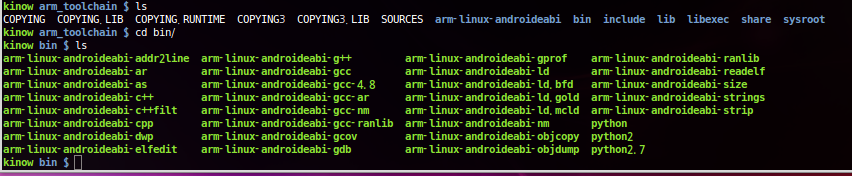
1. Android NDK – ARM build toolchain environment

$ vi toolchain.sh

|  |
| --- |
| #!/bin/bash  # 여기 위치에 toolchain 환경 및 컴파일러가 준비된다.  TARGET\_DIR=$1  ARCH=arm  #ARCH=arm64  SYSTEM=linux-x86  #SYSTEM=linux-x86\_64  # android NDK가 제공하는 편리한 toolchain 컴파일 환경 제공  MAKE\_SCRIPT=$ANDROID\_NDK/build/tools/make-standalone-toolchain.sh  $MAKE\_SCRIPT --arch=$ARCH --ndk-dir=$ANDROID\_NDK --system=$SYSTEM --install-dir=$TARGET\_DIR –platform=android-21 |



2. libpcap build using ARM toolchain

$ vi configure\_env\_pcap.sh

|  |
| --- |
| #!/bin/bash  # 위에서 설치한 ARM toolchain 위치  TOOLCHAIN=$1  CROSS=$TOOLCHAIN/bin/arm-linux-androideabi-  INCDIR=$TOOLCHAIN/sysroot/usr/include  LIBDIR=$TOOLCHAIN/sysroot/usr/lib  SYSROOT=$TOOLCHAIN/sysroot  export CC=$CROSS"gcc"  export AR=$CROSS"ar" $ARFLAGS r  export RANLIB=$CROSS"ranlib"  export NM=$CROSS"nm" |

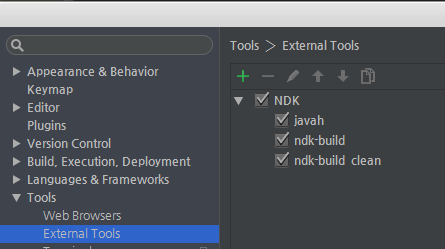
$ ./configure --host=arm-linux –with-pcap=linux



2. NDK sample code using Android Studio

2-1. setting NDK command

File > Settings...



[javah]

Program: /home/kinow/tools/jdk1.7.0\_79/bin/javah

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

Working directory: $SourcepathEntry$/../jni

[ndk-build]

Program: /home/kinow/tools/Android/Sdk/ndk-bundle/ndk-build

Working directory: $SourcepathEntry$

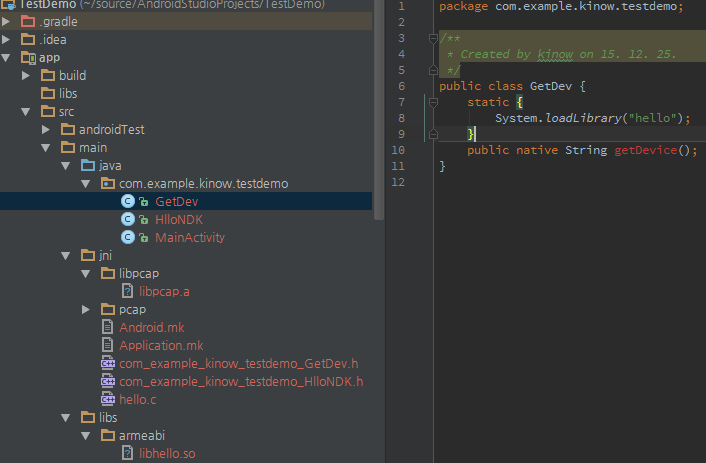
[ndk-build clean]

Program: /home/kinow/tools/Android/Sdk/ndk-bundle/ndk-build

Parameters: clean

Working directory: $SourcepathEntry$

2-2. NDK build



GetDev.java에 Pcap을 이용한 native 함수 호출 구현

[디렉토리 생성]

jni

jni/libpcap ← 여기에 arm으로 compile한 라이브러리를 갖다 놓는다(여기서는 static library로 함, so파일 연동이 잘 안되서... 나중에 다시 시도해봐야겠다 - 아마 될꺼다 ㅎㅎ)

|  |
| --- |
| com\_example\_kinow\_testdemo\_GetDev.h  #include <jni.h>  /\* Header for class com\_example\_kinow\_testdemo\_GetDev \*/  #ifndef \_Included\_com\_example\_kinow\_testdemo\_GetDev  #define \_Included\_com\_example\_kinow\_testdemo\_GetDev  #ifdef \_\_cplusplus  **extern** "C" {  #endif  /\*  \* Class: com\_example\_kinow\_testdemo\_GetDev  \* Method: getDevice  \* Signature: ()Ljava/lang/String;  \*/  JNIEXPORT jstring JNICALL Java\_com\_example\_kinow\_testdemo\_GetDev\_getDevice  (JNIEnv \*, jobject);  #ifdef \_\_cplusplus  }  #endif  #endif |

|  |
| --- |
| hello.c  #include <stdio.h>  #include <pcap/pcap.h>  #include <netinet/in.h>  #include <arpa/inet.h>  #include "com\_example\_kinow\_testdemo\_HlloNDK.h"  #include "com\_example\_kinow\_testdemo\_GetDev.h"  JNIEXPORT jstring JNICALL Java\_com\_example\_kinow\_testdemo\_HlloNDK\_getHello  (JNIEnv \*env, jobject obj)  {  jstring str = (\*env)->NewStringUTF(env, "Hello World!!!");  **return** str;  }  JNIEXPORT jstring JNICALL Java\_com\_example\_kinow\_testdemo\_GetDev\_getDevice  (JNIEnv \*env, jobject obj)  {  jstring str = NULL;  **char** \*dev; // 사용중인 네트웍 디바이스 이름  **char** \*net; // 네트웍 어드레스  **char** \*mask; // 네트웍 mask 어드레스  **int** ret; //  **char** errbuf[PCAP\_ERRBUF\_SIZE] = {0, };  bpf\_u\_int32 netp; // ip  bpf\_u\_int32 maskp; // submet mask  **struct** in\_addr addr;  **char** result[1024] = {0, };  **int** len = 0;  // 네트웍 디바이스 이름을 얻어온다.  dev = pcap\_lookupdev(errbuf);  **if** (dev == NULL) {  str = (\*env)->NewStringUTF(env, errbuf);  **return** str;  }  // 네트웍 디바이스 이름 출력  len += snprintf(result, **sizeof**(result)-len, "DEV:%s", dev);  // 네트웍 디바이스 이름 dev 에 대한  // mask, ip 정보 얻어오기  ret = pcap\_lookupnet(dev, &netp, &maskp, errbuf);  **if** (ret == -1) {  str = (\*env)->NewStringUTF(env, errbuf);  len += snprintf(result, **sizeof**(result)-len, "pcap\_lookupnet:%s", str);  **return** result;  }  // 네트웍 어드레스를 점박이 3형제 스타일로  addr.s\_addr = netp;  net = inet\_ntoa(addr);  **if** (net == NULL) {  str = (\*env)->NewStringUTF(env, "failed to inet\_ntoa");  len += snprintf(result, **sizeof**(result)-len, "inet\_ntoa:%s", str);  **return** str;  }  len += snprintf(result, **sizeof**(result)-len, ",NET:%s", net);  // 마찬가지로 mask 어드레스를 점박이 3형제 스타일로  addr.s\_addr = maskp;  mask = inet\_ntoa(addr);  **if** (mask == NULL) {  str = (\*env)->NewStringUTF(env, "failed to inet\_ntoa");  len += snprintf(result, **sizeof**(result)-len, "inet\_ntoa:%s", str);  **return** str;  }  len += snprintf(result, **sizeof**(result)-len, ",MASK:%s", mask);  str = (\*env)->NewStringUTF(env, result);  **return** str;  } |

|  |
| --- |
| Android.mk  LOCAL\_PATH := $(call my-dir)  include $(CLEAR\_VARS)  LOCAL\_MODULE := pcap  LOCAL\_SRC\_FILES := libpcap/libpcap.a  include $(PREBUILT\_STATIC\_LIBRARY)  include $(CLEAR\_VARS)  LOCAL\_MODULE := hello  LOCAL\_SRC\_FILES := hello.c  LOCAL\_STATIC\_LIBRARIES := pcap  include $(BUILD\_SHARED\_LIBRARY) |

|  |
| --- |
| Application.mk  #APP\_ABI := all #arm64-v8a armeabi armeabi-v7a mips x86 x86\_64  APP\_ABI := armeabi  APP\_PLATFORM := android-21  NDK\_TOOLCHAIN\_VERSION := 4.8 |

2-3. include pcap headers

사실 header include가 잘 안되서 그냥 toolchain환경에다가 pcap 관련 header파일을 복사함

Android.mk를 잘 만들어서 이 과정을 없애야 겠다.

2-4. NDK build

Setting... 해놓은 ndk-build를 수행하면

libs/armeabi/libhello.so 파일이 생성됨 ← 이파일은 기본이 strip 됨

2-5. app/build.gradle 설정

|  |
| --- |
| android {  compileSdkVersion 23  buildToolsVersion "23.0.2"  defaultConfig {  applicationId "com.example.kinow.testdemo"  minSdkVersion 21  targetSdkVersion 23  versionCode 1  versionName "1.0"  ndk {  moduleName "hello"  abiFilters "armeabi"  ldLibs "libhello.so"  }  sourceSets.main {  jniLibs.srcDir 'src/main/libs'  jni.srcDirs = [] //disable automatic ndk-build call  // 요게 필수 이다. 아까 ndk-build를 돌렸으므로 여기서는 필요없다.  // android studio 빌드시에도 ndk-build를 돌릴 수 있는데 header include path가 잘 안잡힌다.  // 엄청난 삽질을 했었다. 그냥 편하게 여기서 ndk-build를 수행하지 않도록 하자  }  }  buildTypes {  release {  minifyEnabled = false  proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'  proguardFiles += file('proguard-rules.txt')  }  debug {  jniDebuggable true  debuggable true  }  }  productFlavors {  }  } |

2-6. app/gradle.properties 설정

android studio에서 ndk 빌드가 deprecated되어서 아래처럼 해주어야한다.

왜 deprecated 되었을까?? 궁금...

|  |
| --- |
| android.useDeprecatedNdk=true |

3. 실행(Run)

3-1. socket error

스마트폰에서 테스트 하였으나 이런 제장

socket permission 에러 발생

AndroidManifest.xml

인터넷 퍼미션 추가

|  |
| --- |
| <uses-permission android:name="android.permission.INTERNET"/> |

3-2. socket error

이런 젠장 또 에러 발생

socket netlink: permission 문제 …

아~ ㅠㅠ netlink 소켓은 super user 권한이 필요하다.

여기서 까지다... 젠장 더 연구 해야 겠다.

android에 pcap 올리는 그 날까지... 더불어 최종 목표는 netfilter를 건드는 것이다!!!!!!

참고로 해당 소스는 github.com/kangaukju에 TestDemo로 올린다.