayCtrl::Sort(Upp::ArrayCtrl::Order const&) (in /export/home/ssikorsk/dvlp
==31285==   by 0x6325DF: Upp::ArrayCtrl::Sort(int, int (*)(Upp::Value const&, Upp::Value const&)) (in /exp
==31285==   by 0x5559DA: Upp::ArrayCtrl::Sort() (in /export/home/ssikorsk/dvlp/upp/svn/upp/out/GCC.Debug.G
==31285==   by 0x566552: Browser::Reload() (in /export/home/ssikorsk/dvlp/upp/svn/upp/out/GCC.Debug.Gui.No
==31285==   by 0x571687: Browser::Browser() (in /export/home/ssikorsk/dvlp/upp/svn/upp/out/GCC.Debug.Gui.N
==31285==   by 0x45C366: Ide::Ide() (in /export/home/ssikorsk/dvlp/upp/svn/upp/out/GCC.Debug.Gui.Nogtk.Sha
==31285==   by 0x482D70: GuiMainFn_() (in /export/home/ssikorsk/dvlp/upp/svn/upp/out/GCC.Debug.Gui.Nogtk.S
==31285==   by 0x484155: main (in /export/home/ssikorsk/dvlp/upp/svn/upp/out/GCC.Debug.Gui.Nogtk.Shared/id

==31285== Conditional jump or move depends on uninitialised value(s)
==31285==   at 0x5C7C22: Upp::LineEdit::Layout() (in /export/home/ssikorsk/dvlp/upp/svn/upp/out/GCC.Debug.
==31285==   by 0x5C407D: Upp::LineEdit::SetFont(Upp::Font) (in /export/home/ssikorsk/dvlp/upp/svn/upp/out/
==31285==   by 0x592D51: Upp::CodeEditor::CodeEditor() (in /export/home/ssikorsk/dvlp/upp/svn/upp/out/GCC.
==31285==   by 0x43089D: AssistEditor::AssistEditor() (in /export/home/ssikorsk/dvlp/upp/svn/upp/out/GCC.D
==31285==   by 0x45C2A8: Ide::Ide() (in /export/home/ssikorsk/dvlp/upp/svn/upp/out/GCC.Debug.Gui.Nogtk.Sha
==31285==   by 0x482D70: GuiMainFn_() (in /export/home/ssikorsk/dvlp/upp/svn/upp/out/GCC.Debug.Gui.Nogtk.S
==31285==   by 0x484155: main (in /export/home/ssikorsk/dvlp/upp/svn/upp/out/GCC.Debug.Gui.Nogtk.Shared/id
==31285==
==31285== Conditional jump or move depends on uninitialised value(s)
==31285==   at 0x5C3040: Upp::LineEdit::SetHBar() (in /export/home/ssikorsk/dvlp/upp/svn/upp/out/GCC.Debug

==31285== by 0x5C7C7E: Upp::LineEdit::Layout() (in
/export/home/ssikorsk/dvlp/upp/svn/upp/out/GCC.Debug.
==31285== by 0x5C407D: Upp::LineEdit::SetFont(Upp::Font) (in
/export/home/ssikorsk/dvlp/upp/svn/upp/out/
==31285== by 0x592D51: Upp::CodeEditor::CodeEditor() (in
/export/home/ssikorsk/dvlp/upp/svn/upp/out/GCC.
==31285== by 0x43089D: AssistEditor::AssistEditor() (in
/export/home/ssikorsk/dvlp/upp/svn/upp/out/GCC.D
==31285== by 0x45C2A8: Ide::Ide() (in
/export/home/ssikorsk/dvlp/upp/svn/upp/out/GCC.Debug.Gui.Nogtk.Sha
==31285== by 0x482D70: GuiMainFn_() (in
/export/home/ssikorsk/dvlp/upp/svn/upp/out/GCC.Debug.Gui.Nogtk.S
==31285== by 0x484155: main (in
/export/home/ssikorsk/dvlp/upp/svn/upp/out/GCC.Debug.Gui.Nogtk.Shared/id

It looks like there is no "Use of uninitialised value of size 8" in debug build.

Probably, a version of GCC is important too ...

Subject: Re: bug in latest svn
Posted by mdelfede on Thu, 01 May 2008 22:16:14 GMT
View Forum Message <> Reply to Message

UPDATE :

The error is the same compiling previous svn versions... on ubuntu hardy. As previous deb builds were made on ubuntu gutsy, I guess that's some problem compiling upp on hardy.

So, GCC 4.2.3 and gnome 2.2.

Max

Subject: Re: bug in latest svn
Posted by bytefield on Thu, 01 May 2008 22:49:01 GMT
View Forum Message <> Reply to Message

I've rebuilt theide and getting "Invalid memory access"
So the problem affect 32 bits build too. Last SVN.219 build was made on Ubuntu 7.10.

Subject: Re: bug in latest svn

Posted by mdelfede on Thu, 01 May 2008 22:55:43 GMT
View Forum Message <> Reply to Message

Error happens on Core/Heap.cpp, line 311 :

m.list = l->next;

in function MemoryAllocSz(). Removing the smart allocation stuff for small sizes theide runs, but again memory error on exit.
So, I think something is breaking memory pool somewhere... quite difficult to locate.
Next error (removing smart allocation...) happens in file Core/LHeap.cpp, line 29, Unlink() function. That shows that pointers are garbaged

Max

Subject: Re: bug in latest svn
Posted by Novo on Thu, 01 May 2008 23:04:51 GMT
View Forum Message <> Reply to Message

bytefield wrote on Thu, 01 May 2008 18:49I've rebuilt theide and getting "Invalid memory access"
So the problem affect 32 bits build too. Last SVN.219 build was made on Ubuntu 7.10.

I'm using pretty old SuSe. GCC version 4.0.1. libc version 2.3.5 (20050720)

Subject: Re: bug in latest svn
Posted by mdelfede on Fri, 02 May 2008 07:44:10 GMT
View Forum Message <> Reply to Message

Novo wrote on Fri, 02 May 2008 01:04bytefield wrote on Thu, 01 May 2008 18:49I've rebuilt theide and getting "Invalid memory access"
So the problem affect 32 bits build too. Last SVN.219 build was made on Ubuntu 7.10.

I'm using pretty old SuSe. GCC version 4.0.1. libc version 2.3.5 (20050720)

So it must be a gcc/libc/gnome version stuff.
BTW, If you compile it in release/optimal mode BUT enabling full debug, it still present the problem but it becomes debuggable....
But I still think that'll not an easy bug to find

Max

EDIT : Enabling DEBUG in Heap.cpp theide stops with "writes to freed blocks detected" error... maybe an hint ?

Subject: Re: bug in latest svn
Posted by Novo on Fri, 02 May 2008 20:15:55 GMT
View Forum Message <> Reply to Message

members.

That can cause application to crash, especially when you change version of compiler.

Subject: Re: bug in latest svn
Posted by mdelfede on Fri, 02 May 2008 21:45:51 GMT
View Forum Message <> Reply to Message

and a bunch of uninitialized class members.

That can cause application to crash, especially when you change version of compiler.

Interesting ! I'm trying to locate the bug, but with no success up to now...

Max

Well, just a new finding... The error seems not depend on optimization, but only on DEBUG flag... if you compile in release mode (DEBUG flag undefined), with or without debugging info, the bug is there. If you manually enable DEBUG flag, the bug disappears. maybe the bug is in memory allocation routines, which are different on DEBUG flag.

Subject: Re: bug in latest svn
Posted by Novo on Sat, 03 May 2008 15:42:28 GMT
View Forum Message <> Reply to Message

Novo wrote on Fri, 02 May 2008 16:15

Sorry, couldn't keep my promice.

I probably miss something, but code below always was a problem.

```
const String& GetCppFile(int i)
{
 INTERLOCKED_(cpp_file_mutex) {
  return cpp_file[i];
 }
 return String();
}
```

Should be something like that:

```
const String& GetCppFile(int i)
{
 INTERLOCKED_(cpp_file_mutex) {
  return cpp_file[i];
 }

        static String value;
 return value;
}
```

I'm not sure about thread-safety of static variable initialization though.

Subject: Re: bug in latest svn
Posted by mdelfede on Sat, 03 May 2008 17:17:20 GMT
View Forum Message <> Reply to Message

Novo wrote on Sat, 03 May 2008 17:42Novo wrote on Fri, 02 May 2008 16:15

Sorry, couldn't keep my promice.

I probably miss something, but code below always was a problem.

```
const String& GetCppFile(int i)
{
 INTERLOCKED_(cpp_file_mutex) {
  return cpp_file[i];
 }
 return String();
}
```

Should be something like that:

```
const String& GetCppFile(int i)
{
 INTERLOCKED_(cpp_file_mutex) {
  return cpp_file[i];
 }

       static String value;
 return value;
}
```

I'm not sure about thread-safety of static variable initialization though.

AFAIK that's a right behaviour, as returned temporary string is firs assigned to result and then deleted. With static variables you loose reentrancy (usually).

BTW, after long debugging I couldn't locate the problem....

Max

Subject: Re: bug in latest svn
Posted by mdelfede on Sat, 03 May 2008 22:07:26 GMT
View Forum Message <> Reply to Message

Well.... found at least a dirty workaround.

In 'heap.cpp', line 148 :

```
//#ifdef _DEBUG
#if 1 // @@@@@ WORKAROUND !!!!!
void FreeFill(dword *ptr, int count)
```

```
{
 while(count--)
  *ptr++ = 0x65657246;
}

void FreeCheck(dword *ptr, int count)
{
 int c = count;
 while(c--)
  if(*ptr++ != 0x65657246)
   HeapPanic("Writes to freed blocks detected", ptr, sizeof(dword) * count);
}

#endif
```

And, (that's the real workaround....), in file String.cpp, line 22 :

```
static inline void MFree_S(void *ptr)
{
 MCache& m = mcache[1];
 ((FreeLink *)ptr)->next = m.list;
 m.list = (FreeLink *)ptr;
//#ifdef _DEBUG
#if 1 // @@@@ WORKAROUND !!!!!
#ifdef CPU_64
 FreeFill((dword *)ptr + 2, 32 / 4 - 2);
#else
 FreeFill((dword *)ptr + 1, 32 / 4 - 1);
#endif
#endif
 if(++m.count > CACHEMAX)
  MFree_Reduce(m, 1);
}
```

This 'hides' the bug, but I guess it's still there.
Theide run fine, and also my (few) test apps.

For Mirek : the bug should be OR in string functions, OR in Vector stuff, in particular in VectorGrow stuffs.

Here a small app that shows the behaviour....BUT, to test it you have to :
1) Build in Optimal mode (NO DEBUG flag set)
2) Change optimal mode debug flag to -O0 (otherwise it's undebuggable because of optimizations...)
3) Enable full debug info even in optimal mode

4) Enable DEBUG just in heap.cpp, and change allocation stuffs to the debugger ones (some changes in core.h, heapdbg.cpp and others...)
5) The hard stuff... the bug arises before main(), so you've got to disable some initialization stuffs (quite long work). If not, you'll have to follow some hundreds of calls before main().

After point 5 done, the app will execute the first loop, then show the error. Following this, you'll find mostly string and vectormap calls. Quite long to follow, yet, but better than debugging theide
I stopped here because your string stuffs are quite un-understandable (and uncommented....).
Here the test app :

```
#include <Core/Core.h>

using namespace Upp;

CONSOLE_APP_MAIN
{
 char key[200];

 VectorMap<String, char *> aMap;

 int nKeys = 100;

 for(int i = 0; i < nKeys; i++)
 {
  sprintf(key, "Chiave di prova n.%d", i);
  RLOG("\n\n============================================================\
nAdding key " << key);
  RLOG("AllocTest before....");
//  AllocTest();
  aMap.Add(key);
  RLOG("Key added, AllocTest after");
//  AllocTest();
 }

 exit(0);
}
```

Commented AllocTest() stuffs are from this routine :

```
void AllocTest(void)
{
 RLOG("AllocTest -- Entering");
 const int numAllocs = 100;
 size_t sizeAlloc = 32;

 void *p[numAllocs];
 for(int i = 0; i < numAllocs; i++)
  p[i] = MemoryAllocSz(sizeAlloc);
```

```
 RLOG("AllocTest -- Memory allocated");
 for(int i = 0; i < numAllocs; i++)
  MemoryFree(p[i]);
 RLOG("AllocTest -- Memory freed");

}
```

These just do some dummy alloc/frees to trigger the bug.

BTW... I've got a suggestion here... just to ease the debugging.
I guess we should have alternate memory allocation stuffs that could be used (independently from DEBUG flag) to check the heap on demand. MemoryCheck() and memoryCheckDebug() doesn't do the job, even when enabled by hand. And, the best would be to have a switchable pointer-checking functions for all Upp containers, called entering and leaving each container's method.
All that could be switched by a CHECKPOINTERS macro, and should check container's internal pointers and free heap.

Another 'stylish' stuff... many functions that resides in heap.cpp should (IMHO) belong to heapdbg.cpp.

Ciao

Max

Well, after some tests on 32 bit (thanks Bytefield), I've seen that workarount don't work for 32 bit builds... so, better to stay on SVN 219 build (for 32 bit) and SVN218 build (for 64) up to the stuff is fixed.

Max

Subject: Re: bug in latest svn
Posted by Novo on Sun, 04 May 2008 01:35:18 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Sat, 03 May 2008 13:17
AFAIK that's a right behavior, as returned temporary string is firs assigned to result and then deleted.

IMHO, the way you describe would work if the function looks like below.

String GetCppFile(int i);

Current implementation is similar to code below.

```
const String* GetCppFile(int i)
{
 INTERLOCKED_(cpp_file_mutex) {
  return &cpp_file[i];
 }
 return &String();
}
```

Does this look correct to you?

Subject: Re: bug in latest svn
Posted by mdelfede on Sun, 04 May 2008 10:17:27 GMT
View Forum Message <> Reply to Message

Novo wrote on Sun, 04 May 2008 03:35mdelfede wrote on Sat, 03 May 2008 13:17
AFAIK that's a right behavior, as returned temporary string is firs assigned to result and then deleted.

IMHO, the way you describe would work if the function looks like below.

String GetCppFile(int i);

Current implementation is similar to code below.

```
const String* GetCppFile(int i)
{
 INTERLOCKED_(cpp_file_mutex) {
  return &cpp_file[i];
 }
 return &String();
}
```

Does this look correct to you?

Nope, I guess... here you're returning a reference to a local variable, that can (and usually IS) destroyed BEFORE function returns. So, you could make it static to solve the problem, but then you'd have many other problems, in particular with MT.
Returning a String() value is less efficient, but guarantees that string is not destroyed on function return before the value is taken.

Max

Subject: Re: bug in latest svn
Posted by Novo on Sun, 04 May 2008 15:04:27 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Sun, 04 May 2008 06:17Returning a String() value is less efficient, but guarantees that string is not destroyed on function return before the value is taken.

Isn't

return String();

equal to

String anonymous_value;
return anonymous_value;

?

const String& GetCppFile(int i);

String value1 = GetCppFile(0); // Case A
const String& value2 = GetCppFile(0); // Case B

In case A GetCppFile() will work correctly.
Case B will introduce a bug.

IMHO, returning "const String&" is just not thread-safe. Object can be deleted in transition.

Subject: Re: bug in latest svn
Posted by Novo on Sun, 04 May 2008 15:11:27 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Sun, 04 May 2008 06:17Nope, I guess... here you're returning a reference to a local variable, that can (and usually IS) destroyed BEFORE function returns.

IMHO, returning reference to a local variable is the same thing as returning pointer to a local variable ...

Subject: Re: bug in latest svn
Posted by mdelfede on Sun, 04 May 2008 15:14:41 GMT
View Forum Message <> Reply to Message

Novo wrote on Sun, 04 May 2008 17:04

const String& GetCppFile(int i);

String value1 = GetCppFile(0); // Case A
const String& value2 = GetCppFile(0); // Case B

In case A GetCppFile() will work correctly.
Case B will introduce a bug.

IMHO, returning "const String&" is just not thread-safe. Object can be deleted in transition.

I guess you're right, with your 2 examples.
On first example it 'should' work, because the source string isn't deleted before assignment, so the value is transferred. The secon example is buggy because you return a reference to an object that'll be deleted soon. BTW, I don't know how does the compiler behave with the const modifier.... but I guess it'll not solve the problem.

So the code is bad, imo.... is that maybe the problem ?

Max

Subject: Re: bug in latest svn
Posted by mdelfede on Sun, 04 May 2008 15:16:04 GMT
View Forum Message <> Reply to Message

Novo wrote on Sun, 04 May 2008 17:11mdelfede wrote on Sun, 04 May 2008 06:17Nope, I guess... here you're returning a reference to a local variable, that can (and usually IS) destroyed BEFORE function returns.

IMHO, returning reference to a local variable is the same thing as returning pointer to a local variable ...

Yes, it is, and you're right, it's a dangerous way.
It's strange that compiler doesn't warn about....

Max

Subject: Re: bug in latest svn
Posted by mdelfede on Sun, 04 May 2008 15:43:34 GMT
View Forum Message <> Reply to Message

This one shows that you're right...

```
long retlong(void)
{
 static long l = 5;
 return l++;

}

const long &test(void)
{
 return retlong();

}

int main(int argc, const char *argv[])
{
 const long &a = test();
 const long &b = test();

 long aa = a;
 long bb = b;

 return 0;
}
```

In my test run, both aa and bb get value of 6.

Max

Subject: Re: bug in latest svn
Posted by Novo on Sun, 04 May 2008 16:07:56 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Sun, 04 May 2008 11:16
It's strange that compiler doesn't warn about....

Actually, it does ...

Subject: Re: bug in latest svn
Posted by mdelfede on Sun, 04 May 2008 16:12:45 GMT
View Forum Message <> Reply to Message

Indeed, I was wrong before :

This piece of code :

```
#include "stdio.h"

static long seed = 0;

class LongClass
{
  long l;
 public :
  long Get(void) const { return l; }
  LongClass() { l = seed++ ; }

};

const LongClass &test(const LongClass &lClass = LongClass())
{
 return lClass;

}
```

```
int main(int argc, const char *argv[])
{
 const LongClass &a = test();
 const LongClass &b = test();

 long aa = a.Get();
 long bb = b.Get();

 return 0;
}
```

Shows that's perfectly legal to return references to const temporaries... Here aa gets a value of 0 and bb of 1, correctly.

That behaviour allows to initialize default reference arguments with objects created on the fly. I don't know where does compile store the actual value....

Max

Subject: Re: bug in latest svn
Posted by mirek on Sun, 04 May 2008 16:28:26 GMT
View Forum Message <> Reply to Message

Novo wrote on Sat, 03 May 2008 11:42Novo wrote on Fri, 02 May 2008 16:15

Sorry, couldn't keep my promice.

I probably miss something, but code below always was a problem.

```
const String& GetCppFile(int i)
{
 INTERLOCKED_(cpp_file_mutex) {
  return cpp_file[i];
 }
 return String();
}
```

Should be something like that:

```
const String& GetCppFile(int i)
{
 INTERLOCKED_(cpp_file_mutex) {
  return cpp_file[i];
 }

       static String value;
 return value;
}
```

I'm not sure about thread-safety of static variable initialization though.

Well, second return is never performed. It is there only to shut-up the compiler about missing return.... (which, of course, in this context sounds like void effort

Mirek

P.S.: Other than that, while working in the current IDE, the code should be changed anyway.

Subject: Re: bug in latest svn
Posted by mdelfede on Sun, 04 May 2008 19:46:17 GMT
View Forum Message <> Reply to Message

Well... maybe found bug's location.
It's on String.cpp, Alloc() or something near this.

Investigating a bit more....

Max

Subject: Re: bug in latest svn
Posted by Novo on Tue, 06 May 2008 02:07:28 GMT
View Forum Message <> Reply to Message

luzr wrote on Sun, 04 May 2008 12:28Well, second return is never performed. It is there only to shut-up the compiler about missing return.... (which, of course, in this context sounds like void effort

In this case I have two questions:

1) Why you are not using something like mutex guard for such code? That would eliminate requirement in "return String()".

2) Why this code was left not thread-safe?

Just curios ...

Subject: Re: bug in latest svn
Posted by Novo on Tue, 06 May 2008 02:10:47 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Sun, 04 May 2008 12:12
Shows that's perfectly legal to return references to const temporaries... Here aa gets a value of 0 and bb of 1, correctly.

Believe me, that is not safe

Subject: Re: bug in latest svn
Posted by mirek on Wed, 07 May 2008 11:59:53 GMT
View Forum Message <> Reply to Message

Novo wrote on Mon, 05 May 2008 22:07luzr wrote on Sun, 04 May 2008 12:28Well, second return is never performed. It is there only to shut-up the compiler about missing return.... (which, of course, in this context sounds like void effort

In this case I have two questions:

1) Why you are not using something like mutex guard for such code? That would eliminate requirement in "return String()".

2) Why this code was left not thread-safe?

Just curios ...

Well, INTERLOCKED_ is a mutex guard...

Unfortunately, the compiler is not able to figure out that second return never takes place...

Mirek

Subject: Re: bug in latest svn
Posted by mr_ped on Wed, 07 May 2008 12:13:51 GMT
View Forum Message <> Reply to Message

The warning may be avoided trough:

```
static const String interlock_problem = "Some problem with INTERLOCKED_ happened!";
const String& GetCppFile(int i)
{
 INTERLOCKED_(cpp_file_mutex) {
  return cpp_file[i];
 }
 return interlock_problem;
}
```

This will also lead to weird String constant in case the INTERLOCKED_ macro gets damaged by something and it will stop work correctly.
(and returning the interlock_problem const reference is thread safe no matter what? I wish I would KNOW, I will have to work on my "knowledge" of ordinary C++ MT and U++ way. The general ASM knowledge doesn't help here too much.)

Subject: Re: bug in latest svn
Posted by mdelfede on Wed, 07 May 2008 12:39:19 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 07 May 2008 13:59Novo wrote on Mon, 05 May 2008 22:07luzr wrote on Sun, 04 May 2008 12:28Well, second return is never performed. It is there only to shut-up the compiler about missing return.... (which, of course, in this context sounds like void effort

In this case I have two questions:

1) Why you are not using something like mutex guard for such code? That would eliminate requirement in "return String()".

2) Why this code was left not thread-safe?

Just curios ...

Well, INTERLOCKED_ is a mutex guard...

Unfortunately, the compiler is not able to figure out that second return never takes place...

Mirek

It should be enough to place a #pragma warn somewhere to disable warning.

Max

Subject: Re: bug in latest svn
Posted by Novo on Wed, 07 May 2008 20:07:09 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 07 May 2008 07:59Novo wrote on Mon, 05 May 2008 22:07luzr wrote on Sun, 04 May 2008 12:28Well, second return is never performed. It is there only to shut-up the compiler about missing return.... (which, of course, in this context sounds like void effort

In this case I have two questions:

1) Why you are not using something like mutex guard for such code? That would eliminate requirement in "return String()".

2) Why this code was left not thread-safe?

Just curios ...

Well, INTERLOCKED_ is a mutex guard...

Unfortunately, the compiler is not able to figure out that second return never takes place...

Mirek

What is wrong with the code below?

```
const String& GetCppFile(int i)
{
 Mutex::Lock guard(cpp_file_mutex);

 return cpp_file[i];
}
```

What is so special about INTERLOCKED_ ?
