



NDK w Android Studio 2.2

Jakub Czekański

O mnie



@JaCzekanski



github.com/JaCzekanski



ja.czekanski@gmail.com



<http://czekanski.info>

Czym jest NDK

- **N**ative **D**evelopment **K**it
- Zbiór narzędzi umożliwiających kompilację kodu C i C++
- Udostępnia część API Androida dla tych języków
- Nie zastępuje SDK
- ... ale umożliwia napisanie prawie całej aplikacji bez Javy (NativeActivity)

Zastosowania

- Używanie istniejących już bibliotek napisanych w C
- Aplikacje wymagające dużej wydajności (OpenGL, Vulkan) czy niskich opóźnień (muzyka)
- Portowanie wcześniej napisanej aplikacji z innej platformy

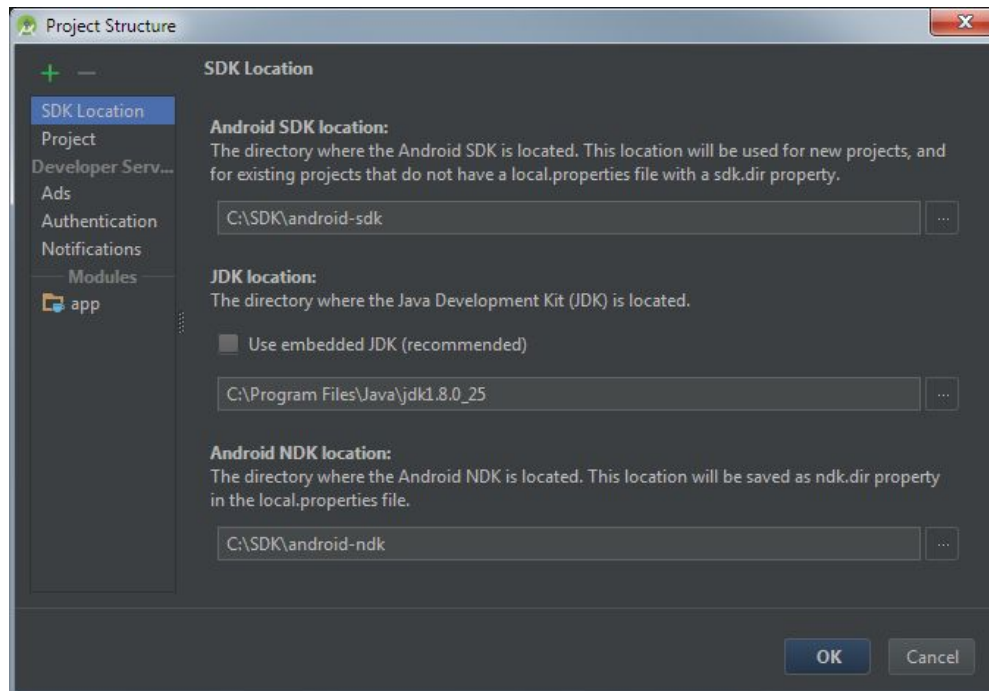
Instalacja


NDK Downloads

Platform	Package	Size (Bytes)	SHA1 Checksum
Windows 32-bit	android-ndk-r13-windows-x86.zip	620325945	cc498ef01d7fe919dcc8aeb4d709f4ff793dee46
Windows 64-bit	android-ndk-r13-windows-x86_64.zip	681188845	39b78dda640f7f2647075d44b0ec1a8ac6ce5eef
Mac OS X	android-ndk-r13-darwin-x86_64.zip	665841170	fd7ec2b511b66c479487138be5413400521edd0e
Linux 64-bit (x86)	android-ndk-r13-linux-x86_64.zip	687179374	a22bfcbe467103e21acc953b0c11158941ab49ee

Select, from the table above, the NDK package for your development platform. For information about the changes in the newest version of the NDK, see [Release Notes](#). For information about earlier revisions, see [NDK Revision History](#).

Konfiguracja



 Edit Tool ✕

Name: javah

Group: NDK ▾

Description: javah

Options

☒ Synchronize files after execution

☒ Open console

Output Filters...

☒ Show console when a message is printed to standard output stream

☒ Show console when a message is printed to standard error stream

Show in

☒ Main menu

☒ Editor menu

☒ Project views

☒ Search results

Tool settings

Program: "\$JDKPath\$/bin/javah" ...

Insert macro...

Parameters: "-classpath "\$Classpath\$" -v -jni "\$FileClass\$"

Insert macro...

Working directory: \$SourcepathEntry\$/..\jni

...

Insert macro...

OK

Cancel

Help

```
public class NativeLib {  
    public native int get42();  
    public native String getHello();  
}
```


// MainActivity.java

// (...)

NativeLib nativeLib = **new** NativeLib();

Log.d(**TAG**, "**get42 returned** " + nativeLib.get42());

Log.d(**TAG**, nativeLib.getHello());

Ndk test has stopped



Open app again



Mute until device restarts

Ndk test has stopped



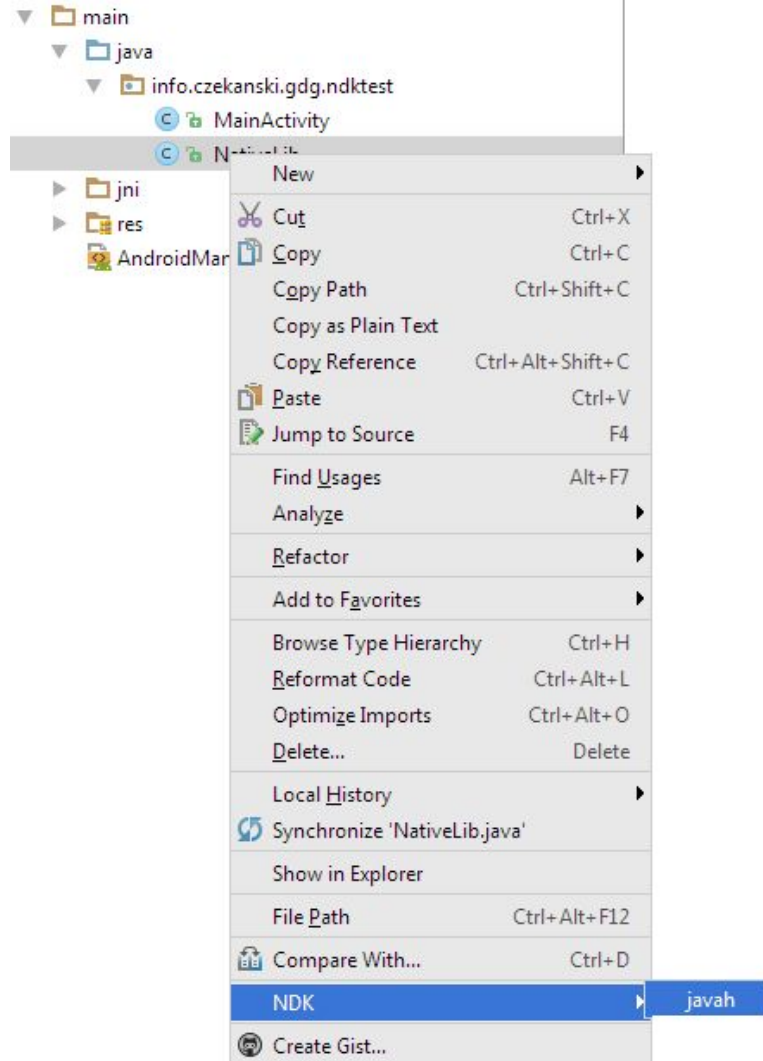
Open app again



Mute until device restarts

`java.lang.UnsatisfiedLinkError:`

No implementation found for int
`info.czekanski.gdg.ndktest.NativeLib.get42()`



```
/* DO NOT EDIT THIS FILE - it is machine generated */
#include <jni.h>
/* Header for class info_czekanski_gdg_ndktest_NativeLib */

#ifdef _Included_info_czekanski_gdg_ndktest_NativeLib
#define _Included_info_czekanski_gdg_ndktest_NativeLib
#ifdef __cplusplus
extern "C" {
#endif
/*
 * Class:      info_czekanski_gdg_ndktest_NativeLib
 * Method:     get42
 * Signature:  ()I
 */
JNIEXPORT jint JNICALL Java_info_czekanski_gdg_ndktest_NativeLib_get42 (JNIEnv *, jobject);

/*
 * Class:      info_czekanski_gdg_ndktest_NativeLib
 * Method:     getHello
 * Signature:  ()Ljava/lang/String;
 */
JNIEXPORT jstring JNICALL Java_info_czekanski_gdg_ndktest_NativeLib_getHello (JNIEnv *, jobject);

#ifdef __cplusplus
}
#endif
#endif
```

```
#include "info_czekanski_gdg_ndktest_NativeLib.h"
#include <android/log.h>

#define LOG_TAG "NdkTest_nativeLib"
#define LOGI(...) __android_log_print(ANDROID_LOG_INFO, LOG_TAG, __VA_ARGS__)

JNIEXPORT jint JNICALL Java_info_czekanski_gdg_ndktest_NativeLib_get42
(JNIEnv *env, jobject obj)
{
    LOGI("In native function");
    return 42;
}

JNIEXPORT jstring JNICALL Java_info_czekanski_gdg_ndktest_NativeLib_getHello
(JNIEnv *env, jobject obj)
{
    char msg[] = "Hello native world!";
    jstring result = env->NewStringUTF(msg);

    return result;
}
```

Systemy budowania

- ndk-build (stary, nadal wspierany)
- CMake (zalecany przez Google)

Android.mk

```
LOCAL_PATH := $(call my-dir)
```

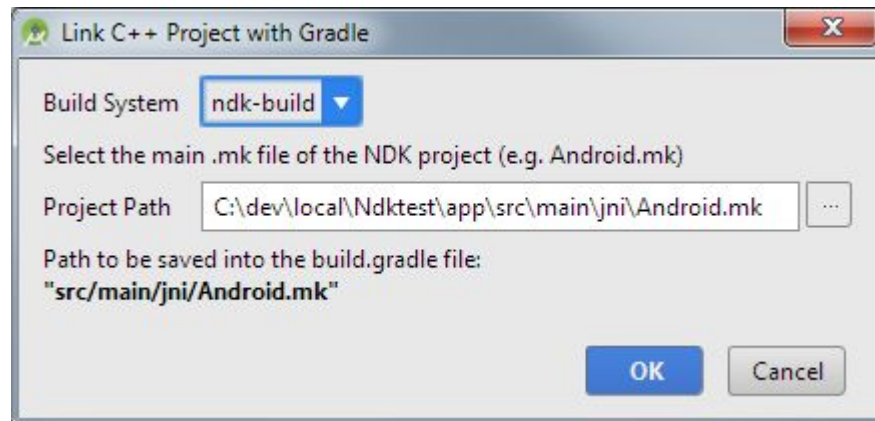
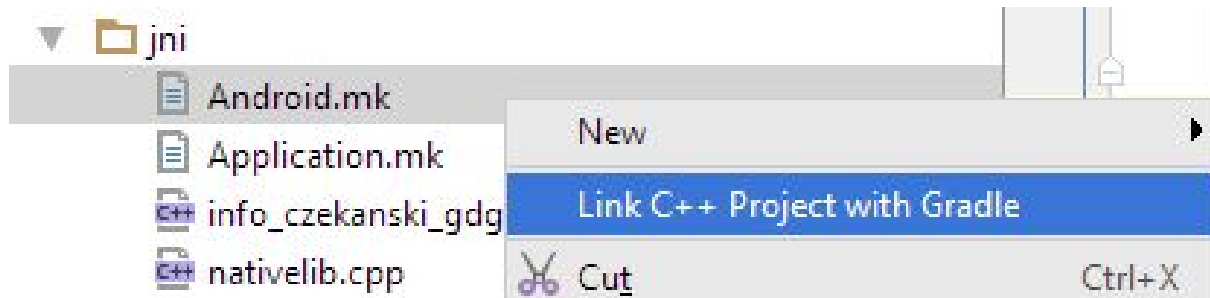
```
include $(CLEAR_VARS)
```

```
LOCAL_MODULE      := nativelib  
LOCAL_SRC_FILES   := nativelib.cpp  
LOCAL_CPP_FEATURES += exceptions  
LOCAL_LDLIBS      := -llog
```

```
include $(BUILD_SHARED_LIBRARY)
```


Application.mk

```
APP_STL := gnustdl_static
```



build.gradle

```
android {  
    // (...)  
  
    externalNativeBuild {  
        ndkBuild {  
            path 'src/main/native/Android.mk'  
        }  
    }  
}
```

build.gradle

```
android {  
    defaultConfig {  
        // (...)  
  
        ndk {  
            abiFilters 'x86_64' // 'x86', 'armeabi-v7a', 'arm64-v8a'  
        }  
    }  
}
```

Na czas developmentu kompilujemy tylko jedną platformę aby przyspieszyć build.

Kompilujemy i ...

Ndk test keeps stopping



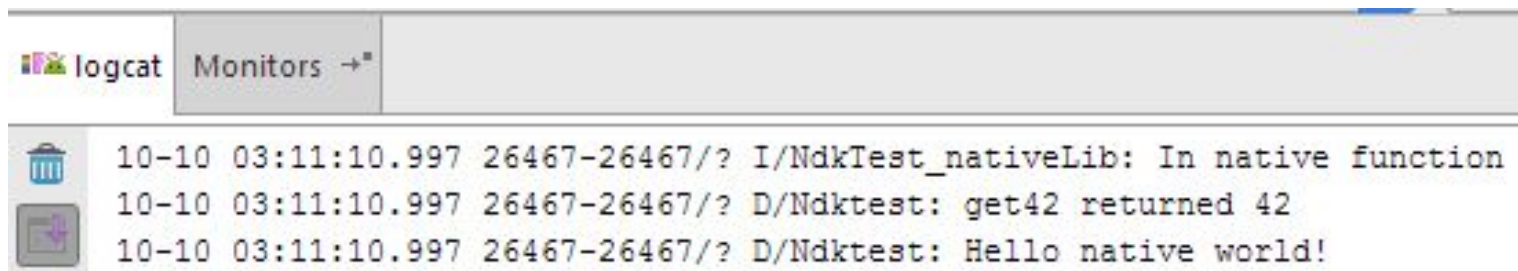
Close app



Mute until device restarts

```
public class NativeLib {  
    static {  
        System.LoadLibrary("nativeLib");  
    }  
    public native int get42();  
    public native String getHello();  
}
```

Logcat - sukces!



Kompilacja poza IDE

```
@echo off
```

```
set NDK_PROJECT_PATH=c:\dev\local\Ndktest\app\src\main  
c:\sdk\android-ndk\ndk-build.cmd
```

```
APP_ABI="x86-64"
```

```
NDK_LIBS_OUT=./libs
```

```
NDK_OUT=./obj
```

```
--directory=app/.externalNativeBuild/ndkBuild/debug
```

```
-j8
```

- APP_ABI - architektura
- -j - jobs (ilość wątków)

Inny przykład

```
public native synchronized TrackInfo getTrackInfo( int track);
```

```
// ...
```

```
@Data
```

```
public class TrackInfo {  
    public int track_number;  
    public int track_count;  
    public int length;  
    public String system;  
    public String game;  
    public String song;  
}
```

```
JNIEXPORT jobject JNICALL Java_com_czekanski_chipplayer_libs_GME_getTrackInfo
(JNIEnv *env, jobject obj, jint track)
{
    if (!emu) return NULL;

    gme_info_t *info;
    gme_track_info(emu, &info, track);

    jclass clazz = env->FindClass("com/czekanski/chipplayer/libs/TrackInfo");
    jobject ti = env->AllocObject(clazz);

    env->SetIntField(ti, env->GetFieldID(clazz, "track_number", "I"), info->track_number);
    env->SetIntField(ti, env->GetFieldID(clazz, "track_count", "I"), info->track_count);
    env->SetIntField(ti, env->GetFieldID(clazz, "length", "I"), info->length);
    env->SetObjectField(ti, env->GetFieldID(clazz, "system", "Ljava/lang/String;"), env->NewStringUTF(info->system));
    env->SetObjectField(ti, env->GetFieldID(clazz, "game", "Ljava/lang/String;"), env->NewStringUTF(info->game));
    env->SetObjectField(ti, env->GetFieldID(clazz, "song", "Ljava/lang/String;"), env->NewStringUTF(info->song));

    return ti;
}
```

JNI

- Duża ilość kodu
- Bardzo łatwo o błąd (zły typ, zła klasa)
- Zmiany w kilku miejscach (Java, .c, .h)

Rozwiązania

SWIG (Simplified Wrapper and Interface Generator)

- Generuje wrapper na podstawie kodu w C
- Używa plików .i wraz z zestawem makr do konfiguracji
- Generuje pliki .java na podstawie .i
- Wspiera struktury, klasy, szablony
- Nie tylko Java (Python, Ruby, Go, ...)

pi.c

```
#include <math.h>

double calculatePi(int iterationCount)
{
    const double SQRT2 = sqrt(2.0);
    double a = 1, b = 1/SQRT2, t = .25, p = 1;
    double an, pi = 1;
    int iteration = 0;
    do {
        ++iteration;
        an = .5 * (a + b);
        b = sqrt(a * b);
        t -= p * (a - an) * (a - an);
        a = an;
        p *= 2;
        pi = (a + b) * (a + b) / (4 * t);
    } while (iteration < iterationCount);

    return pi;
}
```

pi.i

```
%module pi
```

```
%inline %{
```

```
extern double calculatePi(int iterationCount);
```

```
%}
```

swig

```
c:\sdk\swig\swig
```

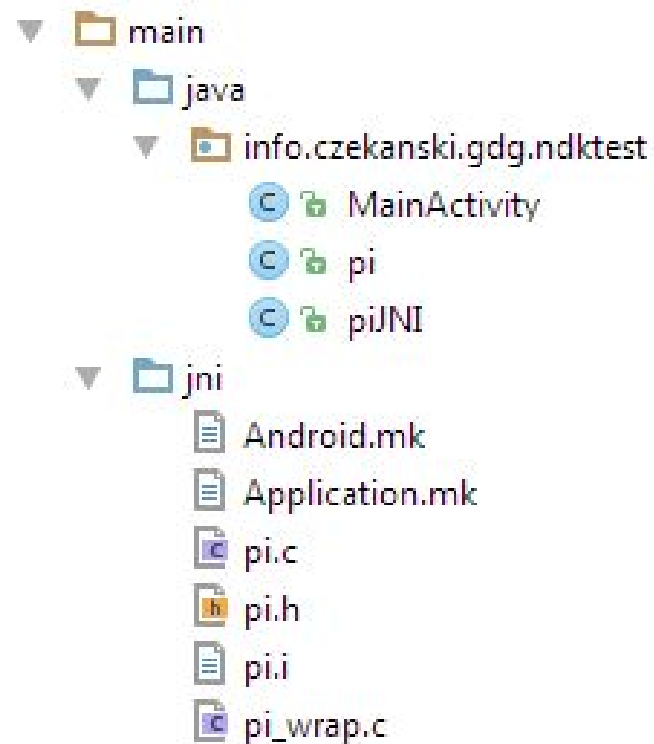
```
-java
```

```
-package info.czekanski.gdg.ndktest
```

```
-outdir java/info/czekanski/gdg/ndktest
```

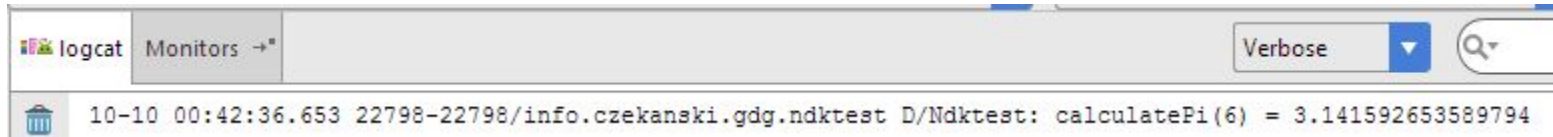
```
-o jni\pi_wrap.cpp
```

```
jni\pi.i
```




```
// (...)
```

```
Log.d(TAG, "calculatePi(6) = "+pi.calculatePi(6));
```



Rozwiązania

JNA (Java Native Access)

- Generuje w locie wrapper na bibliotekę
- Używa interfejsu do importu funkcji
- Brak potrzeby pisania kodu natywnego
- Używa JNI
- Narzut wydajnościowy

Przykład użycia JNA

```
public class ExampleOfPOSIX {  
    public interface POSIX extends Library {  
        public int chmod(String filename, int mode);  
        public int chown(String filename, int user, int group);  
        public int rename(String oldpath, String newpath);  
        public int kill(int pid, int signal);  
        public int link(String oldpath, String newpath);  
        public int mkdir(String path, int mode);  
        public int rmdir(String path);  
    }  
  
    public static void main(String[] args) {  
        POSIX posix = (POSIX) Native.loadLibrary("c", POSIX.class);  
        posix.mkdir("/tmp/newdir", 0777);  
        posix.rename("/tmp/newdir", "/tmp/renamedir");  
    }  
}
```

Rozwiązania

JavaCPP

- Konfiguracja adnotacjami po stronie Javy
- Kod natywny pozostaje prawie bez zmian
- Wspiera wiele funkcjonalności C++

JavaCppTest.java

```
@Platform(library = "testLib", include = {"test.h"})  
public class JavaCppTest {  
    static { Loader.Load(); }  
    public static native double calculatePi(int iterationCount);  
}
```

```
// test.h
#pragma once
extern double calculatePi(int iterationCount);
```

```
// test.cpp
#include "test.h"
#include <cmath>

double calculatePi(int iterationCount)
{
    // (...)
}
```

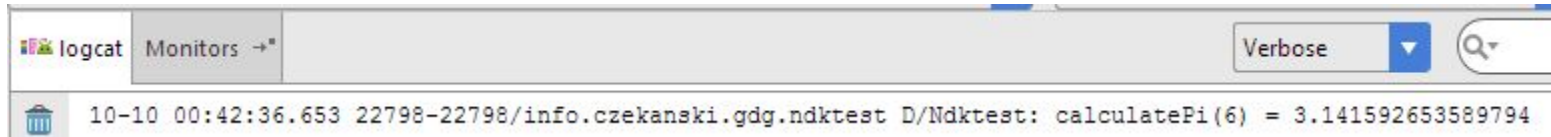
javacpp

java

```
-jar c:\SDK\javacpp\javacpp.jar  
-cp ../../build/intermediates/classes/debug  
-d jni  
-nocompile  
info.czekanski.gdg.ndktest.JavaCppTest
```

```
// (...)
```

```
Log.d(TAG, "calculatePi(6) = "+JavaCppTest.calculatePi(6));
```



Kotlin

- external zamiast native
- Javah nie widział klas
- Cała aplikacja w Kotlinie, stuby do komunikacji z jni w Javie

Podsumowanie

- Komunikacja z kodem natywnym
- Przyspieszenie kodu, obfuskacja, integracja bibliotek
- Portowanie gier/aplikacji (SDL, OpenGL)
- JNI ma swój narzut
- Istnieją biblioteki ułatwiające integracje kodu natywnego
- [Protocol Buffers](#), [Apache Thrift](#)



 **Rzethon**
by **G2A.COM**

10-11.12.2016

www.rzethon.pl

Dziękuję za uwagę



@JaCzekanski