Blog

From Zero to Boot: Porting Android to your ARM platform

Posted by Vasileios Laganakos in Android Community on Sep 18, 2013 4:39:55 PM

This blog was originally posted on 2nd June 2011 on blogs.arm.com

This article describes how to get Android running on *your* favourite ARM-based System on Chip (SoC) board. We run through the overall procedure and point out potential pitfalls and other things that you may encounter.
Since the Android software stack was primarily designed around the ARM Architecture, there are not many things that need amending to get it to work on another ARM platform.

We assume that your workstation has Ubuntu (10.10 or later) Operating System installed, and that you have already followed the instructions found at [1] to be ready to build Android sources. These instructions have been tested with Ubuntu 10.10, but they should be compatible with other GNU/Linux OSes.

Terminology

For the purposes of this document, we use the following terms.

Mainline kernel	the kernel that you can get from the mainline (i.e. http://www.kernel.org); implies that no changes have been made to it. Also known as "vanilla kernel".
Reference kernel	this is the kernel that you have for your board; that is most likely a vanilla kernel, plus any patches required to support your board.

Merged kernel	the kernel that is the result of merging Android patches to the Reference kernel. A Merged kernel should support Android.
Android patches	the patches that you extract from the Google Android kernel. These are the changes from the point that a Mainline Kernel version has been imported to the Android Kernel tree. See http://android.git.kernel.org
Android kernel	the kernel that you can get from Google's Android Open Source Project repository. It contains Google's changes to the Mainline kernel, to support Android. You can get it from :http://android.git.kernel.org/kernel/common.git

Outline

The procedure is split into two parts:

- Kernel
  - Get a Linux kernel that works on your board (Reference kernel)
  - Get the matching Android kernel version for the Android release you want use (e.g. Gingerbread) - the Android kernel version is loosely coupled with an Android release.
  - Extract the patches for the Android kernel (around 2 MB in size) and merge the Android patches with the Reference Linux kernel for your board - or merge the two kernel trees
  - Configure the Merged kernel for use in Android
  - Build the Merged kernel
- Userspace - the Android filesystem
  - Add the Android build target of your SoC in the devices folder of the Android Source tree
  - Build Android for your SoC
  - Configure init.rc to fit your booting media
  - Boot

We must stress the importance of doing the *"configure"* step carefully in both parts of the procedure. If it is not done correctly then it will not be possible to complete the task successfully.
You might want to get Android working on a platform that already has an Android port available. We provide pre-built kernel binaries, kernel sources and kernel .config *files, for certain ARM boards such as the ARM Versatile Express . Please see [2] for more detail.*

Part 1 - Kernel

The following diagram shows the procedure of merging of the Reference with the Android kernel.

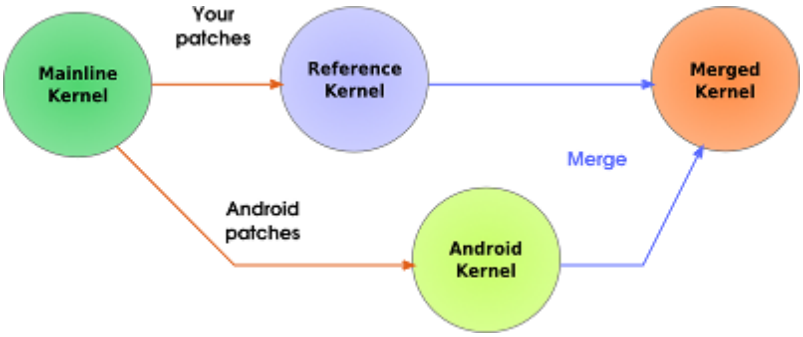

Diagram 1. Kernel Merging Procedure.

a. Known working Linux Reference kernel

Make sure that you have a working kernel and a Linux root filesystem, functional on your board, i.e. you can boot to a GNU/Linux graphical environment.
If this is not possible, there is no point attempting to complete this exercise.

> Note : The Linux root filesystem is only required to test the Merged Kernel, not for the Android port to your SoC.

You need a kernel that boots and provides to the system all of the appropriate support required for a windowing system (e.g. graphics drivers for the framebuffer or your graphics chip) as a reference point. This will be called your Reference kernel.
It should be noted here that every Android release after Cupcake requires double buffering and page flipping support from the graphics system. If your Reference kernel does not support this, then Android releases after Cupcake will not work.
The .config file of the Reference kernel will be used as a base for the Merged kernel. Keep a copy of this as a reference for later use.
For the purpose of this article, we assume that you have a 2.6.38-based functional Reference kernel for your board. You may use any kernel revision you wish, but keep in mind that it may result in deviating from the information in this article.

b. Android kernel

Find an Android kernel which has the same revision as your Reference kernel. If it is not possible, use a revision that is closest, as this will simplify the procedure.
Revision mismatch of the two kernels will probably result in spending more time merging and debugging. From experience, merging a Reference kernel and the Android patches of the same revision requires about a day; in the case of revision mismatch, more time may be required.

c - Merge Android Kernel Changes

In this step you can either obtain the Android patches from the Android kernel source, or use git to merge the two kernel trees using their common point in time.
The first approach will produce a patch file that can be applied to the Reference kernel, while using git-merge will produce a single tree that will keep the patching history, and maintain all of the individual commits.
Please note that you will have to do either c.i or c.ii, not both!

c - i. Extract Single Patch

The expected size of the patch is about 2MB. To extract the patches, find the point in time (using "git log") where the Mainline kernel was imported to the Android kernel source tree.
Go to the top level of the Android kernel source tree and run:

```
git log --pretty=oneline --format="%Cgreen%h %Cr
--grep="Linux 2.6." -n 20
```

which will generate 20 one-line log entries that have the expression "Linux 2.6." in the subject or in the commit message. The output should look like:

```
521cb40 Linux 2.6.38
a5abba9 Linux 2.6.38-rc8
dd9c154 Linux 2.6.38-rc7
f5412be Linux 2.6.38-rc6
85e2efb Linux 2.6.38-rc5
100b33c Linux 2.6.38-rc4
ebf5382 Linux 2.6.38-rc3
```

```
1bae4ce Linux 2.6.38-rc2
c56eb8f Linux 2.6.38-rc1
aa0adb1 batman-adv: Use "__attribute__" shortcut macros
f5afcd3 drm/i915/crt: Check for a analog monitor in case
3c0eee3 Linux 2.6.37
387c31c Linux 2.6.37-rc8
90a8a73 Linux 2.6.37-rc7
b0c3844 Linux 2.6.37-rc6
cf7d7e5 Linux 2.6.37-rc5
e8a7e48 Linux 2.6.37-rc4
3561d43 Linux 2.6.37-rc3
5bd5a45 x86: Add NX protection for kernel data
64edc8e x86: Fix improper large page preservation
```

Since you need the patches to be on top of the 2.6.38 Mainline kernel release, run the following command:

```
git diff 521cb40 HEAD > 2.6.38-to-Android.patch
```

This gives you a patch file (named 2.6.38-to-Android.patch) containing the changes to be merged. Check for merging conflicts, before attempting to perform the actual merge, by running:

```
git apply --test [path/to/2.6.38-to-Android.patch]
```

which produces a list of merging conflicts (if any). Make sure that you resolve these conflicts and that the patch applies cleanly to your tree before you continue.

c - ii. Use git-merge

This approach merges the Android kernel tree with your local Reference kernel tree. First, add the Android kernel tree as one of your remote trees:

```
git remote add [android tree] \
git://android.git.kernel.org/kernel/common.git
```

where [android tree] is the local name for the remote Android repository. Now enter:

```
git fetch [android tree]
```

to get all the git information for that tree (such as branch names, commits and tags). To identify the common point where the merge will be based, enter:

```
git merge-base armdroid-2.6.38 [android tree]/android-
```

where armdroid-2.6.38 is the local branch name for your Reference kernel that you wish to merge with the Android kernel tree, and [android tree]/android-2.6.38 is the branch of the 2.6.38 Android Kernel at the remote repository. This returns the hash of the commit at which the two branches diverged. E.g.:

```
a5abba989deceb731047425812d268daf7536575
```

You can use this to get an estimate of the number and size of changes made from that point in time, to the head of your Reference kernel tree. Then, attempt an automatic merge using:

```
git merge armdroid-2.6.38 [android tree]/android-2.6.3
```

Any conflicts will be reported - review these using:

```
git status
```

For conflict resolution git mergetool can be quite handy, and it allows you to use your favourite diff tool. For example:

```
git mergetool --tool=vimdiff kernel/printk.c
```

lets you review and resolve the merging conflict for kernel/printk.c using vimdiff. When you have resolved the conflicts, commit the changes using:

```
git commit -a
```

d. Configure the Merged kernel for Android

In order to get the Merged kernel configured correctly for Android, you must have a known working configuration tested with a Linux root filesystem.

This should be tested by building a kernel image from the merged source tree, using the known working defconfig or .config file saved in Part 1a, and checking that it boots your existing working Linux root filesystem.

Note: Be aware that due to the Android parts in the Merged kernel, you might get asked about some options that were not available in the reference configuration of your Reference kernel.

If it does not boot, take a step back and make sure that what was merged from the Android kernel source tree did not break anything in your Reference kernel. Checking the files in which you have resolved any

merge conflicts should be a good place to start.

If it does boot, you can then add options to your working configuration that are required for Android.

These are:

- power_management : CONFIG_PM=y
  - CONFIG_HAS_EARLYSUSPEND=y
  - CONFIG_SUSPEND=y
  - wakelocks : CONFIG_HAS_WAKELOCK=y
    - CONFIG_WAKELOCK=y
    - CONFIG_EARLYSUSPEND=y
    - CONFIG_USER_WAKELOCK=y
    - CONFIG_FB_EARLYSUSPEND=y
- ashmem : CONFIG_ASHMEM=y
- pmem : CONFIG_ANDROID_PMEM=y
- switch class support : CONFIG_SWITCH=y
- staging drivers : CONFIG_STAGING=y
  - CONFIG_ANDROID=y
  - binder : CONFIG_ANDROID_BINDER_IPC=y
  - logger : CONFIG_ANDROID_LOGGER=y
  - ram console : CONFIG_ANDROID_RAM_CONSOLE=y
- usb_gadgets with at least adb selected - CONFIG_USB_GADGET=y
  - CONFIG_USB_ANDROID=y
  - CONFIG_USB_ANDROID_ACM=y
  - CONFIG_USB_ANDROID_ADB=y
  - CONFIG_USB_ANDROID_MASS_STORAGE=y
  - CONFIG_USB_ANDROID_MTP=y
- power supply : CONFIG_POWER_SUPPLY=y
  - CONFIG_PDA_POWER=y

Note that the names above are those which can be found in the 2.6.38 Android kernel configuration - be aware that these tend to change from version to version!

We have also found that in multiprocessor systems, CPU hotplugging support is required. You should enable this if your ARM based SoC has more than one core in order to boot a kernel with multiprocessor support.
To get the Merged kernel to work correctly for Android, the above configuration options are the minimum options required.

e. Build the Merged kernel

Build the Merged kernel with:

```
make ARCH=arm CROSS_COMPILE=[path-to-arm-gcc] uImag
```

which will produce a kernel Image, uImage and zImage in arch/arm/boot directory in the Merged kernel source tree. Replace the [path-to-arm-gcc] with the path to the latest version of Linaro or CodeSourceryARM toolchain that you have installed. For this exercise we used Sourcery G++ Lite 2010.09-50) 4.5.1.
Note that you need the uboot-mkimage package in order to generate a uImage. If you do not want a U-boot wrapped Linux kernel image, substitute uImage with Image above.

You now need the Android root filesystem in order to complete your port.

Part 2 - Userspace - The Android filesystem

a. Add a device in the tree

Each build target defines the configuration of the ARM based SoC/board and selects which sources should be built for Android.

The build target directory that the Android build system uses depends on the Android version. In Froyo and Gingerbread the location is:

```
[android_root]/device
```

whereas in Eclair it is:

```
[android_root]/vendor
```

In this article, we assume that you are using Gingerbread.

a - i. Adding a build target

Below is an example of adding a couple of ARM generic Android build

targets, along with a mock target.

Create a directory in [android_root]/device, with the desired name. E.g.:

```
mkdir [android_root]/device/arm
```

Enter the new directory and create a products folder:

```
mkdir [android_root]/device/arm/products
```

In the products folder you need an AndroidProducts.mk file that lists all of the products that you have under the arm folder. This should be as follows:

```
PRODUCT_MAKEFILES := \
$(LOCAL_DIR)/armboards_v7a.mk \
$(LOCAL_DIR)/armboards_v7a_noneon.mk \
$(LOCAL_DIR)/another_product.mk \
```

As this implies, you also need to have one "Product_Makefile" per product in the[android_root]/device/arm/products/ folder. Therefore, for our example you need:

- armboards_v7a.mk
- armboards_v7a_noneon.mk
- another_product.mk

Each of these files must contain the following, adapted to match the product and device naming :

```
$(call inherit-product, $(SRC_TARGET_DIR)/product/generi
#
# Overrides
PRODUCT_NAME := [product_name]
PRODUCT_DEVICE := [board_name]
```

where PRODUCT_NAME is the build target name used by the Android build system, and PRODUCT_DEVICE defines the name of the directory that contains the files which describe the device.

For our example the first product is :

```
$(call inherit-product, $(SRC_TARGET_DIR)/product/generi
#
# Overrides
```

```
PRODUCT_NAME := armboard_v7a
PRODUCT_DEVICE := armboard_v7a
```

Note that the PRODUCT_NAME does not have to be the same as the PRODUCT_DEVICE. For example, if you were to add support for ARM Versatile Express board, the contents of the vexpress.mk file could be :

```
$(call inherit-product, $(SRC_TARGET_DIR)/product/generi
#
# Overrides
PRODUCT_NAME := vexpress
PRODUCT_DEVICE := cortex-a9
```

and the files that define/configure that device would be in a "cortex-a9" directory. You are free to choose the name you wish for your product and device, limited to the characters allowed. For example, you cannot use minus "-" because this is used to separate the PRODUCT prefix and the eng, user or test, suffix from the product name in the build command line.

Following the example above, create the required directories:

```
mkdir [android_root]/device/arm/armboard_v7a/
mkdir [android_root]/device/arm/armboard_v7a_noneon/
mkdir [android_root]/device/arm/another_product/
```

and populate them accordingly.

Each one of these directories should contain at least the "Android.mk" and "BoardConfig.mk" files. The first one is the makefile for the product and the second, as the name implies, is the config file for the product. The Google guidelines [3] for what the Android.mk file should contain are :

```
# make file for new hardware  from
#
LOCAL_PATH := $(call my-dir)
#
# this is here to use the pre-built kernel
ifeq ($(TARGET_PREBUILT_KERNEL),)
```

```
TARGET_PREBUILT_KERNEL := $(LOCAL_PATH)/kernel
endif
#
file := $(INSTALLED_KERNEL_TARGET)
ALL_PREBUILT += $(file)
$(file): $(TARGET_PREBUILT_KERNEL) | $(ACP)
$(transform-prebuilt-to-target)
#
# no boot loader, so we don't need any of that stuff..
#
LOCAL_PATH := vendor/[company_name]/[board_name]
#
include $(CLEAR_VARS) # # include more board specific st
#
```

which, when adapted to our example, becomes:

```
# make file for ARMv7-A based SoC
#
LOCAL_PATH := $(call my-dir)
#
# this is here to use the pre-built kernel
ifeq ($(TARGET_PREBUILT_KERNEL),)
TARGET_PREBUILT_KERNEL := $(LOCAL_PATH)/kernel
endif
#
file := $(INSTALLED_KERNEL_TARGET)
ALL_PREBUILT += $(file)
$(file): $(TARGET_PREBUILT_KERNEL) | $(ACP)
$(transform-prebuilt-to-target)
#
# no boot loader, so we don't need any of that stuff..
#
LOCAL_PATH := device/arm/armboard_v7a
#
include $(CLEAR_VARS)
#
```

```
# include more board specific stuff here? Such as Audio
#
PRODUCT_COPY_FILES += \
$(LOCAL_PATH)/armboard_v7a.kl:system/usr/keylayout/armboard_v7a.kl
```

Note the added line in the PRODUCT_COPY_FILES definition, that contains two locations separated by a colon. The first is the location of the keyboard file in the local Android source tree, and the second is where it will be installed on the target. This is added because we wanted to have our own, tweaked, keyboard layout.

The BoardConfig.mk file for armboard_v7a product is:

```
# These definitions override the defaults in config/conf
#
# TARGET_NO_BOOTLOADER := false
# TARGET_HARDWARE_3D := false
#
TARGET_CPU_ABI := armeabi-v7a
TARGET_CPU_ABI2 := armeabi

TARGET_NO_KERNEL := true
TARGET_ARCH_VARIANT := armv7-a-neon

BOARD_USES_GENERIC_AUDIO := true
USE_CAMERA_STUB := true
```

where you define that you are using the armeabi-v7a and that the target architecture variant is ARMv7-A with NEON . *Note the support for 2 ABIs that was introduced in Froyo*. This allows the distribution to run native ARM applications built for one specific architecture version (using Android NDK).

For any further system properties configuration, add a system.prop file containing the things you need, at the same location as the above files (device/arm/[DEVICE_NAME]). Google's guidelines [3] provide a template for this :

```
# system.prop for
# This overrides settings in the products/generic/system
```

```
#
# rild.libpath=/system/lib/libreference-ril.so
# rild.libargs=-d /dev/ttyS0
```

For any other products/device you wish to add, follow the same procedure outlined here (2.a.i).

You should now be ready to build the Android file system for your SoC, with the device folder containing all the necessary configuration and build files. For our example, the device folder should look like this:

```
  arm/
+-- armboard_v7a
| +-- Android.mk
| +-- armboard_v7a.kl
| +-- BoardConfig.mk
| \-- system.prop
+-- armboard_v7a_noneon
| +-- Android.mk
| +-- armboard_v7a_noneon.kl
| +-- BoardConfig.mk
| \-- system.prop
+-- another_product
| +-- Android.mk
| +-- another_product.kl
| +-- BoardConfig.mk
| \-- system.prop
\-- products
 +-- AndroidProducts.mk
 +-- armboard_v7a.mk
 +-- armboard_v7a_noneon.mk
 \-- another_product.mk
```

b. Build Android for your SoC

Build the Android filesystem for one of your added targets, by going to the top directory of the Android source tree, and entering:

```
make PRODUCT-[PRODUCT_NAME]-eng
```

e.g.:

```
make PRODUCT-armboard_v7a-eng
```

Note: You do not need to build Android as the root user.
After some time, and if everything was set up correctly, you will get the Android root filesystem and a compressed version of it at:

```
[android_root]/out/target/product/[product_name]
```

For our example :

```
[android_root]/out/target/product/armboard_v7a
```

c. Configure the root filesystem

We use the root filesystem over NFS. For that, make sure you have the NFS server set up and serving the location you wish to place the Android root filesystem.
Note: To use a root filesystem on a remote NFS Server, your Reference Kernel (and therefore your Merged kernel) must support Rootfs over NFS (CONFIG_ROOT_NFS=y *).*
The line in the /etc/exports file should be like:

```
/srv/nfs4/android-fs   *(rw,sync,no_root_squash,no_
```

where the path at the beginning is the Android root filesystem location, and the last part is the NFS configuration string.
Now you need to copy the contents of the root directory from the Android source tree to the NFS served location :

```
$ cp -a [android_root]/out/target/product/armboard_
/var/nfs4/android-fs
```

and the contents of the system directory from the Android source tree in the system placeholder - now in the NFS served location :

```
$ cp -a [android_root]/out/target/product/armboard_
/var/nfs4/android-fs/system
```

Use "sudo" at the beginning of each command, if you need to have root privileges to copy things to the defined destination.
Ensure that the ownership and permissions of the files, as they come out of the build system, have not been modified. - cp -a should have done this for you.
For this filesystem to work, you also need to make changes in the [android_root_filesystem]/init.rc file. For Gingerbread you need to do the following:

- Locate and comment out the following lines

    - mount rootfs rootfs /ro remount
    - mount yaffs mtd@system /system
    - mount yaffs2 mtd@system /system ro remout
    - mount yaffs2 mtd@userdata /data nosuid nodev
    - mount yaffs2 mtd@cache /cache nosuid nodev
    - on fs
- To get root privileges in the shell, change:

```
...
    service console /system/bin/sh
    console
    disabled
    user shell
    group log
...
```

- to:

```
...
 service console /system/bin/sh
 console
 disabled
 user root
 group log
...
```

- To get DNS working, add the indicated line # basic network init ifup lo hostname localhost domainname localdomain ---> setprop net.dns1 [DNS] and replace [DNS] with your local DNS server.

Note : The instructions in this step can be also used for deploying the Android root filesystem on other media (such as USB sticks or MMC).

d. Boot

Load the kernel on your board, and set the appropriate boot arguments in order boot successfully.
For our example and for the VExpress board that uses U-Boot, we use the following bootargs:

```
root=/dev/nfs \
```

```
nfsroot=[NFS_IP]:/srv/nfs4/android-fs,nolock,wsize=1
ip=dhcp console=ttyAMA init=init noinitrd mem=1024M
```

Android should come up shortly. Note that the first boot will take longer since Android is optimizing the .dex files (this is known as the dexopt operation). See [4] for more details on dexopt.

Should any minor issues appear, experiment a bit more with the kernel options and the init.rcconfiguration, based on the information you get from Android's logcat and the system's dmesg.

References

[1] http://source.android.com/source/initializing.html
[2] http://linux-arm.org/LinuxKernel/LinuxAndroidPlatform
[3] http://www.netmite.com/android/mydroid/development/pdk/docs/build_new_device.html
[4] http://www.netmite.com/android/mydroid/dalvik/docs/dexopt.html

9 Comments

Alban Rampon Oct 14, 2013 1:02 AM

Ported Comments

Hedge

10 November 2011 - 09:34 AM

This is the first time I am trying Porting topic. I am trying to port linux to Cortex-A8.

Wanted some more details on "Make sure that you have a working

kernel and a Linux root filesystem, functional on your board".

The Pre-built U-Boot and Linux Images available in http://arm.com/commu...ement/linux.php are board specific? Or can I use the same images on Beagleboard as well? Because here its mentioned as images are for RealView PB-A8 platforms.

Thanks,
Hegde.

vlaganakos

10 November 2011 - 06:32 PM

Hegde, on 10 November 2011 - 10:34 AM, said:

This is the first time I am trying Porting topic. I am trying to port linux to Cortex-A8.

Hi Hedge,

Hope that you will enjoy the porting ride! :)

Hegde, on 10 November 2011 - 10:34 AM, said:

Wanted some more details on "Make sure that you have a working kernel and a Linux root filesystem, functional on your board".

The Pre-built U-Boot and Linux Images available in http://arm.com/commu...ement/linux.php are board specific? Or can I use the same images on Beagleboard as well? Because here its mentioned as images are for RealView PB-A8 platforms.

The u-boot and Linux images that you mention are board-specific, which means they won't work on your BeagleBoard.

You will need to get the kernel sources for BeagleBoard to build a kernel with the default configuration for Linux, and ensure that it works with a known functional Linux filesystem - e.g. an Ubuntu or Debian filesystem for ARM. The http://beagleboard.org/ and http://elinux.org/BeagleBoard pages should

have information for this part of the porting stage.

Once you have BeagleBoard working with the Linux kernel you built from source, you are ready to go the next part of the porting procedure: merge the BeagleBoard kernel source tree with the Android kernel tree, configure it for Android, and get it tested with the same Linux filesystem.

If you are not interested in doing the port yourself, you can instead get the pre-built Android package from Linaro at:

Kind regards,
Vassilis L.

Actions Like (1)

Alban Rampon Oct 14, 2013 1:09 AM

Vasileios Laganakos

22 November 2011 - 08:22 PM

> Hello,
> xaviergr, on 20 November 2011 - 06:13 AM, said:
>
> > Can this guide be applied to Ice Cream Sandwich too?
>
> Yes, the whole approach works for Android ICS; we used it here :) The patches that we currently provide need some updating, to support some changes in Android ICS. We will shortly provide files that have been tested and working with Android ICS, as well as an update to the above instructions.
>
> xaviergr, on 20 November 2011 - 06:13 AM, said:
>
> > When compiling the ICS sources, it complains about the PRODUCT_COPY_FILES line in the Android.mk files we created. Removing that line will complete the compilation process, but the binaries I am getting (servicemanager et al.) terminate immediately when the kernel tries to execute them.
>
> I presume that you have configured your kernel accordingly, as well as

with support for Thumb and NEON, so that it can run a Linux distro. We need to ensure these, so the processes are not terminated simply because they cannot be executed; for example when you try to run ARM NEON code while your kernel does not support that. Also make sure that you have enabled the appropriate flags for Android mentioned in the article above.

In addition, you might want to "tell" to the Android build system that your ARMv7 core that is used on the platform, has the TLS register in core/combo/arch/arm/armv7-a-neon.mk, by adapting the patch from: http://linux-arm.org...d.patch;hb=HEAD

Now provided that these points are okay, the following steps are a temporary workaround, that works if you are trying to port Android ICS to a device that does not have a GPU, and therefore will need to fall back to software rendering.

- perform the outlined changes in `init.rc` described in the article above
- in `Android.mk`, comment out the `PRODUCT_COPY_FILES` line (as you've already done)
- in the `BoardConfig.mk` file, uncomment the `TARGET_HARDWARE_3D` flag and set it to "false"
- edit`frameworks/base/packages/SystemUI/src/com/android/systemui/ImageWallpaper.java`in Android ICS sources, and make the following change around line 570, that will prevent WallpaperService from raising an exception and failing to start because it cannot find and use EGL.

```
mEglConfig = chooseEglConfig();
if (mEglConfig == null) {- throw new RuntimeException("eglConfig not initialized");+ return false;+ //throw new RuntimeException("eglConfig not initialized");
}
```

- set `dalvik.vm.heapsize=128m` in `system/build.prop` in your built Android filesystem.

These should allow you to boot Android ICS.

Please let us know how it goes. Stay tuned for our next set of patches (at http://linux-arm.org...=fs/src;hb=HEAD) that will support Android ICS! ;)

Kind regards,
Vassilis L.

0

xaviergr

23 November 2011 - 02:25 AM

> ARM Vassilis, on 22 November 2011 - 08:22 PM, said:
>
> > Hello,
> >
> > Yes, the whole approach works for Android ICS; we used it here :) The patches that we currently provide need some updating, to support some changes in Android ICS. We will shortly provide files that have been tested and working with Android ICS, as well as an update to the above instructions.
>
> Indeed, after my post I remembered the TSL register patch. Applying that change made the binaries to work once again.
>
> The TARGET_HARDWARE_3D was pretty self explanatory. It is quite interesting though that ICS will boot even without double buffering on the frame buffer, something that was explicitly required on Gingerbread.
>
> ARM Vassilis, on 22 November 2011 - 08:22 PM, said:
>
> > edit frameworks/base/packages/SystemUI/src/com/android/systemui/ImageWallpaper.java in Android ICS sources, and make the following change around line 570, that will prevent WallpaperService from raising an exception and failing to start because it cannot find and use EGL.
>
> My system booted successfully with UI, though as you describe without wallpaper and an annoying System UI exception. I will try the change you propose in order to get rid of that popup. All the rest seem to work.
>
> Thanks once again.
>
> 0

Vasileios Laganakos

05 December 2011 - 06:02 PM

> Hello, We have released patches for Android ICS 4.0.1 sources, that will allow you to build it and get a working FS, for boards that don't have a GPU. Please find them at: http://linux-arm.org...andwich;hb=HEAD Any news xaviergr? Did it work for you

Kind regards,
Vassilis L.

0

RKL

08 December 2011 - 11:36 AM

Hi, I have been trying to port ICS file system on S3C6410. I have downloaded the ICS Android source code and have added the device specific folder and the make files. When I am trying to compile i get the following error: build/core/main.mk:542:*** The following variables have been changed: PRODUCT_COPY_FILES. Stop Please let me know where is the problem.

0

Vasileios Laganakos

08 December 2011 - 11:47 AM

Hello RKL,

RKL, on 08 December 2011 - 11:36 AM, said:

> Hi, I have been trying to port ICS file system on S3C6410. I have downloaded the ICS Android source code and have added the device specific folder and the make files. When I am trying to compile i get the following error: build/core/main.mk:542:*** The following variables have been changed: PRODUCT_COPY_FILES. Stop Please let me know where is the problem.

Please see my previous comment, that contains a link to the updated patches for Android ICS. The git commit contains all the information you need to get to build Android ICS, http://linux-arm.org...74beba114f03fb5

Let us know how it goes!

Cheers,
Vassilis L.

Actions Like (1)

Alban Rampon Oct 14, 2013 1:23 AM

xaviergr

03 January 2012 - 07:56 PM

> ARM Vassilis, on 05 December 2011 - 06:02 PM, said:
>
> Hello, We have released patches for Android ICS 4.0.1 sources, that will allow you to build it and get a working FS, for boards that don't have a GPU. Please find them at: http://linux-arm.org...andwich;hb=HEADAny news xaviergr? Did it work for you
>
> Kind regards,
> Vassilis L.

Hello Vassilis,

Yes most things work except the workaround for the SystemUI exception. I had to stop it by executing "pm disable com.android.systemui".

Thanks.

Vasileios Laganakos

09 January 2012 - 11:04 AM

Hello xaviergr

> xaviergr, on 03 January 2012 - 07:56 PM, said:
>
> Yes most things work except the workaround for the SystemUI exception. I had to stop it by executing "pm disable com.android.systemui".

Hmmm, this is strange. With the released set of patches for ICS, I don't get the SystemUI exception. If you use these patches, please make sure that the

/system/etc/graphics/OpenGLwallpapers

file exists and it contains a single word "disabled". If all of this holds, and you still get the SystemUI exception, please let me know!

Thanks,
Vassilis L.

xaviergr

19 January 2012 - 11:06 PM

> ARM Vassilis, on 09 January 2012 - 11:04 AM, said:
>
> Hello xaviergr
>
> Hmmm, this is strange. With the released set of patches for ICS, I don't get the SystemUI exception. If you use these patches, please
> make sure that the
>
> /system/etc/graphics/OpenGLwallpapers
>
> file exists and it contains a single word "disabled". If all of this holds, and you still get the SystemUI exception, please let me know!
>
> Thanks,
> Vassilis L.

Sorry for the late reply.

I found some time to test things out, and it seems that the error was totally on my part. It works perfectly now.

Thanks once again.

mohammads

25 January 2012 - 12:58 PM

Do you mean it's possible to install Android or Linux on any PCB design,any vendor,any hardware ARM CPU ? Am I right ? :o

Vasileios Laganakos

25 January 2012 - 01:09 PM

> xaviergr, on 19 January 2012 - 11:06 PM, said:
>
> Sorry for the late reply.
>
> I found some time to test things out, and it seems that the error was totally on my part. It works perfectly now.
>
> Thanks once again.

No worries xaviergr, I'm glad everything worked for you in the end!

Cheers,
Vassilis L.

insink71

05 February 2012 - 01:45 AM

This is a fantastic guide. I am an android hobbyist on the journey toward developer status. Since you seem quite knowledgeable, could you point me toward a clear device tree building guide [for new devices] &/or software solution. I am good at copying and slightly modifying, but I'd like to build recoveries, aosp roms [like CyanogenMod], etc. These usually begin with the device tree, and as I understand it.. it has to be either made from scratch or copied and modified from someone that already has a working device tree [binaries are the manufacturer's and are proprietary]. I saw the tracing software ARM has and got to thinking, "maybe these tools could write the device tree for me."; well, in a way... help me put it all together in my head if you would be so kind. I am currently working with the Wildfire S which sports a msm8227 and various other hardware [ref http://www.phonearen...e-S-CDMA_id5641]. HCDRJacob made a device tree for marvel [the gsm counterpart to the phone I'm using]; which I forked on github [ref https://github.com/i...el/tree/patch-1] which I'll probably just edit to match the marvelc build.prop settings. Just wanted to grasp it better. Tired of just being a copy [modify if necessary] and paste kind of guy. Thanks again for this wonderful android kernel building guide. Rob

Vasileios Laganakos

17 February 2012 - 06:42 PM

Hi Rob,

> insink71, on 05 February 2012 - 01:45 AM, said:
>
> This is a fantastic guide.

Thank you very much :)

> insink71, on 05 February 2012 - 01:45 AM, said:
>
> I am an android hobbyist on the journey toward developer status. Since you seem quite knowledgeable, could you point me toward a clear device tree building guide [for new devices] &/or software solution. I am good at copying and slightly modifying, but I'd like to build recoveries, aosp roms [like CyanogenMod], etc. These usually begin with the device tree, and as I understand it.. it has to be either made from scratch or copied and modified from someone that already has a working device tree [binaries are the manufacturer's and are proprietary]. I saw the tracing software ARM has and got to thinking, "maybe these tools could write the device tree for me."; well, in a way... help me put it all together in my head if you would be so kind. I am currently working with the Wildfire S which sports a msm8227 and various other hardware [refhttp://www.phonearen...e-S-CDMA_id5641]. HCDRJacob made a device tree for marvel [the gsm counterpart to the phone I'm using]; which I forked on github [ref https://github.com/i...el/tree/patch-1] which I'll probably just edit to match the marvelc build.prop settings. Just wanted to grasp it better. Tired of just being a copy [modify if necessary] and paste kind of guy. Thanks again for this wonderful android kernel building guide. Rob

Right, if I understood correctly you want to add a new device in the Android Source Tree (under the`device` folder). We actually had based the code we share for Android (please check our git repo) on the code that is already in the source tree (see `device/sample`).

I guess this is just another case where the information/documentation is in the code. It is the same approach with the one we follow here to maintain the port to our devices.

Now at the location of the link that you gave, they indeed use some binaries that they have produced; these are specific to the device you want to add/support. I guess if you plan to ship these with the code, you first must be sure that you are legally cleared to do that, and of course make sure that they work :)

Please let us know if this answers your question.

insink71, on 05 February 2012 - 01:45 AM, said:

Thanks again for this wonderful android kernel building guide.

You are very much welcome!

Vassilis L.

hegde

18 February 2012 - 02:33 AM

ARM Vassilis, on 22 November 2011 - 06:16 PM, said:

They are both board-specific :) Different boards might have different settings that the bootloader is based on.
As far as the Linux kernel is concerned you can find the default configuration files for the supported boards of an architecture in your linux source code.
For example at

arch/arm/configs

you have the defconfigs for the supported ARM boards.

You are right for the boot loader, but something similar holds for the linux kernel too. Different boards have different memory mappings of hardware (i.e. a hardware device corresponds to a specified address in memory). Hence, for each board you need its bootloader and its linux kernel.

If you are referring to the .config file, it is the file that holds

the configuration of the kernel you are building.

No problem ;)

Kind regards,
Vassilis L.

Thanks for your guide Vassilis.
As you mentioned in the beginning, it is a wonderful journey :)
Keep posting such kind of articles which are very useful and helpful.
Thanks once again...!!

-Hegde

insink71

25 March 2012 - 03:53 AM

Still love this guide.. best reference I've found.My question today pertains to the recent merge news of mainline and android kernel.I assume [at least in near future] this will make it easier for developers and maybe just make porting single-arm [no pun intended]; in other words mainline -> reference = done. However, most reference kernels I've spied are using a 2.xx kernel base [when I finally see manufacturers release their kernel sources]...So my question: Does mainline kernel need to be of similar build? Or can a 3.xx mainline be made to work [easily] with a 2.xx reference kernel?Thanks again for your time.Regards,Rob

Vasileios Laganakos

19 April 2012 - 09:51 AM

Hi Rob,

> insink71, on 25 March 2012 - 03:53 AM, said:
>
> Still love this guide.. best reference I've found.

Thanks! I'm happy you find it useful! :)

insink71 said:

> My question today pertains to the recent merge news of mainline and android kernel.I assume [at least in near future] this will make it easier for developers and maybe just make porting single-arm [no pun intended]; in other words mainline -> reference = done. However, most reference kernels I've spied are using a 2.xx kernel base [when I finally see manufacturers release their kernel sources]...So my question: Does mainline kernel need to be of similar build? Or can a 3.xx mainline be made to work [easily] with a 2.xx reference kernel?

Indeed, that will make things even easier to get Android working on their ARM platforms. The transition from Mainline to Android kernel will be simpler, since most of the Android patches would be already on the Mainline kernel.

Out of curiosity I also tend to check the kernel revision Android devices (habits, habits :)), and there are a few that use 2.x.x revisions. I think that is a decision that the vendor makes. I can only take a guess why, and - among others - I think it might be that it is more time-consuming for the vendor to forward port the device patches to a newer revision, than it is for them to back-port the Android patches to an older revision.

Now the answer to your question is: it depends on the size of your patches, the life cycle of the Reference kernel tree, and what parts of the kernel are affected by your patches.

If your patches applied to a 2.x.y Mainline kernel (that make it a 2.x.y Reference kernel), do not touch or do not depend on the things that have changed in the 3.x.x Mainline since version 2.x.x, then it shouldn't be very difficult. Git is your friend for this, since it can tell you when the two trees diverged in time, whether it would be possible to fast-forward the 2.x.y Reference kernel to 3.x.x Mainline, and give you the number of conflicts. Then again, it is guaranteed that a lot of things would have changed over that long period of time.
Hope this makes sense.

insink71 said:

> Thanks again for your time.Regards,Rob

No problem, anytime!

Regards,
Vassilis L.

akavassis

29 June 2012 - 10:10 AM

Dear Vassilis, really enjoyed reading your guide. I'd like to try and see if I can get Android 2.2 to work on Raspberry Pi so I will try following the guide and see what happens. Many thanks,Antonis K.

Manoj vivek

28 August 2012 - 05:01 PM

hello,, This was an really informative post and I got curious to install android on my device. But before that I woulf like to clarify "whether it will be able to install android on my nokia 5233"? Waiting on your reply to continue the install process.

Vasileios Laganakos

29 August 2012 - 11:51 PM

Hello Antonis,

> akavassis, on 29 June 2012 - 11:10 AM, said:
>
> > Dear Vassilis, really enjoyed reading your guide. I'd like to try and see if I can get Android 2.2 to work on Raspberry Pi so I will try following the guide and see what happens. Many thanks,Antonis K.

Thank you for your kind words :) How did that go for you, got Rasperry Pi running Android now?

Cheers,
Vassilis L.

Vasileios Laganakos

30 August 2012 - 01:05 AM

Hiya!

> Manoj vivek, on 28 August 2012 - 06:01 PM, said:
>
> hello,, This was an really informative post and I got curious to install android on my device. But before that I woulf like to clarify "whether it will be able to install android on my nokia 5233"? Waiting on your reply to continue the install process.

I'm glad you found it useful! I don't know about that specific device, but firstly you would need to have a working Linux kernel for it, and most importantly to be able to flash the kernel, the filesystem, and possibly change/update the bootloader on the device. Also, I don't know if the kernel supports talking to the framebuffer, and you would need that because I assume you would want Android graphics working. If not, I presume you would need the graphics drivers sources, its integration information, and get to adapt them to work and get built through and for Android; I don't know if all this is available though.

Cheers,
Vassilis L.

tanasis

11 September 2012 - 12:06 PM

Great Guide Vassilis! Nice insights about the porting procedure. :) Well I have been using Androind on my ARM board quite a while, but only the 2.3.3 version. The kernel I am using is pretty much customized for the setup. I want now to upgrade to the ICS, but without having to change the kernel. Do I have to change anything, regaring the U-BOOT or any other file in the boot partition? Or a new rootfs (ICS) is sufficient? Can you give me some directions on how to upgrade the system to ICS? Th

Vasileios Laganakos

13 November 2012 - 03:04 PM

Hello Tanasis,

> tanasis, on 11 September 2012 - 01:06 PM, said:
>
> Great Guide Vassilis! Nice insights about the porting procedure. :)

Cheers :)

> tanasis said:
>
> Well I have been using Androind on my ARM board quite a while, but only the 2.3.3 version. The kernel I am using is pretty much customized for the setup. I want now to upgrade to the ICS, but without having to change the kernel. Do I have to change anything, regaring the U-BOOT or any other file in the boot partition? Or a new rootfs (ICS) is sufficient? Can you give me some directions on how to upgrade the system to ICS? Thank you!

I think that there shouldn't be any need to change anything other than the root file system, at least to get things booting. Is the kernel you use a recent revision? The more recent the kernel the less the probability to miss something required for recent Android versions.

For the ICS file system, you can safely follow the directions described in the blog entry above to get it built. It will give you a generic ARM Android filesystem, without and graphics acceleration though.

Then deploy it in the same way you deployed it on your current ARM dev board. The kernel bootargs should point you to which partition the Android filesystem lives currently.

Make sure to compare the Android configuration used for your board (e.g. the init.rc) with the new ICS fs. Also, don't forget to check for kernel modules that the init.rc might be loading at boot time, and copy them across from the old Android fs to the new ICS (locations should be in init.rc or any other configuration file that is sourced). I guess these are the main things that you need to check for now. If you let me know how your system is configured I might be able to help more :)

Please let us know how did everything go!

Cheers,
Vassilis L.

Vasileios Laganakos

11 February 2013 - 04:54 PM

Hi all, We have updated our blog entry at http://www.linux-arm...#Jelly_Bean_MR1with info on how to build JellyBean MR1 for VersatileExpress. The procedure above remains the same, with a few more details in the procedure of deploying the built fs that are JB-MR1-specific and applicable in platforms other than VersatileExpress. Keep on porting! ;) Vassilis L.

Antonino

15 March 2013 - 02:55 PM

Hi Vasileios, I'm a computer engineer student, for my thesis work i'm trying to port Android on a Zedboard zynq 7000 with dual Cortex A9. i'm already able to been run on my board a version desktop of ubuntu with 3.3.0 kernel. Can I base my projet on your guide? I'm trying to following your fantastic guide but when try to enter this commands: git remote add localRepositoryhttps://android.goog...rnel/common.git I receive the following error message : fatal: Not a git repository (or any of the parent directories): .git do you know why??? i use https://android.googlesource.combecause android.git.kernel.org doesn't reach the contact. is it correct?? thank you very much

Vasileios Laganakos

26 March 2013 - 06:40 PM

Hello Antonino,

> Antonino, on 15 March 2013 - 02:55 PM, said:
>
> Hi Vasileios, I'm a computer engineer student, for my thesis work i'm trying to port Android on a Zedboard zynq 7000 with dual Cortex A9. i'm already able to been run on my board a version desktop of

ubuntu with 3.3.0 kernel. Can I base my projet on your guide? I'm trying to following your fantastic guide but when try to enter this commands: git remote add localRepository https://android.goog...rnel/common.git I receive the following error message : fatal: Not a git repository (or any of the parent directories): .git do you know why??? i use https://android.googlesource.combecause android.git.kernel.org doesn't reach the contact. is it correct?? thank you very much

You can surely base your project to this guide if it proves to be helpful to you! :-)

I just tried :

git remote add localrepo https://android.goog...rnel/common.git
git fetch

and everything worked fine. Maybe that was some temporary problem. So with :

git checkout <branch_name>

you should be able to clone the branch you wish.

Good luck with your project and do let us know how it goes!

Cheers,
Vassilis L.

Actions Like (1)

tootwo163@163.com Feb 20, 2014 2:15 AM

Dear Vasileios Laganakos，I need you a help for a question about TrustZone，and my email： majiangang9001@gmail.com.Thank you.

Actions Like (0)

Alban Rampon Feb 20, 2014 9:05 AM (in response to tootwo163@163.com)

Questions about TrustZone can be asked in the ARM Processors group. I have invited you to join that group so you can ask your question.

Actions Like (1)

ackmicro Aug 31, 2014 9:31 AM

Thank's Vasileios for this article & thank's Alban for all your comments on this article. pleased to read this

Actions Like (1)

Lori Kate Smith Aug 31, 2014 6:19 PM (in response to ackmicro)

Agreed - blog and comments are as useful now as they were in 2011. It was one of the first blogs on arm.com and one of the most read. Thanks for keeping the conversation going.

Actions Like (1)

martinkbrown Sep 25, 2014 5:45 AM

Hi all,

Is there a timeline for making Android KitKat 4.4 sources available?

Thanks!

Actions Like (0)

Matthew Du Puy Sep 25, 2014 9:30 PM (in response to martinkbrown)

The source, builds and specific version tags for AOSP are already here:

Codenames, Tags, and Build Numbers | Android Developers

You'll need the Repo to download the source:

Downloading the Source | Android Developers

Actions Like (1)