CrystaX NDK - Bug #1079

Objective-C

09/29/2015 02:25 PM - Christophe Delannoy

Status:	Invalid	**Start date:**	09/29/2015
Priority:	High	**Due date:**	
Assignee:	Dmitry Moskalchuk	**% Done:**	0%
Category:	cocotron	**Estimated time:**	0.00 hour
Target version:	10.3.0		
CPU Architecture:		**Android version:**	
Host OS:		**CrystaX Version:**	
Toolchain:			

Description

On XCode, a code like
@ NSString *temp;
NSLog(@"%@", temp);@
won't crash, on CrystaX (Android Studio) it will... ; you can also sen the "count" message to a non-allocated NSArray (from XCode), it will return zero.

Is the no-crash a specification from Objective-C, or an Apple-only implementation ?

History

#1 - 09/29/2015 02:52 PM - Dmitry Moskalchuk

- Tracker changed from Task to Bug

- Category set to cocotron

- Assignee set to Dmitry Moskalchuk

- Priority changed from Normal to High

- Target version set to 10.3.0

This is definitely a bug, since our Foundation API behaviour should mimic Apple's one as much as possible.

#2 - 09/29/2015 02:52 PM - Dmitry Moskalchuk

- Status changed from Open to In Progress

#3 - 09/29/2015 03:34 PM - Dmitry Moskalchuk

I've just created separate test case for this issue, but I can't reproduce bug. Here is source of test:

```
#import

int main()
{
NSString *test;
NSLog(@"%", test);

return 0;

}
```

This works just fine, without any problem. Could you provide minimal example reproducing your problem (example with NSArray would be helpful too)?

#4 - 09/29/2015 05:32 PM - Christophe Delannoy

> I've just created separate test case for this issue, but I can't reproduce bug.
> [...] Could you provide minimal example reproducing your problem (example with NSArray would be helpful too)?

File WM_Connection.h :
@class WM_Connection
@{@
...
static int TEST();
}@

File WM_Connection.mm (simplified) :
@int WM_Connection::TEST() {
NSString *temp; // = @@@"blabla";
NSLog(@"%@", temp);
return 1;
}@

java part :
@int tt = WM_Connection.TEST();@

The wrapper is generated by SWIG, I don't know if it has an effect... the wrapped function contains :
@result = (int)WM_Connection::TEST();
jresult = (jint)result;
return jresult;@

#5 - 09/29/2015 05:39 PM - Christophe Delannoy

Here is source of test:

> [...]
>
> This works just fine, without any problem. Could you provide minimal example reproducing your problem (example with NSArray would be helpful too)?
> I just realise, in
> @NSLog(@@@"%", test);@
> you'd just forgot the second "@@@"... :
> @NSLog(@@@"%@@@", test);@
> So the parameter was ignored, it just produces a warning in Xcode

By the way, can you tell me the sexy formating you used in your response (with highlights) ?

#6 - 09/29/2015 05:42 PM - Christophe Delannoy

> Here is source of test:
> [...]

I just realise, in
@NSLog(@@@"%", test);@
you'd just forgot the second "@@@"... :
@NSLog(@@@"%@@@", test);@
So the parameter was ignored, it just produces a warning in Xcode

By the way, can you tell me the sexy formating you used in your response (with highlights) ?

#7 - 09/29/2015 06:11 PM - Christophe Delannoy

> By the way, can you tell me the sexy formating you used in your response (with highlights) ?
> Forgot this, I've found how to

#8 - 09/30/2015 09:30 AM - Dmitry Moskalchuk

Christophe Delannoy wrote:

> I just realise, in
> @NSLog(@@@"%", test);@
> you'd just forgot the second "@@@"... :
> @NSLog(@@@"%@@@", test);@
> So the parameter was ignored, it just produces a warning in Xcode

Well, it's completely another picture then! In this case, you're passing uninitialized pointer to NSLog, and ask him to format NSString object it points to. But in your case it can point anywhere since it's uninitialized. This is wrong code and it cause crash by obvious reason. I'm pretty sure the same would be in Xcode, if you build and run release version of code. I guess the only reason why it don't crash in Xcode on your side, is that you're running debug version of code, where all unitialized variables are filled by zeros. Anyway, this is wrong code, and not a bug in Foundation API.

I suspect the same is regarding NSArray, right? You've said something about non-allocated NSArray - if it's just uninitialized pointer, then sending any message to it is undefined behaviour (crash, most frequently).

> By the way, can you tell me the sexy formating you used in your response (with highlights) ?

https://tracker.crystax.net/help/en/wiki_syntax_detailed_textile.html

#9 - 09/30/2015 11:58 AM - Dmitry Moskalchuk

- Status changed from In Progress to Invalid

#10 - 09/30/2015 05:04 PM - Christophe Delannoy

Dmitry Moskalchuk wrote:

> Christophe Delannoy wrote:
>
> > I just realise, in
> > @NSLog(@@@"%", test);@
> > you'd just forgot the second "@@@"... :
> > @NSLog(@@@"%@@@", test);@
> > So the parameter was ignored, it just produces a warning in Xcode
>
> Well, it's completely another picture then! In this case, you're passing uninitialized pointer to NSLog, and ask him to format NSString object it points to. But in your case it can point anywhere since it's uninitialized. This is wrong code and it cause crash by obvious reason. I'm pretty sure the same would be in Xcode, if you build and run release version of code. I guess the only reason why it don't crash in Xcode on your side, is that you're running debug version of code, where all unitialized variables are filled by zeros. Anyway, this is wrong code, and not a bug in Foundation API.

(1) I think that in Obj-C, ids are initialized with zero, look this : I tried this with gcc (not Xcode) :

```
#import

int main(int argc, char * argv[])
{
NSString *temp = 0;
NSString *temp2 = [NSString stringWithFormat:@"Hello %@!", temp];
char temp3[16];
bool t = [temp2 getCString:temp3 maxLength:16 encoding:NSMacOSRomanStringEncoding];
#pragma unused(t)
printf("%s\n", temp3);
}
```

then I do in terminal :

```
patricia:testObjC Administrateur$ gcc test.m -framework Cocoa
patricia:testObjC Administrateur$ ./a.out
So I get the line :
Hello (null)!
```

> I suspect the same is regarding NSArray, right? You've said something about non-allocated NSArray - if it's just uninitialized pointer, then sending any message to it is undefined behaviour (crash, most frequently).

(2) I found a doc at http://cagt.bu.edu/w/images/b/b6/Objective-C_Programming_Language.pdf,
at the bottom of page #17 we can read
"In Objective-C it is valid to send a message to nil -it simply has no effect at runtime"
...followed with sample code, the kind of code that is really really often used, but I can't copy/paste from the pdf, please follow the link

#11 - 09/30/2015 05:18 PM - Dmitry Moskalchuk

Christophe Delannoy wrote:

> (1) I think that in Obj-C, ids are initialized with zero, look this : I tried this with gcc (not Xcode) :
> [...]
>
> then I do in terminal :

> [...]
> So I get the line :
> [...]

Yes, if you initialize pointer by zero, it will work - in both Apple and CrystaX NDK environment. But if you pass uninitialized pointer, code will crash - again, in both Apple and CrystaX NDK environments.

Your initial example:

NSString *temp;
NSLog(@"%@", temp);

Your last example:

NSString *temp = 0;
NSString *temp2 = [NSString stringWithFormat:@"Hello %@!", temp];

In initial example you have unitialized @temp@ pointer, but in last examle @temp@ is initialized by zero. That's it.

#12 - 10/01/2015 10:41 AM - Christophe Delannoy

> In initial example you have unitialized temp pointer, but in last examle temp is initialized by zero. That's it.

I'm sorry, please replace the line with:

NSString *temp;// = 0;

This code won't crash.

I tried to check the NSArray behaviour, here is the whole file:

#import

int main(int argc, char * argv[])
{
NSString *temp;// = 0;
printf("([temp length] : %li)\n", [temp length]);
NSString *temp2 = [NSString stringWithFormat:@"Hello 1 %@!", temp];
char temp3[16];
bool t = [temp2 getCString:temp3 maxLength:16 encoding:NSMacOSRomanStringEncoding];
if(t)
printf("%s\n", temp3);

```
NSArray *array = 0;
NSUInteger test = [array count];
printf("(test : %li)\n", test);
temp = (NSString *)[array firstObject];
printf("(&temp : %p)\n", temp);
temp2 = [NSString stringWithFormat:@"Hello 2 %@!", temp];
printf("(&temp2 : %p)\n", temp2);
t = [temp2 getCString:temp3 maxLength:16 encoding:NSMacOSRomanStringEncoding];
if(t)
    printf("%s\n", temp3);
```

}

This code works perfectly, BUT... here, if array is not initialised this comes to a "Segmentation fault: 11" error. Not with @NSString@. Thats annoying because we often use the

[[NSArray alloc] init];

form as Apple strongly encourages it. So the issue for the moment is that Android crashes with @NSString@ ; I'm looking at @NSArray@ as soon as possible.

#13 - 10/01/2015 11:12 AM - Dmitry Moskalchuk

Christophe Delannoy wrote:

> This code works perfectly, BUT... here, if @array@ is not initialised this comes to a "@Segmentation fault: 11@" error. Not with @NSString@.

Again, using uninitialised pointers is undefined behaviour - this code could crash or it could work occasionally - this depends on compiler options, on current memory status, etc. It's "undefined behaviour":https://en.wikipedia.org/wiki/Undefined_behavior. Instead of trying to make wrong code working, better just fix wrong code.

Here are results of running your code on OS X 10.11:

```
$ clang --version
Apple LLVM version 7.0.0 (clang-700.0.72)
Target: x86_64-apple-darwin15.0.0
Thread model: posix

$ cat test.m
#import

int main(int argc, char * argv[])
{
NSString *temp;// = 0;
printf("([temp length] : %li)\n", [temp length]);
NSString *temp2 = [NSString stringWithFormat:@"Hello 1 %@!", temp];
char temp3[16];
bool t = [temp2 getCString:temp3 maxLength:16 encoding:NSMacOSRomanStringEncoding];
if(t)
printf("%s\n", temp3);

NSArray *array = 0;
NSUInteger test = [array count];
printf("(test : %li)\n", test);
temp = (NSString *)[array firstObject];
printf("(&temp : %p)\n", temp);
temp2 = [NSString stringWithFormat:@"Hello 2 %@!", temp];
printf("(&temp2 : %p)\n", temp2);
t = [temp2 getCString:temp3 maxLength:16 encoding:NSMacOSRomanStringEncoding];
if(t)
    printf("%s\n", temp3);

}

$ clang -O0 -o test test.m -framework Foundation && ./test
([temp length] : 0)
Hello 1 (null)!
(test : 0)
(&temp : 0x0)
(&temp2 : 0x7ffa1940f130)
Hello 2 (null)!

$ clang -O2 -o test test.m -framework Foundation && ./test
([temp length] : 1)
[1]    56357 segmentation fault  ./test
```

As you can see, it works on OS X only if optimisations are disabled, in which case clang fill uninitialised values by zeros. If you enable optimisations (-O2), your example just crashes.

> Thats annoying because we often use the [...] form as Apple strongly encourages it.

This form is completely correct and has nothing to uninitialised pointers.

#14 - 10/01/2015 11:46 AM - Christophe Delannoy

> [...] If you enable optimisations (-O2), your example just crashes.

You are right ...

> > Thats annoying because we often use the [...] form as Apple strongly encourages it.
> > This form is completely correct and has nothing to uninitialised pointers.

I wondered about it, because @alloc@ could fail... but in this case it precisely returns @nil@, so... you are right too !
Many thanks for your work.
Regards.

#15 - 10/01/2015 11:57 AM - Christophe Delannoy

hem... just as a glance, Xcode has a really performing analyzer tool ((menu)>Product>Analyze) that's does not complain at all about the non-initialized @NSString@, this currently means no-crash... when you talked about mimic Apple's behavior it may been an interesting aspect, considering xcoders usages... at least a note about it ?

#16 - 10/01/2015 12:03 PM - Dmitry Moskalchuk

Christophe Delannoy wrote:

> hem... just as a glance, Xcode has a really performing analyzer tool ((menu)>Product>Analyze) that's does not complain at all about the non-initialized @NSString@, this currently means no-crash... when you talked about mimic Apple's behavior it may been an interesting aspect, considering xcoders usages... at least a note about it ?

If you enable warnings in Xcode project settings, it *will* warn you. No need to use clang analyzer for that - this is what compiler itself can say you, if you ask him:

```
$ clang -Wall -o test test.m -framework Foundation
test.m:6:40: warning: variable 'temp' is uninitialized when used here [-Wuninitialized]
printf("([temp length] : %li)\n", [temp length]);
~~~

test.m:5:19: note: initialize the variable 'temp' to silence this warning
NSString *temp;// = 0;
^
= nil
1 warning generated.
```

The only problem here is that Xcode don't enable warnings by default in newly created projects, and this is wrong in my opinion. First thing I always do in any Xcode project - edit its settings and enable compiler warnings.

BTW, in CrystaX NDK we've enabled warnings by default, so if you look on warnings when building your example, you'll see it.

#17 - 10/01/2015 08:20 PM - Christophe Delannoy

Aïe aïe aïe, I spend many hours on this (in Xcode !): switched on the non-initialized warning flag, added @-Wall@ or @-Wuninitialized@ to "Other C Flags", ... nothing work, I mean no "uninitialized" warning... and I find this note on the LLVM list: "[...] under ARC [...], all strong references are initialized to 'nil', which means this is a completely standard message to nil case. If you turn off ARC, you get the expected warning (along with leak warnings, of course)." !

Try this: @$ clang -Wall -fobjc-arc -o test test.m -framework Foundation@, you get no error !
For my Xcode project I needed (as it is activated by default) to (temporarily) uncheck the ARC feature (in can requiere some @#if __has_feature(objc_arc)@ here and there) and all is right now with warnings.

I just got a serious valuable time gain in iOS->Android portage, many thanks !

#18 - 10/01/2015 08:41 PM - Dmitry Moskalchuk

Christophe Delannoy wrote:

> Aïe aïe aïe, I spend many hours on this (in Xcode !): switched on the non-initialized warning flag, added @-Wall@ or @-Wuninitialized@ to "Other C Flags", ... nothing work, I mean no "uninitialized" warning... and I find this note on the LLVM list: "[...] under ARC [...], all strong references are initialized to 'nil', which means this is a completely standard message to nil case. If you turn off ARC, you get the expected warning (along with leak warnings, of course)." !
>
> Try this: @$ clang -Wall -fobjc-arc -o test test.m -framework Foundation@, you get no error !

Yes, if ARC enabled, then compiler put many additional code around your code lines, in particular zeroing uninitialized pointers. But, to enable it, you should explicitly say about that adding @-fobjc-arc@ compiler option.

ARC is supported in latest builds of CrystaX NDK, so you don't need to modify your code, just download latest build from https://dl.crystax.net/ndk/darwin/ and use it instead of crystax-ndk-10.2.1. Then, to enable ARC, add @-fobjc-arc@ to your Android.mk:

```
LOCAL_OBJCFLAGS += -fobjc-arc
```

If you do that, your example will work with CrystaX NDK too, the same as with Apple's clang.

#19 - 10/02/2015 04:05 PM - Christophe Delannoy

> to enable [ARC], you should explicitly say about that adding @-fobjc-arc@ compiler option.
> It is enabled by default in Xcode. No matter.
>
> ARC is supported in latest builds of CrystaX NDK, so you don't need to modify your code, just download latest build from

> https://dl.crystax.net/ndk/darwin/ and use it instead of crystax-ndk-10.2.1. Then[...]

Wonderfull...

Did I mess something ?
I got the latest version (@...b689-darwin-x86_64.bin@) and any attempt to open it make it is compressed as a @.bin.cpgz@ file, the @...-x86.bin@ too...
I' downloading the #687version (2 hours remaining...) for another try

#20 - 10/02/2015 04:11 PM - Dmitry Moskalchuk

Christophe Delannoy wrote:

> I got the latest version (@...b689-darwin-x86_64.bin@) and any attempt to open it make it is compressed as a @.bin.cpgz@ file, the @...-x86.bin@ too...

b689 is ok. Double clicking on it in Finder doesn't work - for some reason Finder tries to pack it into .cpgz, instead of unpacking. So to unpack it you should open Terminal, cd to folder where .bin stored, and type the following:

chmod +x crystax-ndk-10.2.1-b689-darwin-x86_64.bin
./crystax-ndk-10.2.1-b689-darwin-x86_64.bin

#21 - 10/09/2015 02:13 PM - Christophe Delannoy

Hi, I got in great troubles since my last connexion,

> for some reason Finder tries to pack it into .cpgz, instead of unpacking. So [...]
> ... some Magic Number about the byte order

(1) the *b689* does'nt work, some folder is missing (@build/core/@) but probably it comes from an incomplete extraction (my disk is full, see #2)
(2) the *b694* is better, as no @.mk@ is missing, but the size of the folder is +17,43Go+, as the size for the extracted official .tar gives +6,75Go+ ! Witch tool do you use ? for me it's @lipo@, as it prevents for some additional file in the archive (like @__MAC_OS@ if I well remember), I use it for Mozilla plug-ins for example.
(3) according to Gradle errors (bellow) it seems that the @-fobjc-arc@ option is the one by default. The @-fno-objc-arc@ one let me compile and execute in the non-ARC mode
(4) *The true question is coming*, I flag the @.mk@ to ARC-mode and here are the Gradle errors :

Error:(581, 22) error: pointer to non-const type 'NSMutableString *' with no explicit ownership
Error:(581, 4) error: C-style cast from 'jlong *' (aka 'long long *') to 'NSMutableString *__strong *' casts away qualifiers

My wrappers are generated with *SWIG* (not really Obj-C compliant), and for a method that returns a @NSMutableString@ object, e.g.:

NSMutableString *WM_Connection_Prefs::url() { return p_Url; }*
it generates :
*result = (NSMutableString *)(arg1)->url();‰*
**(NSMutableString *)&jresult = result; // <- errors here ‰*
I'm changing my code to return C++ objects only

What do you think : (a) SWIG is not intended for this usage, (b) it's an issue from Crystax, or (c) an issue for me (replace all Obj-C returned objects with compatibles ones) ?

PS:
It could be a good idea to implement a "jni.h" alternative

#22 - 10/09/2015 03:02 PM - Dmitry Moskalchuk

Christophe,

Your questions are not related to this ticket, so lets move this discussion into "crystax-ndk":https://groups.google.com/forum/#!forum/crystax-ndk group. However, since you already asked here, see below my answers (but further questions please post into "crystax-ndk": https://groups.google.com/forum/#!forum/crystax-ndk group):

> (1) the *b689* does'nt work, some folder is missing (@build/core/@) but probably it comes from an incomplete extraction (my disk is full, see #2)
> (2) the *b694* is better, as no @.mk@ is missing, but the size of the folder is +17,43Go+, as the size for the extracted official .tar gives +6,75Go+ ! Witch tool do you use ? for me it's @lipo@, as it prevents for some additional file in the archive (like @__MAC_OS@ if I well remember), I use it for Mozilla plug-ins for example.

This is because we include Boost prebuilt libraries into CrystaX NDK. These libraries are big, especially taking into account they include full debug information. If you don't need Boost, just remove @$NDK/sources/boost@ and @$NDK/sources/boost+icu@ folders.

> (3) according to Gradle errors (bellow) it seems that the @-fobjc-arc@ option is the one by default. The @-fno-objc-arc@ one let me compile and

> execute in the non-ARC mode

It depends on how you build your app. If you're using NDK-provided @ndk-build@ tool, then yes, @-fobjc-arc@ is added by default to clang's compiler options. If you're using another build system, where clang is called separately, @-fobjc-arc@ need to be passed explicitly.

> (4) *The true question is coming*, I flag the @.mk@ to ARC-mode and here are the Gradle errors :
>
> [...]

This is typical issue with interaction between ARC-enabled and ARC-disabled code. Shortly speaking, compiler don't know how to manage pointers to Objective-C classes when they are crossing ObjC code boundaries, so you need to explicitly say him how to deal with that. Please read official "Apple's tutorial":https://developer.apple.com/library/ios/releasenotes/ObjectiveC/RN-TransitioningToARC/Introduction/Introduction.html and "clang's ARC":http://clang.llvm.org/docs/AutomaticReferenceCounting.html document (speficially, "bridged casts": http://clang.llvm.org/docs/AutomaticReferenceCounting.html#bridged-casts section, but reading whole document is needed too).

> What do you think : (a) SWIG is not intended for this usage, (b) it’s an issue from Crystax, or (c) an issue for me (replace all Obj-C returned objects with compatibles ones) ?

I can't say for sure, since I don't see your code, but it looks very much like a defect of SWIG, which don't know anything about ARC.

> PS:
> It could be a good idea to implement a "jni.h" alternative

We're actively working on providing full comprehensive set of native API, allowing developers don't touch Java at all, unless they really need it. But this is still work-in-progress, so don't expect it soon; on the other hand, CrystaX NDK is open source project so any contribution is always welcome.