Origenboard.org

Linaro

Android Build System Rev-0.1

고도리 of AESOP

2012/04/12

InSignal

INDEX

- Android build environment setup
- envsetup.sh
- Android source tree
- Android.mk

- **Android build environment**

Initializing the Build Environment

- **Android source download**
  - ✓ http://source.android.com/source/initializing.html
    - ➢ Build releated documents & source download site
- **Setting up a Linux build environment**
  - ✓ Android를 컴파일 하기 위해서는 여러 가지 tool이 필요함
  - ✓ 일반적으로 필요한 환경
    - ➢ Ubuntu 10.04+ or MAC OS
    - ➢ Python 2.5 ~ 2.7
    - ➢ GNU Make 3.81
      - 위의 사이트에는 3.82도 상관없다라고 되어 있지만, ICS 버전 컴파일 시 에러 남
    - ➢ JDK 6
      - Gingerbread or newer version
      - JDK 5 for Froyo or older version
    - ➢ Git 1.7 or newer
  - ✓ Ubuntu를 사용할 경우 version에 따라 환경 문제가 많이 발생
    - ➢ Ubuntu 11.10의 경우 gcc compiler version issue 발생

Downloading the Source Tree

- **Download site**
  - ✓ http://source.android.com/source/downloading.html
  - ✓ ※Android source 는 6Gbytes 이상임.
- **Install repo**
  - ✓ Repo is a tool that makes it easier to work with Git in the context of Android
  - ✓ Repo is installed in the following order
    - ➢ cd /usr/local/bin
    - ➢ curl https://dl-ssl.google.com/dl/googlesource/git-repo/repo > /usr/local/bin
    - ➢ chmod 755 /usr/local/bin/repo
- **Initializing a Android working directory**
  - ✓ Android source is downloaded in the following order
    - ➢ mkdir YOUR_ANDROID_DIRECTORY
    - ➢ cd YOUR_ANDROID_DIRECTORY
    - ➢ repo init -u https://android.googlesource.com/platform/manifest -b android-4.0.4_r1.1
      - In case of android ICS 4.0.4 version at 04/12/2012
    - ➢ repo sync

Building the System

- **Build environment setup**
  - ✓ **In the Android source's top directory**
  - ✓ **Use the envsetup.sh**
    - ➢ ***source build/envsetup.sh***
    - ➢ **or**
    - ➢ ***. build/envsetup.sh***
  - ✓ **Choose a Target**
    - ➢ **You can use 'lunch' command**
      - ***lunch***
      - **You can show prepared lunch menu with target devices**
    - ➢ **Android emulator를 위한 lunch 명령어**
      - ***lunch full-eng***

Building the System

- **lunch 명령**
  - ✓ 컴파일 하려고 하는 타겟 디바이스를 설정하는 명령
  - ✓ build target을 지정하는 형식은 BUILD-BUILDTYPE 형태
    - ➢ BUILD는 다음과 같은 조합을 말함

Build name	Device	Notes
full	emulator	fully configured with all languages, apps, input methods
full_maguro	maguro	full build running on Galaxy Nexus GSM/HSPA+ ("maguro")
full_panda	panda	full build running on PandaBoard ("panda")

    - ➢ BUILDTYPE은 다음 중 하나이다

Buildtype	Use
user	limited access; suited for production
userdebug	like "user" but with root access and debuggability; preferred for debugging
eng	development configuration with additional debugging tools

Building the System

- lunch menu에 새로운 target device 등록하기
  - ✓ build/envsetup.sh 실행 시
    - ➢ 다음과 같은 파일을 검색해서 lunch menu에 등록
      - vendor/*/vendorsetup.sh
      - vendor/*/*/vendorsetup.sh
      - device/*/*/vendorsetup.sh
    - ➢ vendorsetup.sh의 내용은 다음과 같이 작성(eg.> Nexus-S의 경우)
      - device/samsung/crespo/vendorsetup.sh
        - add_lunch_combo full_crespo-userdebug
- choosecombo
  - ✓ lunch 명령과 같은 역할. 차이점은 lunch의 경우 envsetup.sh 시 lunch menu에 해당 target이 등록되어 있어야 하지만
  - ✓ choosecombo의 경우는 build target 지정을 직접 command line에서 할 수 있도록 되어 있음

Building the System

- Target 설정이 끝나면 build를 시작
  - ✓ *make* 혹은 *make –j4*와 같은 명령 이용, 여기서 4는 JOB 개수
  - ✓ CPU core가 6개일 경우는 –j6 옵션을 주면 됨
  - ✓ *'showcommands'*옵션
    - ➢ *make showcommands*
    - ➢ 위와 같은 옵션을 주고 make를 하게 되면 어떻게 compile이 되고 link되는지 전체 진행 과정을 보여준다.

envsetup.sh

- **build/envsetup.sh**
  - ✓ Android build system에서 제공하는 shell script 명령어들
  - ✓ 부분 컴파일을 위한 toolchain 경로 지정 등의 역할
  - ✓ 다음과 같이 실행

```
In Android source top directory

$ANDROID_HOME #> . build/envsetup.sh
혹은
$ANDROID_HOME #> source build/envsetup.sh
```

- **help**
  - ✓ envsetup.sh의 실행이 끝난 후 host의 shell상에서 help를 실행

```
root@darkstar:~/android/ics/20120406# help
Invoke ". build/envsetup.sh" from your shell to add the following functions to your environment:
- croot:   Changes directory to the top of the tree.
- m:       Makes from the top of the tree.
- mm:      Builds all of the modules in the current directory.
- mmm:     Builds all of the modules in the supplied directories.
- cgrep:   Greps on all local C/C++ files.
- jgrep:   Greps on all local Java files.
- resgrep: Greps on all local res/*.xml files.
- godir:   Go to the directory containing a file.

Look at the source to view more functions. The complete list is:
_lunch add_lunch_combo addcompletions cgrep check_product check_variant choosecombo chooseproduct choosetype choosevariant cproj cro
ot findmakefile gdbclient get_abs_build_var get_build_var getbugreports getprebuilt gettop godir help isviewserverstarted jgrep key_
back key_home key_menu lunch m mm mmm pid print_lunch_menu printconfig resgrep runhat runtest set_java_home set_sequence_number set_
stuff_for_environment setpaths settitle smoketest startviewserver stopviewserver systemstack tapas tracedmdump
root@darkstar:~/android/ics/20120406#
```

envsetup.sh

- **croot**
  - ✓ Changes directory to the top of the tree
  - ✓ Android source directory의 하위 디렉토리에서만 실행

```
root@darkstar:~/android/ics/20120406# cd external/tinyalsa
root@darkstar:~/android/ics/20120406/external/tinyalsa# pwd
/root/android/ics/20120406/external/tinyalsa
root@darkstar:~/android/ics/20120406/external/tinyalsa# croot
root@darkstar:~/android/ics/20120406# pwd
/root/android/ics/20120406
root@darkstar:~/android/ics/20120406#
root@darkstar:~/android/ics/20120406# cd ../
root@darkstar:~/android/ics# croot
Couldn't locate the top of the tree.  Try setting TOP.
root@darkstar:~/android/ics#
```

envsetup.sh

- m, mm, mmm
  - ✓ Android source build help commands
- m
  - ✓ Android sub tree에서 source top directory로 이동 후 Android 전체 make
  - ✓ make 완료 후 명령어를 실행한 sub directory로 다시 돌아옴

```
root@darkstar:~/android/ics/20120406# cd external/tinyalsa/
root@darkstar:~/android/ics/20120406/external/tinyalsa# m
============================================
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=4.0.4
TARGET_PRODUCT=full
TARGET_BUILD_VARIANT=eng
TARGET_BUILD_TYPE=release
TARGET_BUILD_APPS=
TARGET_ARCH=arm
TARGET_ARCH_VARIANT=armv7-a
HOST_ARCH=x86
HOST_OS=linux
HOST_BUILD_TYPE=release
BUILD_ID=IMM76D
============================================
make: Entering directory `/root/android/ics/20120406'
target SharedLib: audio.primary.herring (out/target/product/generic/obj/SHARED_LIBRARIES/audio.primary.herring_
intermediates/LINKED/audio.primary.herring.so)
target Symbolic: audio.primary.herring (out/target/product/generic/symbols/system/lib/hw/audio.primary.herring.
so)
target Strip: audio.primary.herring (out/target/product/generic/obj/lib/audio.primary.herring.so)
target SharedLib: audio.primary.tuna (out/target/product/generic/obj/SHARED_LIBRARIES/audio.primary.tuna_interm
ediates/LINKED/audio.primary.tuna.so)
target Symbolic: audio.primary.tuna (out/target/product/generic/symbols/system/lib/hw/audio.primary.tuna.so)
target Strip: audio.primary.tuna (out/target/product/generic/obj/lib/audio.primary.tuna.so)
make: Leaving directory `/root/android/ics/20120406'
root@darkstar:~/android/ics/20120406/external/tinyalsa#
```

envsetup.sh

- **mm**
  - ✓ 현재 directory 밑의 모든 Android.mk 파일을 찾아서 컴파일을 해 준다
  - ✓ mm 명령은 오직 컴파일과 root filesystem에 관련 file들을 update만 함
  - ✓ 컴파일 후 system.img를 자동으로 생성하기 위해서는 snod 명령 같이 사용

```
root@darkstar:~/android/ics/20120406/external/tinyalsa# touch pcm.c
root@darkstar:~/android/ics/20120406/external/tinyalsa# mm
============================================
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=4.0.4
TARGET_PRODUCT=full
TARGET_BUILD_VARIANT=eng
TARGET_BUILD_TYPE=release
TARGET_BUILD_APPS=
TARGET_ARCH=arm
TARGET_ARCH_VARIANT=armv7-a
HOST_ARCH=x86
HOST_OS=linux
HOST_BUILD_TYPE=release
BUILD_ID=IMM76D
============================================
make: Entering directory `/root/android/ics/20120406'
target thumb C: libtinyalsa <= external/tinyalsa/pcm.c
target SharedLib: libtinyalsa (out/target/product/generic/obj/SHARED_LIBRARIES/libtinyalsa_intermediates/LINKED
/libtinyalsa.so)
target Symbolic: libtinyalsa (out/target/product/generic/symbols/system/lib/libtinyalsa.so)
target Strip: libtinyalsa (out/target/product/generic/obj/lib/libtinyalsa.so)
Install: out/target/product/generic/system/lib/libtinyalsa.so
target Executable: tinyplay (out/target/product/generic/obj/EXECUTABLES/tinyplay_intermediates/LINKED/tinyplay)
target Symbolic: tinyplay (out/target/product/generic/symbols/system/bin/tinyplay)
target Strip: tinyplay (out/target/product/generic/obj/EXECUTABLES/tinyplay_intermediates/tinyplay)
Install: out/target/product/generic/system/bin/tinyplay
target Executable: tinycap (out/target/product/generic/obj/EXECUTABLES/tinycap_intermediates/LINKED/tinycap)
target Symbolic: tinycap (out/target/product/generic/symbols/system/bin/tinycap)
target Strip: tinycap (out/target/product/generic/obj/EXECUTABLES/tinycap_intermediates/tinycap)
Install: out/target/product/generic/system/bin/tinycap
target Executable: tinymix (out/target/product/generic/obj/EXECUTABLES/tinymix_intermediates/LINKED/tinymix)
target Symbolic: tinymix (out/target/product/generic/symbols/system/bin/tinymix)
target Strip: tinymix (out/target/product/generic/obj/EXECUTABLES/tinymix_intermediates/tinymix)
Install: out/target/product/generic/system/bin/tinymix
make: Leaving directory `/root/android/ics/20120406'
root@darkstar:~/android/ics/20120406/external/tinyalsa#
```

envsetup.sh

- **mm command with snod option**
  - ✓ **Build all sources in the current directory and sub-directories.**
  - ✓ ***And builds new system image from current binaries***

```
root@darkstar:~/android/ics/20120406/external/tinyalsa# touch pcm.c
root@darkstar:~/android/ics/20120406/external/tinyalsa# mm snod
============================================
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=4.0.4
TARGET_PRODUCT=full
TARGET_BUILD_VARIANT=eng
TARGET_BUILD_TYPE=release
TARGET_BUILD_APPS=
TARGET_ARCH=arm
TARGET_ARCH_VARIANT=armv7-a
HOST_ARCH=x86
HOST_OS=linux
HOST_BUILD_TYPE=release
BUILD_ID=IMM76D
============================================
make: Entering directory `/root/android/ics/20120406'
target thumb C: libtinyalsa <= external/tinyalsa/pcm.c
target SharedLib: libtinyalsa (out/target/product/generic/obj/SHARED_LIBRARIES/libtinyalsa_intermediates/LINKED
/libtinyalsa.so)
target Symbolic: libtinyalsa (out/target/product/generic/symbols/system/lib/libtinyalsa.so)
target Strip: libtinyalsa (out/target/product/generic/obj/lib/libtinyalsa.so)
Install: out/target/product/generic/system/lib/libtinyalsa.so
target Executable: tinyplay (out/target/product/generic/obj/EXECUTABLES/tinyplay_intermediates/LINKED/tinyplay)
target Symbolic: tinyplay (out/target/product/generic/symbols/system/bin/tinyplay)
target Strip: tinyplay (out/target/product/generic/obj/EXECUTABLES/tinyplay_intermediates/tinyplay)
Install: out/target/product/generic/system/bin/tinyplay
target Executable: tinycap (out/target/product/generic/obj/EXECUTABLES/tinycap_intermediates/LINKED/tinycap)
target Symbolic: tinycap (out/target/product/generic/symbols/system/bin/tinycap)
target Strip: tinycap (out/target/product/generic/obj/EXECUTABLES/tinycap_intermediates/tinycap)
Install: out/target/product/generic/system/bin/tinycap
target Executable: tinymix (out/target/product/generic/obj/EXECUTABLES/tinymix_intermediates/LINKED/tinymix)
target Symbolic: tinymix (out/target/product/generic/symbols/system/bin/tinymix)
target Strip: tinymix (out/target/product/generic/obj/EXECUTABLES/tinymix_intermediates/tinymix)
Install: out/target/product/generic/system/bin/tinymix
make snod: ignoring dependencies
Target system fs image: out/target/product/generic/system.img
make: Leaving directory `/root/android/ics/20120406'
root@darkstar:~/android/ics/20120406/external/tinyalsa#
```

envsetup.sh

- **mmm**
  - ✓ **Build specified directory or list of directories.**
  - ✓ **You can use with 'snod' option like 'mm' command.**

```
root@darkstar:~/android/ics/20120406/external# touch ../frameworks/base/media/libstagefright/AwesomePlayer.cpp
 tinyalsa/pcm.c
root@darkstar:~/android/ics/20120406/external#
root@darkstar:~/android/ics/20120406/external# mmm ../frameworks/base/media/libstagefright/ tinyalsa
============================================
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=4.0.4
TARGET_PRODUCT=full
TARGET_BUILD_VARIANT=eng
TARGET_BUILD_TYPE=release
TARGET_BUILD_APPS=
TARGET_ARCH=arm
TARGET_ARCH_VARIANT=armv7-a
HOST_ARCH=x86
HOST_OS=linux
HOST_BUILD_TYPE=release
BUILD_ID=IMM76D
============================================
make: Entering directory `/root/android/ics/20120406'
target thumb C++: libstagefright <= external/../frameworks/base/media/libstagefright/AwesomePlayer.cpp
```

envsetup.sh

- **cgrep, jgrep, resgrep**
  - ✓ 각각 C/C++ source, JAVA source, resource file서부터 특정 string을 찾아준다.

- **godir**
  - ✓ 지정된 디렉토리로 이동할 수 있음

```
root@darkstar:~/android/ics/20120406/external# godir tinyalsa
Creating index... Done

   [1] ./external/tinyalsa
   [2] ./external/tinyalsa/.git
   [3] ./external/tinyalsa/include/tinyalsa

Select one: 1
root@darkstar:~/android/ics/20120406/external/tinyalsa#
```

Android source tree

- bionic – bionic libc
- build
  - ✓ envsetup.sh
    - ➢ build shell script들이 있음, 툴체인 경로 설정등 환경설정
  - ✓ generic board에 대한 configuration
    - ➢ build/target/board/generic/device.mk
      - root filesystem의 구성을 어떻게 해야하는가에 대한 방향을 지정하는 파일
      - android 최종 결과물 구성시 자동으로 포함하고 싶은 binary들에 대한 install을 결정
    - ➢ build/target/board/generic/BoardConfig.mk
      - Android의 makefile인 Android.mk에 기본적으로 포함되는 최상위 Makefile
      - 주로 HAL 혹은 기능들에 대한 enable/disable과 관련이 깊음
      - Android.mk 파일에 영향을 주고 Android.mk 파일에서 define을 제어할 수 있도록 설정하는 경우가 많음
      - ex> BoardConfig.mk의 *BOARD_USES_GENERIC_AUDIO := true* 의 경우
        - frameworks/base/services/audioflinger/Android.mk
        - frameworks/base/services/audioflinger/AudioHardwareInterface.cpp
        - 파일의 해당 부분을 참조
- CTS – Compatibility Test Suite관련 소스 디렉토리

Android source tree

- development – 개발용 app등…
- external
  - ✓ Android 고유의 library가 아닌 linux 혹은 기존에 작성된 library/binary 소스
  - ✓ 작성된 library 중 shared library(*.so) 파일은 root filesystem의 system/lib 로 install
  - ✓ 작성된 library 중 binary 파일은 root filesystem의 system/bin 으로 install
  - ✓ *framework test app와 library등은 이 디렉토리에서 작업하는게 좋음*
- frameworks
  - ✓ base/policy
    - ➢ PhoneWindowManager 소스
    - ➢ Android 전체 시스템을 background에서 제어하는 최상위 application인 PhoneWindowManager는 다음과 같은 기능을 갖는다
      - KeyGuard관련 – LockScreen
      - 화면 전체 동작 제어(ex> Rotation)
      - event 관리 – Key event등

Android source tree

- **frameworks**
  - ✓ **base – android framework source**
    - ➢ **libs/ui**
      - **Android framework에서 JNI를 통해서 호출되는 android client part**
      - **HAL – Input device(key, touch)의 경우**
    - ➢ **libs/utils**
      - **Wrapping class, 압축관련 유틸리티 등…**
    - ➢ **libs/binder**
      - **Android binder & Anonymous shared memory 제어 클래스**
    - ➢ **cmds**
      - **binder관련 binary인 servicemanager 소스와 여러가지 command들**
    - ➢ **media**
      - **media관련 JAVA, JNI, Client, Service, Media engine(libstagefright)등의 소스**
      - ***media관련은 너무 크고, 독립적이기 때문에 따로 디렉토리를 만들어 관련 소스를 하나의 디렉토리에 넣음***

Android source tree

- device
  - ✓ 각 vendor에서 만드는 vendor별 device들에 대한 설정파일
  - ✓ 자체 device와 product를 만들기 위해서는 이 디렉토리에서 관련 파일들을 작성하는 것이 좋음
  - ✓ 작성되는 주요 파일 리스트
    - ➢ AndroidProducts.mk
      - Product, device 관련 이름들을 지정
    - ➢ BoardConfig.mk, device.mk
      - generic board에서와 마찬가지 역할을 한다.
- vendor
  - ✓ device 디렉토리와 같은 역할
  - ✓ Android 2.1/2.2 버전에서는 이 디렉토리를 기본 device들에 대한 디렉토리로 사용
  - ✓ 현재(Ice Cream Sandwich)도 이 디렉토리를 사용하는 경우 있음

Android source tree

- out
  - ✓ android compile 결과물 디렉토리
- packages
  - ✓ android 기본 application source
  - ✓ 주의해야할 점은 모든 app가 컴파일 되지 않는다.
  - ✓ 컴파일 되는 패키지들은 각 device별 device.mk에 지정이 되는 PRODUCT_PACKAGES 변수에 포함되어야 함
  - ✓ ex> build/target/product/generic-no-telephony.mk
- prebuilt
  - ✓ toolchain & 필요한 binary
- system
  - ✓ android의 root filesystem에 포함되는 기본 binary 소스(ex> init)
  - ✓ /core/init – android init source
  - ✓ /vold – vold2, android 2.3 버전서부터 사용됨

Android source tree

- **hardware**
  - ✓ HAL source & header file – HAL Class의 prototype
  - ✓ 일반적으로 android에서 사용되는 hardware 관련 소스들을 포함, 반드시 이 디렉토리에만 위치하는 것은 아님
    - ➢ device, vendor 디렉토리에 존재하는 경우가 많음
  - ✓ libhardware
    - ➢ board hardware dependency가 높은 하드웨어 모듈들의 example 및 header
    - ➢ 일반적으로 안드로이드 폰에 기본적으로 탑재되지 않고, cpu혹은 제조사의 모듈에 의존성이 높은 부분들에 관련된 class prototype 코드가 있음
    - ➢ 안드로이드 소스 전체를 컴파일 하지 않는 경우도 있고, 모듈단위로 직접 컴파일해서 안드로이드 root filesystem에 install하는 경우도 있음
    - ➢ Android framework에서 동적으로 module(*.so 파일 형태)을 loading하는 *hw_get_module()* 함수에 대한 소스가 있음
      - hw_get_module()함수는 다음과 같은 순서로 동적 module을 loading한다.
        - $(MODULE_NAME).%*ro.hardware*%.so ➔ ex> sensors.origen.so
        - $(MODULE_NAME).%*ro.product.board*%.so ➔ ex> sensors.origenboard.so
        - $(MODULE_NAME).%*ro.board.platform*%.so ➔ ex> sensors.insignal_origen.so
        - $(MODULE_NAME).%*ro.arch*%.so ➔ ex> sensors.exynos4.so
      - hw_get_module()함수는 rootfs의 system/lib/hw, vendor/lib/hw 디렉토리를 검색

Android source tree

- hardware
  - ✓ libhardware_legacy
    - ➢ 일반적으로 android phone에 존재해야 하는 hardware에 대한 제어 코드들이 들어 있음
    - ➢ Android compile시 같이 컴파일 된다
    - ➢ ex> wifi/wifi.c
    - ➢ include/*
      - Android built-in HAL의 일부 prototype이 선언되어 있음
      - ex> Audio
- ndk
  - ✓ Native Development Kit 관련 파일들이 있음
  - ✓ docs/ANDROID-MK.html
    - ➢ Android에서 사용되는 makefile인 Android.mk 파일을 어떻게 작성해야 하는지에 대해서 설명되어 있음
    - ➢ PDK(Platform Development Kit – Android 소스)에 적용되는 부분에 대한 설명보다는 NDK용 Android.mk 파일에 대한 설명임
    - ➢ 하지만, PDK용 문서로 참고하기 좋음

Android.mk

- **Android.mk**
  - ✓ Android 시스템에서 사용되는 makefile
  - ✓ Build system이 어떻게 소스들을 다뤄야 하는지에 대한 설명
  - ✓ 최종 결과물은 다음 중 하나
    - ➢ Executable binary
    - ➢ Shared library
    - ➢ Static library
    - ➢ Host executable binary
    - ➢ Application package
    - ➢ etc….
      - build/core/config.mk 에 많은 부분이 더 define되어 있으나 사용빈도가 낮다
  - ✓ Android source top directory에서 make 명령 사용시
    - ➢ Sub directory의 모든 Android.mk 파일을 검색
    - ➢ Source file들의 update 상황 검사
      - 결과물이 static library가 아닐 경우 컴파일 후 library/binary등을 root filesystem에 update

Android.mk

- **Executable binary example**

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

LOCAL_MODULE_TAGS := eng

LOCAL_SRC_FILES:= server.cpp

LOCAL_SHARED_LIBRARIES := libcutils libutils libbinder libgadd01

base := $(LOCAL_PATH)/../../..

LOCAL_C_INCLUDES := \
      $(base)/binder01

LOCAL_MODULE:= server01

include $(BUILD_EXECUTABLE)
```

Android.mk

- **Android.mk를 구성하는 문법 & definition**
  - ✓ ***LOCAL_PATH := $(call my-dir)***
    - ➢ Android.mk 파일은 반드시 이 정의로 시작되어야 한다.
    - ➢ Android 소스트리에서 소스 파일들의 위치를 지정하는데 사용
    - ➢ 'my-dir' macro를 이용해서 현재 디렉토리로 지정
  - ✓ ***include $(CLEAR_VARS)***
    - ➢ Android.mk 파일 내에서 사용되는 LOCAL_XXX 형태의 변수들을 clear함
    - ➢ e.g. LOCAL_MODULE, LOCAL_SRC_FILES, LOCAL_STATIC_LIBRARIES, etc...
      - LOCAL_PATH는 제외됨
    - ➢ build/core/clear_vars.mk
      - 어떤 definition들이 clear되는지 기록되어 있음
      - build/core/config.mk 파일에서 include됨

Android.mk

- **Android.mk를 구성하는 문법 & definition**
  - ✓ ***LOCAL_MODULE***
    - ➢ 최종 출력 결과물의 이름, 반드시 unique한 이름이어야 함
    - ➢ 최종 결과물의 종류 지정에 따라 생성 파일이름이 달라짐
  - ✓ **Build output type definition**
    - ➢ ***include $(BUILD_EXECUTABLE)***
      - 최종 결과물은 LOCAL_MODULE에서 지정된 이름 그대로 생성됨
      - out/target/product/$(TARGET_PRODUCT)/obj/EXECUTABLES/$(LOCAL_MODULE)_intermediates/LINKED/$(LOCAL_MODULE)
      - Root filesystem의 system/bin 디렉토리로 복사 install됨
        - out/target/product/$(TARGET_PRODUCT)/system/bin/$(LOCAL_MODULE)
    - ➢ ***include $(BUILD_SHARED_LIBRARY)***
      - 최종 결과물은 LOCAL_MODULE.so 의 형태를 갖는 shared object를 생성
      - out/target/product/$(TARGET_PRODUCT)/obj/SHARED_LIBRARIES/$(LOCAL_MODULE)_intermediates/LINKED/$(LOCAL_MODULE).so
      - Root filesystem의 system/lib 디렉토리로 복사 install됨
        - out/target/product/$(TARGET_PRODUCT)/system/lib/$(LOCAL_MODULE).so

Android.mk

- **Android.mk를 구성하는 문법 & definition**
  - ✓ **Build output type definition**
    - ➢ ***include $(BUILD_STATIC_LIBRARY)***
      - 최종 결과물은 LOCAL_MODULE.a의 형태를 갖는 static library 생성
      - out/target/product/$(TARGET_PRODUCT)/obj/STATIC_LIBRARIES/$(LOCAL_MODULE)_intermediates/$(LOCAL_MODULE).a
      - Root filesystem으로는 포함되지 않는다
  - ✓ ***LOCAL_MODULE_TAGS***
    - ➢ build/core/base_rules.mk 에 사용 tag option 지정
    - ➢ Android build system의 ***BUILDTYPE(e.g. eng)***에 따라 지정된 tag와 매칭이 될 경우만 해당 모듈을 컴파일 하도록 하는 지시자
    - ➢ ICS 버전에서는 ***optional, debug, eng, test, samples*** 태그를 지원함
    - ➢ optional 태그는 GingerBread서부터 생겨남
      - 각 PRODUCT 혹은 device의 PRODUCT_PACKAGES 에 해당 module이 지정되어 있지 않으면 전체 make시 root filesystem에 install 하지 않음

Android.mk

- **Android.mk를 구성하는 문법 & definition**
  - ✓ ***LOCAL_MODULE_TAGS(계속)***
    - ➢ **ex> LOCAL_MODULE_TAGS := eng optional**
      - **eng 빌드타입에서는 build가 됨**
      - **하지만, eng 이 외의 buildtype에서는 PRODUCT_PACKAGES 변수에 해당 모듈 이름이 포함되어 있지 않은 경우는 전체 build시 root filesystem에 install되지 않는다**
      - **➔ 만일 user 빌드타입으로 전체 build를 하면 PRODUCT_PACKAGES 변수에 지정되어 있지 않으면 root filesystem에 포함되지 않음**
    - ➢ **PRODUCT_PACKAGES 변수로의 지정**
      - **build/target/product/*.mk 파일에서 지정을 하거나**
      - **device/*.mk 파일등에서 지정한다**
      - **일반적으로 device 디렉토리의 $(PRODUCT)/device.mk 등에서 많이 지정된다.**