Android Device

From eLinux.org

This is a breakdown of the files build/envsetup.sh, Makefile and the files they use, which describes how a device is specified and how Android is configured for it.

For each file there are some comments and code lines from the make files or scripts, which describe what they are doing and which files they are using. Also the files which can be used as an example are presented and the commands how to search for them.

This text is for developers who want to add a new device or change the configuration of an existing device. This gives some indications which files are involved.

Contents

build/envsetup.sh

Some functions are defined by calling

```
. build/envsetup.sh
```

in the top directory.

Some environment variables are set by calling

```
lunch
```

in the top directory.

```
export TARGET_PRODUCT=$product
export TARGET_BUILD_VARIANT=$variant
export TARGET_SIMULATOR=false
export TARGET_BUILD_TYPE=release
```

vendorsetup.sh is searched at this places:

```
vendor/*/vendorsetup.sh
vendor/*/*/vendorsetup.sh
device/*/*/vendorsetup.sh
```

vendorsetup.sh

This file is executed by build/envsetup.sh, and can use anything defined in envsetup.sh.

In particular, you can add lunch options with the add_lunch_combo function:

```
add_lunch_combo full_crespo-userdebug
```

The values of the macros TARGET_PRODUCT and TARGET_BUILD_VARIANT are derived from the option name: add_lunch_combo $TARGET_PRODUCT-$TARGET_BUILD_VARIANT
In the above example the resulting values are TARGET_PRODUCT=full_crespo and TARGET_BUILD_VARIANT=userdebug.

These files can be used as an example:

```
find . -name vendorsetup.sh
```

```
./device/samsung/crespo/vendorsetup.sh
./device/samsung/crespo4g/vendorsetup.sh
./device/htc/passion/vendorsetup.sh
```

Makefile

Build process is started by calling

```
make
```

in the top directory.

The Makefile calls build/core/main.mk

build/core/main.mk

Set up various standard variables based on configuration and host information.

```
include $(BUILD_SYSTEM)/config.mk
```

This allows us to force a clean build - included after the config.make environment setup is done, but before we generate any dependencies. This file does the rm -rf inline so the deps which are all done below will be generated correctly

```
include $(BUILD_SYSTEM)/cleanbuild.mk
```

These are the modifier targets that don't do anything themselves, but change the behavior of the build. (must be defined before including definitions.make)

```
INTERNAL_MODIFIER_TARGETS := showcommands checkbuild all
```

Bring in standard build system definitions.

```
include $(BUILD_SYSTEM)/definitions.mk
```

build/core/config.mk

Various mappings to avoid hard-coding paths all over the place

```
include $(BUILD_SYSTEM)/pathmap.mk
```

Try to include buildspec.mk, which will try to set stuff up. If this file doesn't exist, the environemnt variables will be used, and if that doesn't work, then the default is an arm build

```
-include $(TOPDIR)buildspec.mk
```

Define most of the global variables. These are the ones that are specific to the user's build configuration.

```
include $(BUILD_SYSTEM)/envsetup.mk
```

Search for BoardConfig.mk in
$(SRC_TARGET_DIR)/board/$(TARGET_DEVICE)/BoardConfig.mk
device/*/$(TARGET_DEVICE)/BoardConfig.mk
vendor/*/$(TARGET_DEVICE)/BoardConfig.mk
and load the file

```
 include $(board_config_mk)
```

```
 include $(BUILD_SYSTEM)/dumpvar.mk
```

BoardConfig.mk

These files can be used as an example:

```
find . -name BoardConfig.mk
```

```
./device/samsung/crespo/BoardConfig.mk
./device/samsung/crespo4g/BoardConfig.mk
./device/htc/passion/BoardConfig.mk
./build/target/board/generic/BoardConfig.mk
./build/target/board/generic_x86/BoardConfig.mk
./build/target/board/emulator/BoardConfig.mk
./build/target/board/sim/BoardConfig.mk
```

build/buildspec.mk.default

This is a do-nothing template file. To use it, copy it to a file named "buildspec.mk" in the root directory, and uncomment or change the variables necessary for your desired configuration. The file "buildspec.mk" should never be checked in to source control.

Choose a product to build for. Look in the products directory for ones that work.
TARGET_PRODUCT

Choose a variant to build. If you don't pick one, the default is eng.
User is what we ship.
Userdebug is that, with a few flags turned on for debugging.
Eng has lots of extra tools for development.
TARGET_BUILD_VARIANT

CUSTOM_MODULES
TARGET_SIMULATOR

Set this to debug or release if you care. Otherwise, it defaults to release for arm and debug for the simulator.
TARGET_BUILD_TYPE

HOST_BUILD_TYPE
DEBUG_MODULE_ModuleName
TARGET_TOOLS_PREFIX
HOST_CUSTOM_DEBUG_CFLAGS
TARGET_CUSTOM_DEBUG_CFLAGS
CUSTOM_LOCALES
OUT_DIR
ADDITIONAL_BUILD_PROPERTIES
NO_FALLBACK_FONT
WEBCORE_INSTRUMENTATION
ENABLE_SVG
BUILD_ENV_SEQUENCE_NUMBER

build/envsetup.mk

Set up version information.

```
include $(BUILD_SYSTEM)/version_defaults.mk
```

If you update the build system such that the environment setup or buildspec.mk need to be updated, increment this number, and people who haven't re-run those will have to do so before they can build. Make sure to also update the corresponding value in buildspec.mk.default and envsetup.sh.

```
CORRECT_BUILD_ENV_SEQUENCE_NUMBER := 10
```

```
include $(BUILD_SYSTEM)/product_config.mk
```

TARGET_PRODUCT: sim full
TARGET_BUILD_VARIANT: eng user userdebug tests

build/core/version_defaults.mk

Handle various build version information.

Guarantees that the following are defined:
PLATFORM_VERSION
PLATFORM_SDK_VERSION
PLATFORM_VERSION_CODENAME
DEFAULT_APP_TARGET_SDK
BUILD_ID
BUILD_NUMBER

Look for an optional file $(BUILD_SYSTEM)/build_id.mk containing overrides of the defaults
INTERNAL_BUILD_ID_MAKEFILE

```
include $(BUILD_SYSTEM)/build_id.mk
```

build/core/build_id.mk

BUILD_ID is usually used to specify the branch name
BUILD_ID

DISPLAY_BUILD_NUMBER

build/product_config.mk

Provide "PRODUCT-<prodname>-<goal>" targets, which lets you build a particular configuration without needing to set up the environment.

```
TARGET_PRODUCT := $(word 1,$(product_goals))
TARGET_BUILD_VARIANT := $(word 2,$(product_goals))
```

Provide "APP-<appname>" targets, which lets you build an unbundled app.

Include the product definitions. We need to do this to translate TARGET_PRODUCT into its underlying TARGET_DEVICE before we start defining any rules.

PRODUCT_DEVICE is defined in the product file $(TARGET_PRODUCT).mk.
The product file $(TARGET_PRODUCT).mk is searched in the list of product make files $(PRODUCT_MAKEFILES).
PRODUCT_MAKEFILES is set in AndroidProducts.mk files.

```
$(call import-products,$(call get-product-makefiles,
      $(SRC_TARGET_DIR)/product/AndroidProducts.mk))
```

Convert a short name like "sooner" into the path to the product file defining that product.

```
INTERNAL_PRODUCT := $(call resolve-short-product-name, $(TARGET_PRODUCT))
```

```
TARGET_DEVICE := $(PRODUCTS.$(INTERNAL_PRODUCT).PRODUCT_DEVICE)
PRODUCT_LOCALES := $(strip $(PRODUCTS.$(INTERNAL_PRODUCT).PRODUCT_LOCALES))
PRODUCT_BRAND
PRODUCT_MODEL
PRODUCT_MANUFACTURER
```

```
PRODUCT_OTA_PUBLIC_KEYS
```

AndroidProducts.mk

This file should set PRODUCT_MAKEFILES to a list of product makefiles to expose to the build system. LOCAL_DIR will already be set to the directory containing this file.

This file may not rely on the value of any variable other than LOCAL_DIR; do not use any conditionals, and do not look up the value of any variable that isn't set in this file or in a file that it includes.

File device/samsung/crespo/AndroidProducts.mk

```
PRODUCT_MAKEFILES := \
    $(LOCAL_DIR)/full_crespo.mk
```

These files can be used as an example:

```
find . -name AndroidProducts.mk
```

```
./device/sample/products/AndroidProducts.mk
```

```
./device/samsung/crespo/AndroidProducts.mk
./device/samsung/crespo4g/AndroidProducts.mk
./device/htc/passion/AndroidProducts.mk
./build/target/product/AndroidProducts.mk
```

The command which returns the list of all AndroidProducts.mk files is defined in build/core/product.mk :

```
define _find-android-products-files
$(shell test -d device && find device -maxdepth 6 -name AndroidProducts.mk) \
  $(shell test -d vendor && find vendor -maxdepth 6 -name AndroidProducts.mk) \
  $(SRC_TARGET_DIR)/product/AndroidProducts.mk
endef
```

Product Files

Search for the files which can be used as an example:

```
grep -R PRODUCT_DEVICE device build
```

```
device/samsung/crespo/full_crespo.mk:PRODUCT_DEVICE := crespo
device/samsung/crespo4g/full_crespo4g.mk:PRODUCT_DEVICE := crespo4g
device/htc/passion/full_passion.mk:PRODUCT_DEVICE := passion
build/target/product/sdk.mk:PRODUCT_DEVICE := generic
build/target/product/generic.mk:PRODUCT_DEVICE := generic
build/target/product/generic_x86.mk:PRODUCT_DEVICE := generic_x86
build/target/product/core.mk:PRODUCT_DEVICE := generic
build/target/product/full_x86.mk:PRODUCT_DEVICE := generic_x86
build/target/product/full.mk:PRODUCT_DEVICE := generic
build/target/product/sim.mk:PRODUCT_DEVICE := sim
```

PRODUCT_DEVICE is used in these files

```
build/core/product.mk:    PRODUCT_DEVICE \
build/core/product_config.mk:TARGET_DEVICE := $(PRODUCTS.$(INTERNAL_PRODUCT).PRODUCT_DEVICE)
```

Add new device

Add the configuration files for the new device mydevice of the company mycompany.

Create AndroidProducts.mk

```
mkdir -p device/mycompany/mydevice
nano device/mycompany/mydevice/AndroidProducts.mk
```

```
PRODUCT_MAKEFILES := \
    $(LOCAL_DIR)/full_mydevice.mk
```

Create file full_mydevice.mk
Example is build/target/product/full.mk

```
nano device/mycompany/mydevice/full_mydevice.mk
```

```
$(call inherit-product, $(SRC_TARGET_DIR)/product/full_base.mk)
$(call inherit-product, $(SRC_TARGET_DIR)/board/generic/device.mk)

# Overrides
PRODUCT_NAME := full_mydevice
PRODUCT_DEVICE := mydevice
PRODUCT_BRAND := Android
PRODUCT_MODEL := Full Android on mydevice
```

Create file vendorsetup.sh

```
nano device/mycompany/mydevice/vendorsetup.sh
```

```
add_lunch_combo full_mydevice-eng
```

Create file BoardConfig.mk

Examples are
build/target/board/generic/BoardConfig.mk
device/samsung/crespo/BoardConfig.mk
device/samsung/crespo/BoardConfigCommon.mk

```
mkdir -p device/mycompany/mydevice
nano device/mycompany/mydevice/BoardConfig.mk
```

```
# config.mk
#
# Product-specific compile-time definitions.
#

# The generic product target doesn't have any hardware-specific pieces.
TARGET_NO_BOOTLOADER := true
TARGET_NO_KERNEL := true
TARGET_CPU_ABI := armeabi
HAVE_HTC_AUDIO_DRIVER := true
BOARD_USES_GENERIC_AUDIO := true

# no hardware camera
USE_CAMERA_STUB := true

# Set /system/bin/sh to mksh, not ash, to test the transition.
TARGET_SHELL := mksh

# CPU
TARGET_ARCH_VARIANT := armv7-a-neon
ARCH_ARM_HAVE_TLS_REGISTER := true
```

Configure Android for mydevice

```
. build/envsetup.sh
```

```
including device/htc/passion/vendorsetup.sh
including device/mycompany/mydevice/vendorsetup.sh
including device/samsung/crespo4g/vendorsetup.sh
including device/samsung/crespo/vendorsetup.sh
```

```
lunch
```

```
You're building on Linux

Lunch menu... pick a combo:
     1. full-eng
     2. full_x86-eng
     3. simulator
     4. full_passion-userdebug
     5. full_mydevice-eng
     6. full_crespo4g-userdebug
     7. full_crespo-userdebug

Which would you like? [full-eng] 5

============================================
PLATFORM_VERSION_CODENAME=AOSP
PLATFORM_VERSION=AOSP
TARGET_PRODUCT=full_mydevice
TARGET_BUILD_VARIANT=eng
TARGET_SIMULATOR=false
TARGET_BUILD_TYPE=release
TARGET_BUILD_APPS=
TARGET_ARCH=arm
TARGET_ARCH_VARIANT=armv7-a-neon
HOST_ARCH=x86
HOST_OS=linux
HOST_BUILD_TYPE=release
BUILD_ID=OPENMASTER
============================================
```

Build Android for mydevice

```
make -j4
```

```
Combining NOTICE files: out/target/product/mydevice/obj/NOTICE.html
Target system fs image: out/target/product/mydevice/obj/PACKAGING/systemimage_intermediates/syst
Install system fs image: out/target/product/mydevice/system.img
Installed file list: out/target/product/mydevice/installed-files.txt
```