2021

第11章 Android NDK开发

复旦大学 陈辰

学习目标

AIMS

01 了解Android NDK的用途和不足

02 掌握Android NDK编译环境的安装与配置方法

03 掌握Android NDK的开发步骤

11.1 NDK简介

- Android NDK
  - Android NDK（Android Native Development Kit）是一系列的开发工具，允许程序开发人员在Android应用程序中嵌入C或C++语言编写的本地代码
  - Android NDK优势
    - 解决了核心模块使用托管语言开发执行效率低下的问题
    - 允许直接使用C/C++源代码，极大的提高了Android应用程序开发的灵活性
  - Android NDK不足
    - Android NDK并不会自动提升所有Android程序的执行效率，但一定会增加程序的复杂程度和调试难度

11.1 NDK简介

- Android NDK
  - 程序开发人员需要仔细权衡Android NDK所能提升的运行效率与增加的复杂程度是否在可接受的范围内。选择使用Android NDK应主要出于以下两种目的
    - 一是Android应用程序框架无法满足运行效率时
    - 二是需要使用大量已有C/C++源代码

11.1 NDK简介

- Android NDK
  - Android NDK提供一系列的工具、编译文件、文档和示例代码，用于从C/C++源代码中生产本地代码库，还提供了将本地代码库嵌入到apk文件的方法
  - Android NDK所包含的大量本地系统头文件和库文件，主要是用来支持未来版本的Android系统
  - Android NDK所支持的最低版本的Android系统是1.5版本，如果使用本地Activity则所需要的最低Android系统版本为2.3版本

11.1 NDK简介

- Android NDK
  - 最新版本的Android NDK支持ARM指令集，包括ARMv5TE、ARMv7-A、ARMv8、x86和x86_64
  - ARMv5TE机器码可以在所有基于ARM的Android设备上使用，ARMv7-A机器码则只能运行在具有ARM7 CPU的Android设备上，如Verizon Droid手机和Google Nexus One手机
  - ARMv7-A与ARMv5TE指令集的差别主要在于，ARMv7-A支持硬件FPU（浮点运算单元）、Thumb-2和NEON指令集
  - 程序开发人员可以针对不同目标设备，在Android NDK中使用不同的ARM指令集支持不同的架构，也可以同时将支持多个架构的指令集编译到同一个apk文件中

11.2 NDK开发环境

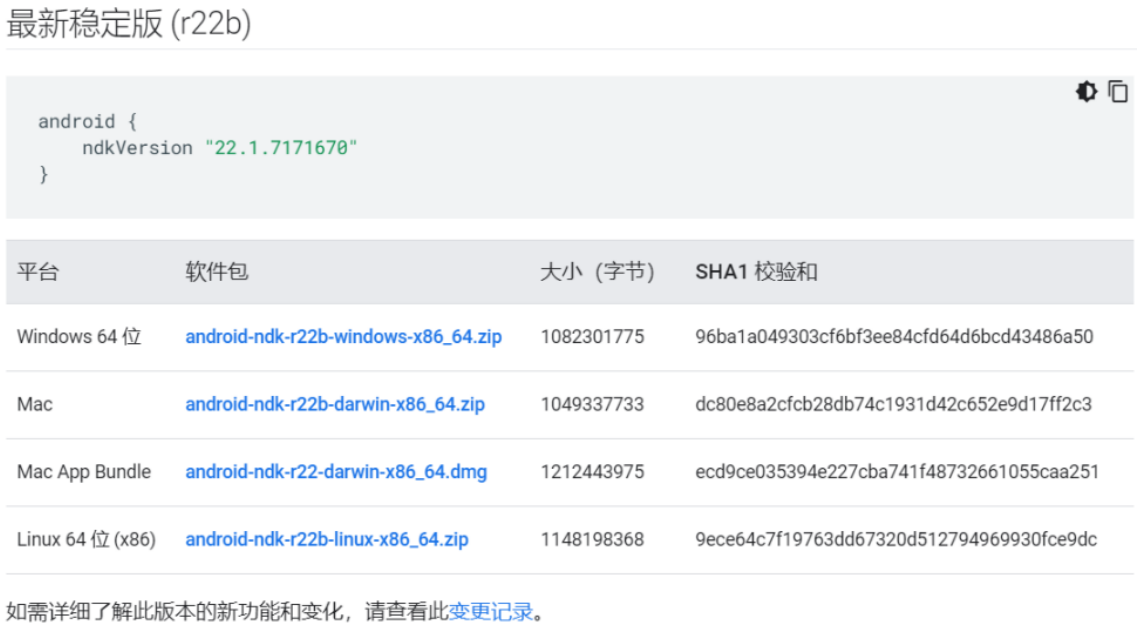
- NDK开发环境包括Android Studio、Android NDK和Cygwin
  - Android Studio用来建立Android工程和编写程序代码
  - Android NDK提供编译脚本和工具
  - Cygwin完成Linux环境下的交叉编译，将C/C++的源代码文件编译成Android系统可调用的共享连接库文件

11.2 NDK开发环境

- Android NDK编译环境支持Windows XP、Linux和MacOS，本书仅介绍Windows系统的编译环境配置方法
- Windows系统的编译环境配置方法
  - 下载Android NDK的安装包
  - 下载并安装Cygwin

11.2 NDK开发环境

- 下载Android NDK的安装包
  - Google的Android开发者网站下载Android NDK安装包，下载地址是https://developer.android.com/ndk/downloads，下载页面如下：



最新稳定版 (r22b)

```
android {
    ndkVersion "22.1.7171670"
}
```

平台	软件包	大小（字节）	SHA1 校验和
Windows 64 位	android-ndk-r22b-windows-x86_64.zip	1082301775	96ba1a049303cf6bf3ee84cfd64d6bcd43486a50
Mac	android-ndk-r22b-darwin-x86_64.zip	1049337733	dc80e8a2cfcb28db74c1931d42c652e9d17ff2c3
Mac App Bundle	android-ndk-r22-darwin-x86_64.dmg	1212443975	ecd9ce035394e227cba741f48732661055caa251
Linux 64 位 (x86)	android-ndk-r22b-linux-x86_64.zip	1148198368	9ece64c7f19763dd67320d512794969930fce9dc

如需详细了解此版本的新功能和变化，请查看此变更记录。

11.2 NDK开发环境

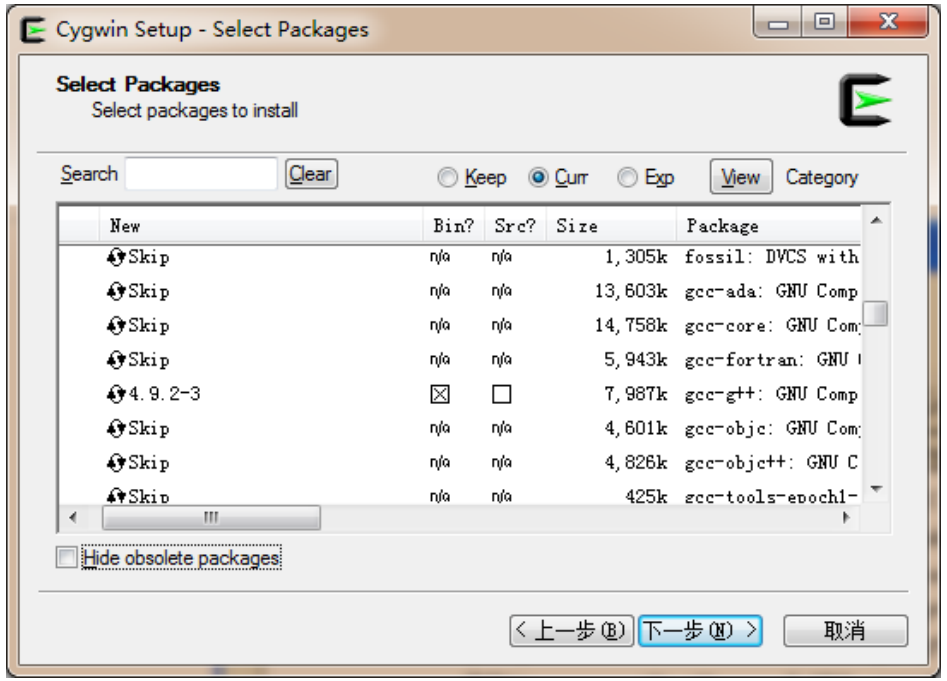
- 下载Android NDK的安装包
  - 笔者下载的Android NDK是Windows的r22b版本，下载的文件为android-ndk-r22b-windows-x86_64.zip。笔者将Android NDK解压到D:\目录中，Android NDK的最终路径为D:\android-ndk-r22b

11.2 NDK开发环境

- 下载并安装Cygwin
  - Android NDK目前可以在Windows系统上使用CMake进行交叉编译，也可以在Windows系统中安装Linux的模拟器环境Cygwin，完成C/C++代码的交叉编译工作
  - Android NDK要求Cygwin的版本高于1.7，因此最好安装较新版本的Cygwin
  - Cygwin最新版本可以到官方网站http://www.cygwin.com下载

11.2 NDK开发环境

- 下载并安装Cygwin
  - 在Cygwin的安装过程中，需要将Devel下的gcc和make的相关选项选上，如下图所示，否则Cygwin将无法编译C/C++源代码文件



11.3 NDK文档

- Android NDK的目录结构
  - 介绍NDK开发前，首先熟悉一下Android NDK为程序开发人员提供的资料和示例。Android NDK的目录中包含6个子目录和7个文件
    - build目录保存了编译脚本和配置文件
    - docs目录是帮助文档目录
    - platforms是保存了编译过程中可能用到的头文件和库文件，并根据Android版本和CPU类型进行了分类

(+)build
(+)docs
(+)platforms
(+)prebuilt
(+)sources
(+)tests
(+)toolchains
(-)find-win-host.cmd
(-)GNUmakefile
(-)ndk-build
(-)ndk-build.cmd
(-)ndk-depends.exe
(-)ndk-gdb
(-)ndk-gdb.py
(-)ndk-gdb-py
(-)ndk-gdb-py.cmd
(-)ndk-statck.exe
(-)ndk-which
(-)README.txt
(-)RELEASE.txt
(-)remove-windows-symlink.sh

11.3 NDK文档

- Android NDK的目录结构
  - sources目录中保留了程序中可能用到的C/C++源代码文件，CPU类型检查和本地Activity的C/C++源代码文件就在这个目录中
  - tests是测试代码目录
  - toolchains是交叉编译工具目录

(+)build
(+)docs
(+)platforms
(+)prebuilt
(+)samples
(+)sources
(+)tests
(+)toolchains
(-)find-win-host.cmd
(-)GNUmakefile
(-)ndk-build
(-)ndk-build.cmd
(-)ndk-depends.exe
(-)ndk-gdb
(-)ndk-gdb.py
(-)ndk-gdb-py
(-)ndk-gdb-py.cmd
(-)ndk-statck.exe
(-)ndk-which
(-)README.txt
(-)RELEASE.txt
(-)remove-windows-symlink.sh

11.3 NDK文档

- Android NDK的目录结构
  - GNUmakefile编译配置文件
  - ndk-build是交叉编译的快捷脚本
  - ndk-gdb用于Debug调试的脚本
  - README.txt和RELEASE.txt分别是Android NDK的说明文档和版本信息

(+)build
(+)docs
(+)platforms
(+)prebuilt
(+)samples
(+)sources
(+)tests
(+)toolchains
(-)find-win-host.cmd
(-)GNUmakefile
(-)ndk-build
(-)ndk-build.cmd
(-)ndk-depends.exe
(-)ndk-gdb
(-)ndk-gdb.py
(-)ndk-gdb-py
(-)ndk-gdb-py.cmd
(-)ndk-statck.exe
(-)ndk-which
(-)README.txt
(-)RELEASE.txt
(-)remove-windows-symlink.sh

11.3 NDK文档

- Android NDK的目录结构
  - docs目录中的帮助文件说明

文件名	说明
OVERVIEW.html	Android NDK的概括性说明，包括NDK的目标、适用范围、开发步骤和NDK关键配置文件的简要说明等
INSTALL.html	NDK的安装与配置说明文档
DEVELOPMENT.html	说明如何对NDK进行修改，以及如何发布新的实验性NDK包
HOWTO.html	关于NDK通用性问题的说明
ANDROID-MK.html	说明构建Android.mk文件的语法格式。Android.mk定义了模块的编译信息，包括模块（module）名称、与C/C++源代码文件的对应关系
APPLICATION-MK.html	说明构建Application.mk文件的语法格式。Application.mk定义了应用程序的编译信息，包括CPU体系类型、模块列表、编译器的参数等
CPU-ARCH-ABIS.html	处理器ABIS（应用程序二进制接口）说明文档
CPU-ARM-NEON.html	ARM处理器NEON扩展指令集说明文档

11.3 NDK文档和示例

- Android NDK的目录结构
  - docs目录中的帮助文件说明

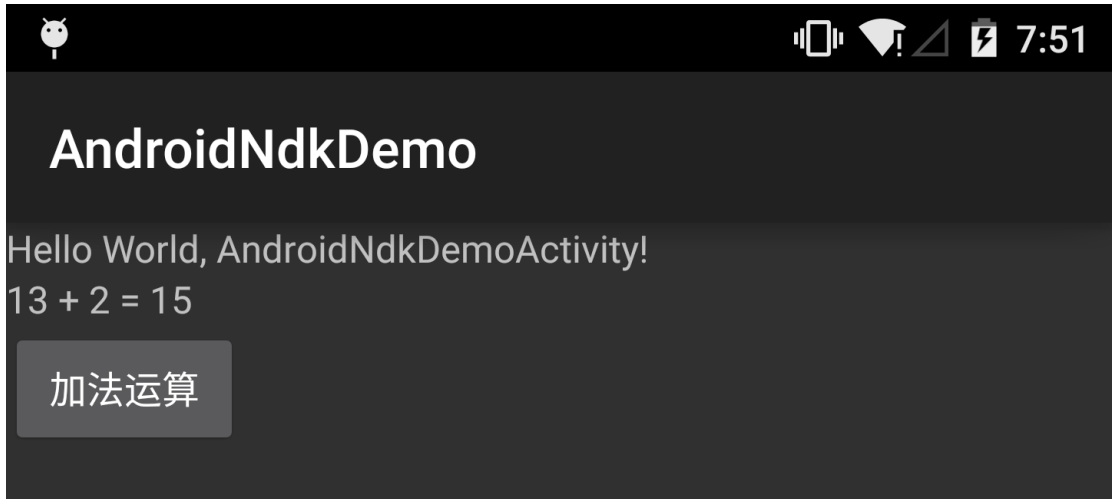
文件名	说明
CPU-FEATURES.html	处理器类型和指令集特征的检查说明文档
IMPORT-MODULE.html	说明如何在Android.mk中引用其他模块，以及建立引用模块的方法
NDK-BUILD.html	如何使用ndk-build脚本进行编译
NDK-GDB.html	关于NDK通用性问题的说明
PREBUILTS.html	如何制作预编译库文件
STABLE-APIS.html	支持的稳定的API类表
STANDALONE-TOOLCHAIN.html	如何将NDK提供的交叉编译工具作为独立的编译器使用
system/libc/OVERVIEW.html	Bionic C库的简介
system/libc/SYSV-IPC.html	介绍NDK不支持system v进程间通信的原因
system/libc/CHANGES.html	不同版本下Bionic的区别
CHANGES.html	不同版本NDK的区别
SYSTEM-ISSUES.html	NDK开发所需要注意的问题
LICENSES.html	NDK的使用许可

11.4 NDK示例

- 在进行NDK开发时，一般先要建立Android工程，在Android工程中创建存放C/C++代码的jni目录
- 然后在Cygwin环境中编译C/C++代码，NDK的编译脚本会在Android工程中自动建立libs目录，将编译后形成的共享库文件保存在libs目录中
- 最后，在编译Android工程时，libs中的共享库文件会被打包到apk文件中，保证Android程序可以正常运行

11.4 NDK示例

- 下面的内容以AndroidNdkDemo为例来说明如何进行Android NDK开发。AndroidNdkDemo是一个加法运算的示例，程序会随机生产两个整数，然后调用C语言开发的共享库对这两个整数进行加法运算，最后将运算结果显示在用户界面上。AndroidNdkDemo示例的界面如下图所示：



11.4 NDK示例

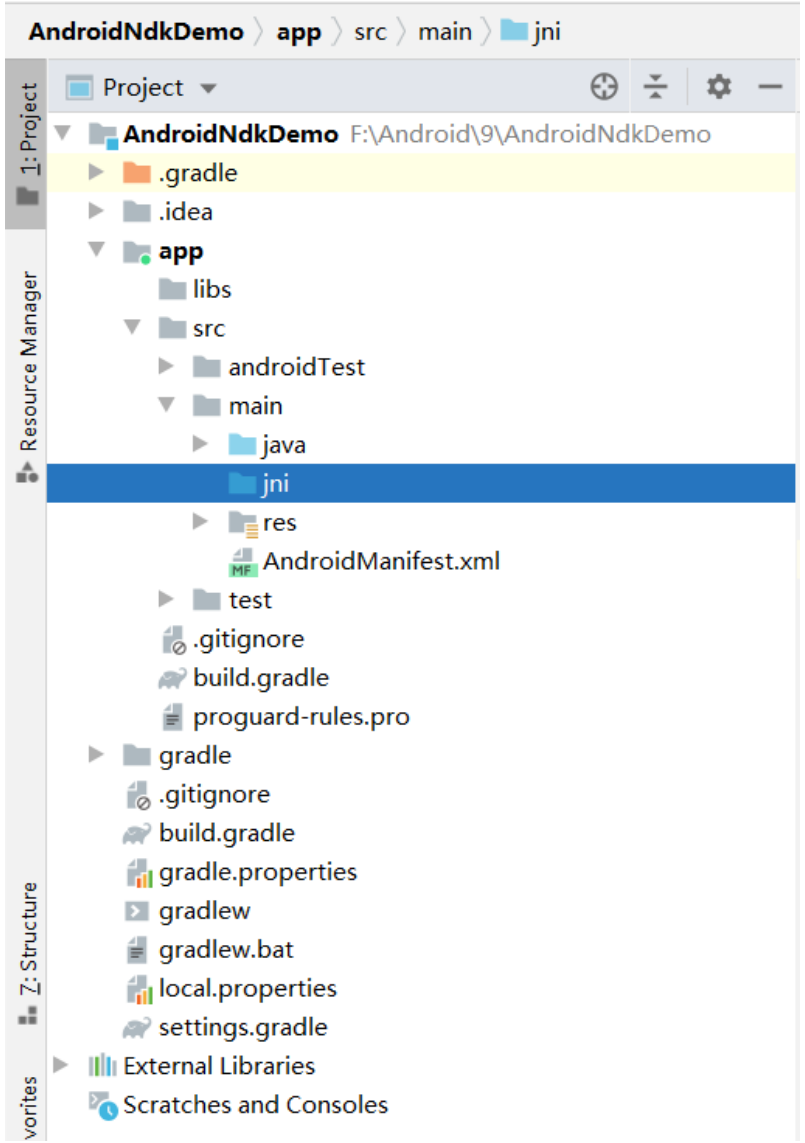
- 进行Android NDK开发一般要经过如下的步骤:
  - 建立Android工程
  - 建立Android.mk文件
  - 建立C源代码文件
  - 编译共享库模块
  - 运行Android程序

11.4 NDK示例

- 建立Android工程
  - 首先在Android Studio中建立Android工程时，工程名称为AndroidNdkDemo，并在工程的app/src/main目录中建立一个新目录jni，用来保存C/C++代码文件
  - jni的子目录结构不必遵循Java代码的目录结构，如com.<mycompany>.<myproject>，可以将所有的C/C++代码文件放置在jni目录下，也可以创建子目录保存，并不影响最后的编译结果

11.4 NDK示例

- 建立Android工程
  - AndroidNdkDemo工程的目录结构如下图所示：



11.4 NDK示例

- 建立Android工程
  - 这个示例中采用“自顶向下”的方式进行开发，首先编写Anroid程序的用户界面
  - 然后开发C/C++的共享库
  - 为了调试方便，先在Java代码中编写一个功能相近函数，在用户界面调试中使用，当完成C/C++的共享库开发后，再用共享库中的函数替代这个Java代码函数
  - 在建立AndroidNdkDemo工程后，修改main.xml文件，添加一个id为display的TextView和一个id为add_btn的Button按钮
  - 程序中的产生随机数和调用的代码在AndroidNdkDemoActivity.java文件中，下面是AndroidNdkDemoActivity.java文件的核心代码

11.4 NDK示例

- AndroidNdkDemoActivity.java文件代码:

```
1   public class AndroidNdkDemoAcitivity extends Activity {
2       @Override
3       public void onCreate(Bundle savedInstanceState) {
4           super.onCreate(savedInstanceState);
5           setContentView(R.layout.main);
6           final TextView displayLable = (TextView)findViewById(R.id.display);
7           Button btn = (Button)findViewById(R.id.add_btn);
8           btn.setOnClickListener(new View.OnClickListener(){
9               @Override
10              public void onClick(View v) {
11                  double randomDouble = Math.random();
12                  long x = Math.round(randomDouble*100);
13                  randomDouble = Math.random();
14                  long y = Math.round(randomDouble*100);
```

11.4 NDK示例

- AndroidNdkDemoActivity.java文件代码:

```
15
16                  //System.loadLibrary("add-module");
17                  long  z = add(x, y);
18                  String msg = x+" + " +y+" = "+z;
19                  displayLable.setText(msg);
20              }
21          });
22      }
23      //public native long add(long  x, long  y);
24
25      public long add(long  x, long  y){
26          return x+y;
27      }
28  }
```

11.4 NDK示例

- 在代码第17行本应该调用共享库add()函数，但为了便于开发和调试，在代码第25行到第27行，使用Java代码开发了一个功能相同的add()函数，这样即使没有完成C/C++共享库的开发前，也可以对Android工程进行界面部分的调试
- 第16行和第23行注释掉的代码，就是在C/C++的共享库开发完毕后需要使用的代码，其中第16行是动态加载共享库的代码，加载的共享库名称为add-module
- 动态加载是在调用共享库中的函数前，在程序代码中指明需要加载的模块名称
- 除动态加载以外，程序开发人员还可以使用静态加载的方式，在类加载时加载共享库，代码如下:

```
static{
    System.loadLibrary("add-module");
}
```

11.4 NDK示例

- 建立Android.mk文件
  - Android.mk是jni根目录下必须存在描述C/C++代码文件模块信息的文件，将代码模块的编译信息传递给NDK编译系统，是NDK编译系统编译脚本的一部分
  - 在编写C/C++源代码文件前，首先在jni目录中建立Android.mk文件

11.4 NDK示例

- 建立Android.mk文件
  - Android.mk是jni根目录下必须存在描述C/C++代码文件模块信息的文件，将代码模块的编译信息传递给NDK编译系统，是NDK编译系统编译脚本的一部分
  - 在编写C/C++源代码文件前，首先在jni目录中建立Android.mk文件

11.4 NDK示例

- 建立Android.mk文件
  - 一般情况下，NDK编译系统会搜寻<project>/jni目录中的Android.mk文件，其中<project>是Android的工程目录
  - 但如果程序开发人员将Android.mk文件放置在下一级目录中，则需要在上一级目录中的Android.mk文件中告知NDK编译系统遍历所有子目录中的Android.mk文件，在jni目录下Android.mk文件添加的代码如下：

```
include $(call all-subdir-makefiles)
```

11.4 NDK示例

- 建立Android.mk文件

  下面来分析AndroidNdkDemo示例jni目录下的Android.mk文件。

  Android.mk文件的代码如下：

```
1  LOCAL_PATH := $(call my-dir)
2
3  include $(CLEAR_VARS)
4
5  LOCAL_MODULE    := add-module
6  LOCAL_SRC_FILES := add-module.c
7
8  include $(BUILD_SHARED_LIBRARY)
```

11.4 NDK示例

- 建立Android.mk文件
  - 每个Android.mk文件都必须以第1行代码开始
  - 变量LOCAL_PATH用来定义需要编译的C/C++源代码的位置
  - my-dir由NDK编译系统提供，表示当前目录的位置。在AndroidNdkDemo示例中
  - my-dir表示Android.mk所在的jni目录
  - 代码第3行的include $(CLEAR_VARS)表示清空所有以LOCAL_开始的变量，例如LOCAL_MODULE、LOCAL_SRC_FILES、LOCAL_STATIC_LIBRARIES等等，但第1行定义的LOCAL_PATH不在清空的范围内

11.4 NDK示例

- 建立Android.mk文件
  - 因为所有的编译脚本都将在同一个GNU Make的执行环境中，而且所有变量都是全局变量，因此在每次使用前必须清空所有以前用过的变量
  - 第5行代码变量LOCAL_MODULE用来声明模块名称，模块名称必须唯一，而且中间不能存在空格
  - NDK编译系统将会在模块名称前自动添加lib前缀，然后生产so文件
  - 这里的模块名称为add-module，生产的共享库文件名为libadd-module.so。但需要注意的是，如果程序开发人员使用具有lib前缀的模块名称，NDK编译系统将不再添加前缀，例如模块名称为libsub，生产的共享库文件名为libsub.so

11.4 NDK示例

- 建立Android.mk文件
  - 第6行代码中的变量LOCAL_SRC_FILES表示编译模块所需要使用的C/C++文件列表，但不需要给出头文件的列表，因为NDK编译系统会自动计算依赖关系
  - add-module模块仅需要一个C文件，文件名为add-module.c。缺省情况下，结尾名为.c的文件是C语言源文件，结尾名为.cpp的文件是C++语言源文件
  - 第8行代码include $(BUILD_SHARED_LIBRARY)表示Android NDK编译系统需要构建共享库，如果变量BUILD_SHARED_LIBRARY更改为BUILD_STATIC_LIBRARY，则表示需要NDK编译系统构建静态库

11.4 NDK示例

- 建立Android.mk文件
  - 共享库和静态库文件有着不同的用途，共享库可以被Android工程中的Java代码调用，并打包到apk文件中
  - 静态库不能被Java代码调用，也不能打包到apk文件中，只能在生产共享库的过程中被共享库中的C/C++代码所调用

11.4 NDK示例

- 建立C源代码文件
  - 根据Android.mk文件的声明，add-module模块仅包含一个C源代码文件add-module.c
  - 在jni目录中建立add-module.c文件，在该文件中实现整数加法运算功能，全部代码如下

```
1   #include <jni.h>
2
3   jlong Java_edu_hrbeu_AndroidNdkDemo_AndroidNdkDemoActivity_add( JNIEnv*  env,
                    jobject  this, jlong     x, jlong     y )
4   {
5       return x+y;
6   }
```

11.4 NDK示例

- 建立C源代码文件
  - 代码第1行引入的是JNI（Java Native Interface）的头文件
  - 第3行代码是函数名称，jlong表示Java长型整数，Java_edu_hrbeu_AndroidNdkDemo_AndroidNdkDemo_add的构成为Java_<包名称>_<类>_<函数>，其中<函数>的名称和参数要与Android工程中AndroidNdkDemoActivity.java文件定义的函数一致
  - 第5行代码用来返回加法运算结果

11.4 NDK示例

- 编译共享库模块
  - 到目前为止，编译前的准备工作基本就绪，程序开发人员可以编译C语言开发的共享库模块了
  - 首先启动Cygwin，然后切换到Androd NDK的主目录下，键入如下的编译命令：

  - export NDK=/cygdrive/d/android-ndk-r10e

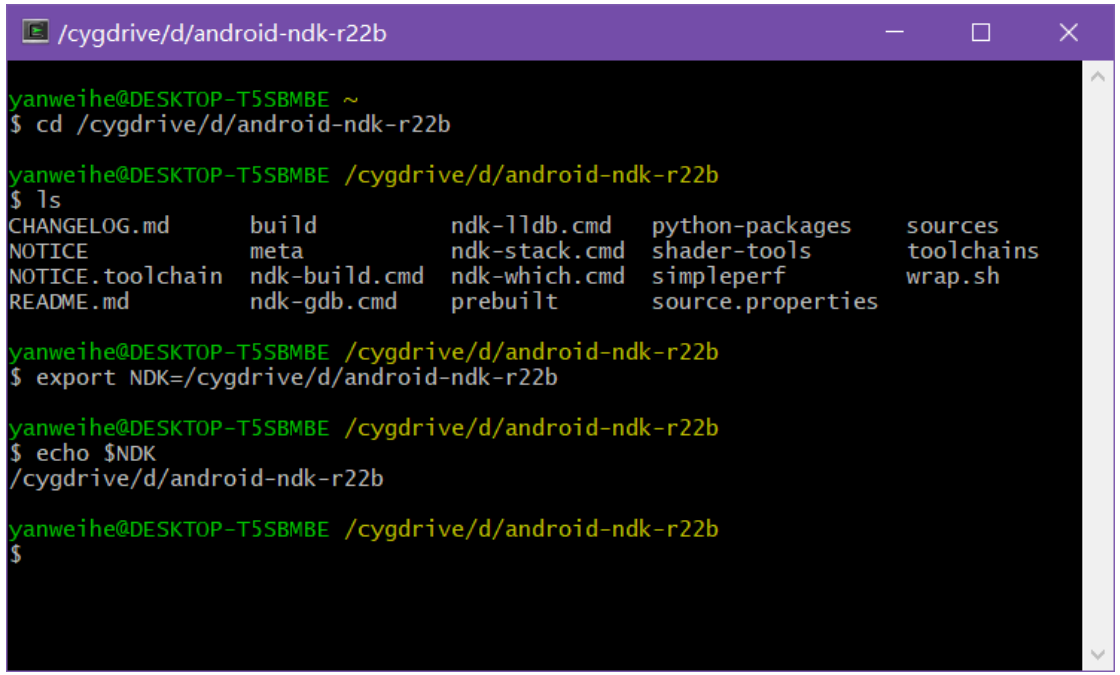
  - export是Linux下的变量设置命令，设置一个名为NDK的变量（变量名称可以更换），用来保存Android NDK的主目录位置

11.4 NDK示例

- 编译共享库模块
  - 笔者的NDK保存在D:\android-ndk-r22b，因此在Cygwin中的目录则是/cygdrive/d/android-ndk-r22b
  - 设置NDK变量的目的是简化后面编译过程中的命令输入操作

11.4 NDK示例

- 编译共享库模块
  - 程序开发人员可以使用Linux的echo命令查看NDK变量的值，如下图所示：

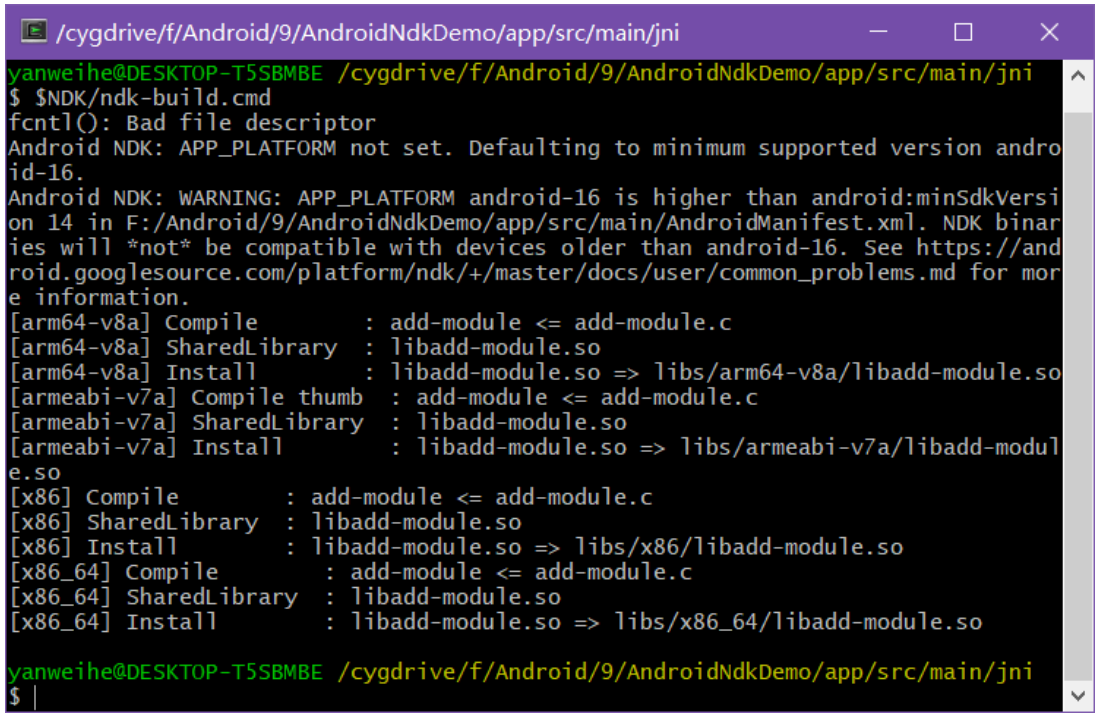


11.4 NDK示例

- 编译共享库模块
  - 使用cd命令和cd..命令切换到Android的工程目录下，使用Android NDK目录中提供的脚本文件ndk-build编译C代码模块
  - ndk-build脚本是Android NDK为简化编译过程而在v4版本推出的，该脚本会自动探索Android工程目录中的文件，以确定哪些文件需要编译，以及如何进行编译
  - 程序开发人员只需要在Android的工程目录下输入如下命令：
  - $NDK/ndk-build .cmd

11.4 NDK示例

- 编译共享库模块
  - 编译成功的提示如下图所示
  - 提示信息说明将add-module.c源文件编译成add-module模块，产生的libadd-module.so文件保存在<project>/app/src/main/libs目录中



11.4 NDK示例

- 编译共享库模块
  - 为了保证程序能正常运行还需要把<project>/app/src/main/libs目录中的文件复制到<project>/app/libs目录之下
  - 然后在app目录下的build.gradle文件中配置android块的sourceSets标签
  - 最后点击右上角的“Sync Now”按钮进行重新构建。

11.4 NDK示例

- 编译共享库模块
  - build.gradle文件的核心配置代码如下

```
1. ……
2. android {
3. ……
4.     sourceSets {
5.         main {
6.             jniLibs.srcDir 'libs'
7.         }
8.     }
9. }
10. ……
```

11.4 NDK示例

- 运行Android程序
  - 在运行AndroidNdkDemo示例程序前，务必将AndroidNdkDemoActivity.java文件中第16行和第23行的注释取消，并注释掉第25行到第27行代码
  - 代码修改后，AndroidNdkDemo示例将调用libadd-module.so文件中的add()函数，完成加法运算，并将结果显示在用户界面上

习题：

- 1.简述Android NDK开发的优势和不足。
- 2.说明Android NDK应用程序开发的一般步骤。

THANKS

谢　　谢　　观　　看