大家好，我是腾讯Bugly的精神哥（英文名：spirit），是Bugly资深码奴的同时，又是Bugly神秘的Crash实验室研究员哦！我的主要任务就是泡在实验室里，嗑着瓜子嚼着鸡爪，研究移动App中各种Crash（专挑疑难、坑爹、时髦、有趣的Crash），并通过“**精神哥讲Crash**”系列定期分享给大家！

今天精神哥给大家分享的第一个Crash是“**UnsatisfiedLinkError**”。

一、UnsatisfiedLinkError基本介绍

全名	java.lang.UnsatisfiedLinkError
官方解释	Throw if the java Virtural Machine cannot find an appropriate native-language definition of method declared native意思就是JVM找不到native method的native实现！
影响力排名	出错量排名第16位
精神哥点评	抛出这异常，肯定是你加载SO的姿势不对！

现在App很多功能都是通过集成第三方工具实现的，第三方工具很有可能在SO动态库里实现核心功能（**Bugly提供的libBugly.so，能捕获这类C/C++异常！**），所以就算你不用NDK开发也一定会跟SO打交道，你确定你加载SO的姿势都对了吗？

二、错误场景分析

1、低级错误——根本木有SO，你加载个球啊！

code	System.loadLibrary(Bugly);
libs	空
运行设备	Android ARM设备
运行结果	Crash！java.lang.UnsatisfiedLinkError: dalvik.system.PathClassLoader[DexPathList[[zip file “/data/app/com.tencent.bugly.demo-1/base.apk”],nativeLibraryDirectories=[/vendor/lib, /systemb]]] couldn’t find “libBugly.so”
原因分析	apk安装时，系统会把apk中libs目录下armeabi的SO拷贝到应用的私有目录下。所以libs里没有放入SO，运行时肯定找不到SO。
修复方式	添加SO：libs\armeabi\libBugly.so或加载代码注释掉：//System.loadLibrary(Bugly)；

2、进阶错误——根本木有X86的SO，在X86的设备上你加载个球啊！

code	System.loadLibrary(Bugly);
libs	libs\armeabi\libBugly.so
运行设备	Android X86设备
运行结果	Crash！java.lang.UnsatisfiedLinkError: dalvik.system.PathClassLoader[DexPathList[[zip file “/data/app/com.tencent.bugly.demo-1/base.apk”],nativeLibraryDirectories=[/vendor/lib, /systemb]]] couldn’t find “libBugly.so”
原因分析	apk安装时，x86设备上系统会把apk中libs目录下x86的SO，拷贝到应用的私有目录下。虽然libs下有armeabi的SO，但没有放入x86的SO，运行时还是找不到libbugly.so。
修复方式	添加SO：libs\x86\libBugly.so或加载代码注释掉：//System.loadLibrary(Bugly)；

3、大坑——尼玛，好难发现！

code	if(getArch().contain(“arm”)){//只在arm下加载System.loadLibrary(Bugly) ;System.loadLibrary(Bugly2);
	
	}

libs	libs\armeabi\libBugly.solibs\armeabi\libBugly2.solibs\armeabi-v7a\libBugly.so
运行设备	Android ARMv7设备
运行结果	Crash！java.lang.UnsatisfiedLinkError: dalvik.system.PathClassLoader[DexPathList[[zip file “/data/app/com.tencent.bugly.demo-1/base.apk”],nativeLibraryDirectories=[/vendor/lib, /systemb]]] couldn’t find “libBugly2.so”
原因分析	apk安装时，系统会把apk中libs目录下armeabi-v7a整个目录下的SO拷贝到应用的私有目录下。因为armeabi-v7a下没有放入libBugly2.so，运行时找不到libBugly2.so。不同的工具兼容的CPU架构不一致，就容易出这个错误了！例如： libBugly.so提供armeabi、armeabi-v7a、x86三种。 但其它产品可能只提供了armeabi。 如果把这些so都直接拷贝进apk，就会因为上述的原因直接crash，会误以为该Crash是因为不同产品的so不能兼容导致的！
修复方式	添加SO：libs\armeabi-v7a\libBugly2.so或直接删除armeabi-v7a目录，arm设备上系统会自动选择armeabi

4、天坑——尼玛，巨难发现！

虽然出错原因很简单，但犯错的人很确实多，这货都挤到Bugly Crash影响力第16位了！

精神哥发现**java.lang.UnsatisfiedLinkError**中**couldn’t find “XX.so”**的占比非常高，上面提的三个场景都是这种错误！

但你见过下面这种错误吗？

java.lang.UnsatisfiedLinkError:dlopen failed: “/*/arm/*.so” has unexpected e_machine: 3**

这是天坑啊，肯定是实习生挖的！如何出现的呢？

code	if(getArch().contain(“arm”)){//只在arm下加载System.loadLibrary(Bugly) ;}
libs	libs\armeabi\libBugly.so **坑爹实习生放入了x86编译的libBugly.so(同名很容易出错)**
运行设备	Android ARM设备
运行结果	Crash！java.lang.UnsatisfiedLinkError: dlopen failed: “/data/app/com.tencent.bugly.crashreport.demo-2/lib/arm/libBugly.so” has unexpected e_machine: 3
原因分析	apk安装时，系统把armeabi下的libBugly.so放入应用的私有目录中了！但这个libBugly.so不是arm的，而是x86编译的libBugly.so运行时，系统检察ELF文件中的e_machine字段的值，跟arm的不匹配，就会抛出这个异常了！

“精神哥讲Crash”第一期圆满结束，感谢大家支持，也敬请期待下次分享！