Android-x86 Project Marshmallow Porting

https://drive.google.com/open?id=1mND8K-AXbMMI8-wOTe75NOpM0xOcJbVy8UorryHOWsY

黃志偉
cwhuang@android-x86.org
2015/11/28
http://www.android-x86.org

Agenda

- Introduction: what, why, how?
- History and milestones
- Current status
- Porting procedure
- Develop android-x86
- Future plans

About Me

- A free software and open source amateur and promoter from Taiwan
  - CLDP / CLE
  - GNU Gatekeeper
  - Android-x86 Open Source Project
- https://zh.wikipedia.org/wiki/Cwhuang

Introduction

- What's Android-x86?
- Why needs Android-x86?
- How can we do it?

What's Android-x86 ?

- An open source project aimed to provide a complete solution for Android on x86 devices
- Android BSP (Board support Package) for x86 platform
- At first we use ASUS Eee PC and Virtualbox as the reference platform.
- Some vendors donate tablets, like Tegatech Tegav2, 4tiitoo AG WeTab and AMD

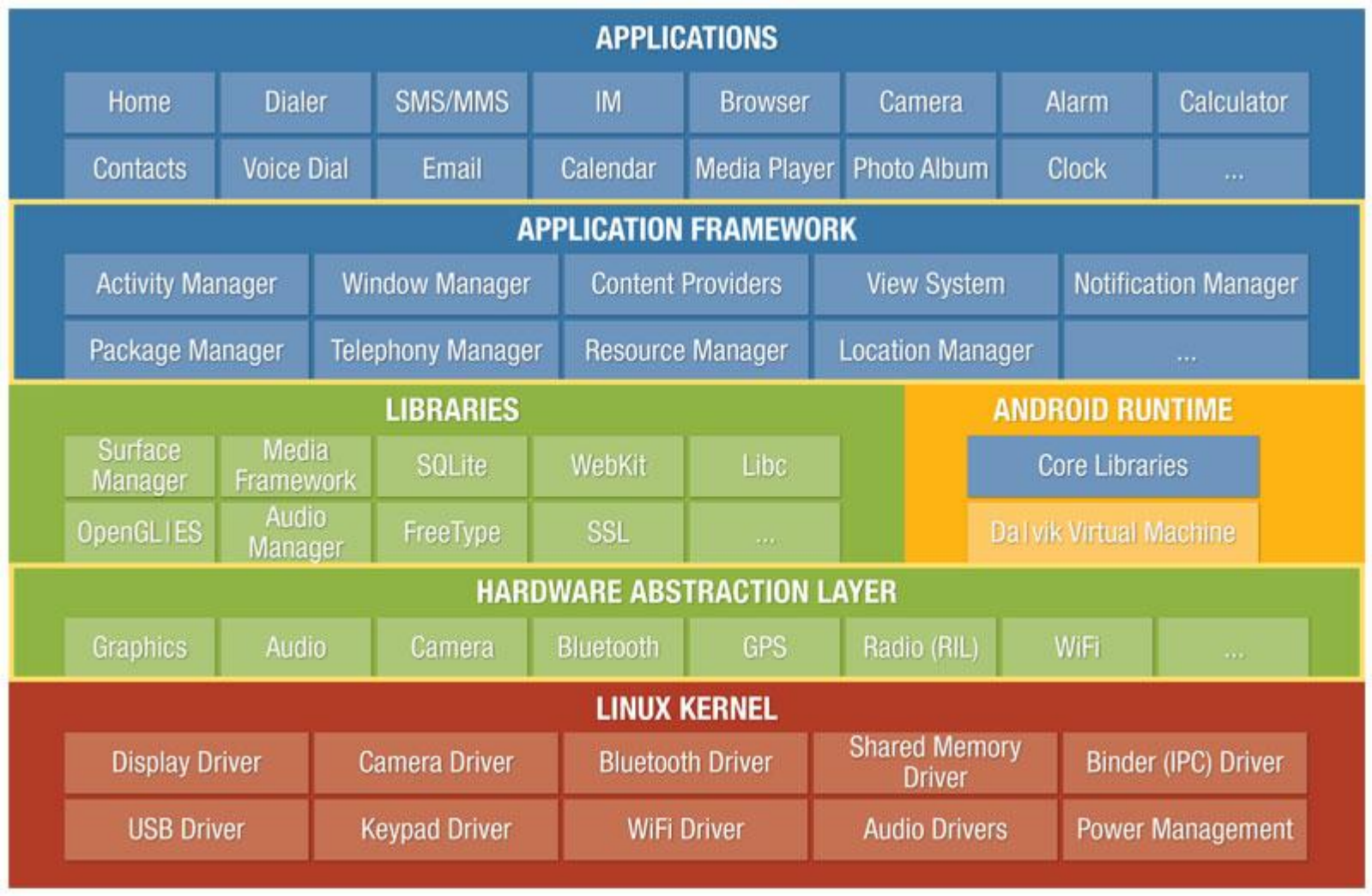
Why needs Android-x86?

- Android is an open source operating-system originally designed for **arm** platform
- It's open source, we can port it to other platforms, like mips, PowerPC and x86
- AOSP officially supports x86 now
  - AOSP doesn’t have specific hardware components
  - Still a lot of work to do to make it run on a real device

But what are the benefits?

- Understanding Android porting process
- The x86 platform is widely available
- A test platform much faster than SDK emulator
  - Android-x86 on vbox / vmware
  - Suitable for tablet apps

Android architecture



How to do that?

- ~~Toolchains – already in AOSP, but old...~~
- Kernel – vanilla almost works, except power management
  - Use newer kernel to support more hardwares
  - Patch kernel to support specific hardware we want
- Libraries – bugfixes or improvements
  - Building time – bionic, llvm, ~~ppp, v8~~, ...
  - Runtime – bluetooth, vold, ...
- HAL (hardware abstraction layer)
- Framework
- Apps

History (2009)

- Mar: Not maintained, need patches
- Apr: Patch hosting for android x86 support by Yi Sun <beyounn at gmail.com>
- May: Hard to maintain and sync with upstream
- Jun: Need a common, complete code base
- Jul: android-x86.org, release v0.9
- Aug: Build 20090820
- Sep: Build 20090916
- Oct: Build 20091024
- Nov: Android-x86 1.6 released
- Dec: Eclair 2.0 porting

History (2010)

- Jan: Build 20100115 (Android 2.0)
- Feb: Create eclair-x86 branch (Android 2.1)
- Mar: Android-x86 1.6-r2 released
- Apr: Eclair-x86 updated to 2.1-update1
- Jul: Working on froyo-x86
- Aug: A test iso for vm released
- Sep: target eeepc, i915 HAL, 3G support
- Oct: 3D hardware acceleration
- Nov: Improved vold, target viewpad10
- Dec: Installer read-write support, i965 support

History (2011)

- Jan: Android-x86 2.2 released
- Feb: Gingerbread-x86 branch (Android 2.3)
- Mar: Honeycomb-x86 porting (private)
- Jul: Android-x86 2.2-r2 released
- Aug: Android-x86 2.3-RC1 released
- Nov: Android-x86 3.2-RC2 released
- Dec: ICS-x86 branch (Android 4.0)

History (2012)

- Jan: Build 20120101
- Feb: Android-x86 4.0-RC1 released
- Apr: Update ics-x86 to 4.0.4
- Jul: Android-x86 4.0-RC2 released
- Nov: Jb-x86 porting based on Android 4.2
- Jul: Build 20121225 (Android 4.2)

History (2013)

- Feb: Build 20130228 (Android 4.2)
- Jun: Android-x86 4.0-r1 released
- Jul: Build 20130725 (Android 4.3)
- Nov: Kitkat-x86 porting based on Android 4.4

History (2014)

- Feb: Android-x86 4.4-RC1 released
- Apr: Add ntfs/exfat/ext4 support to vold
- May: Android-x86 4.4-RC2 released
- Aug: Android-x86 4.4-r1 released
- Nov: Lollipop-x86 porting based on Android 5.0
- Dec: Kernel 3.18 porting

History (2015)

- Jan: Android-x86 4.4-r2 released
- Mar: Lollipop-x86 updated to Android 5.1
- Apr: Kernel 4.0 porting
- Oct: Android-x86 5.1-rc1 released
- Oct: Marshmallow-x86 porting for Android 6.0
- Oct: Move git server to SourceForge
- Nov: Succeed to bring up Marshmallow-x86

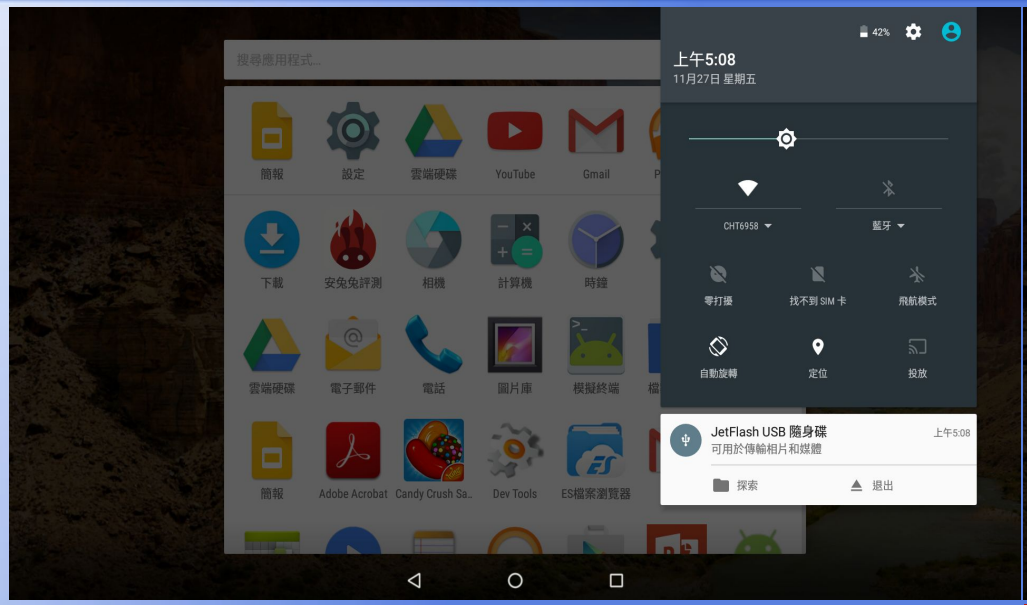
Feature of Marshmallow-x86

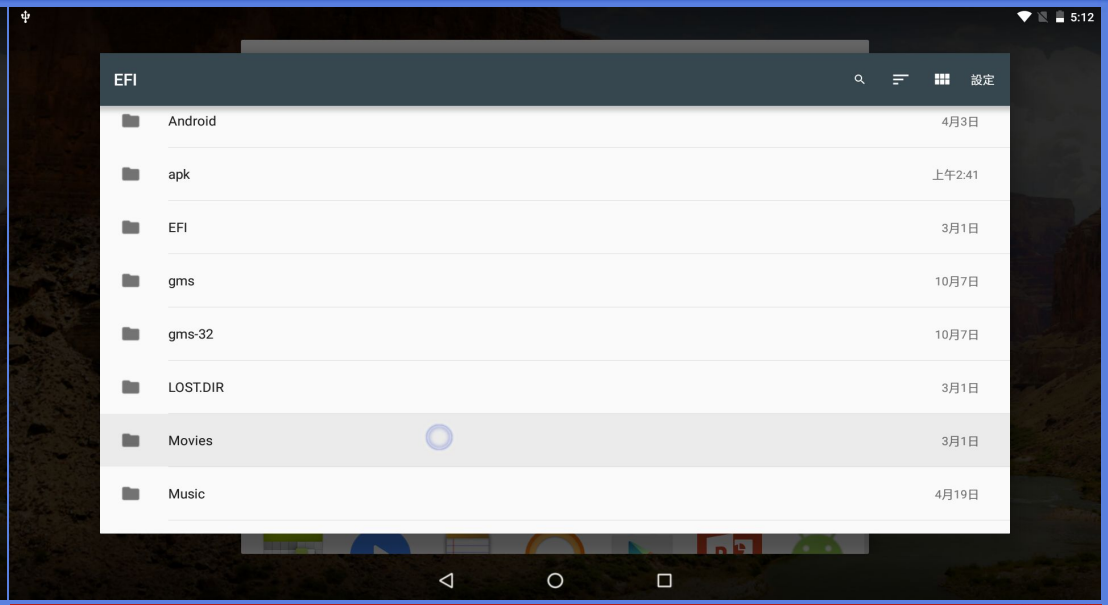
- Android 6.0.0_r26 (Marshmallow) based
- Support both 32-bit and 64-bit systems
- Kernel 4.0.9/4.x, KMS enabled
- Live CD & disk installer, support ext4/ext3/ext2/ntfs/fat32
- Provide hybrid iso image and EFI image
- Hardware 3D acceleration (intel/radeon/nvidia chipsets)
- Support Wifi, Ethernet, Bluetooth, Audio and Sensors
- Auto-detect drivers
- Touchscreen support, including calibration
- Compressed filesystem (squashfs)
- Support virtual machine like Virtualbox, Qemu and VMware
- Debug mode and tool (vc switching, busybox, mksh, …)

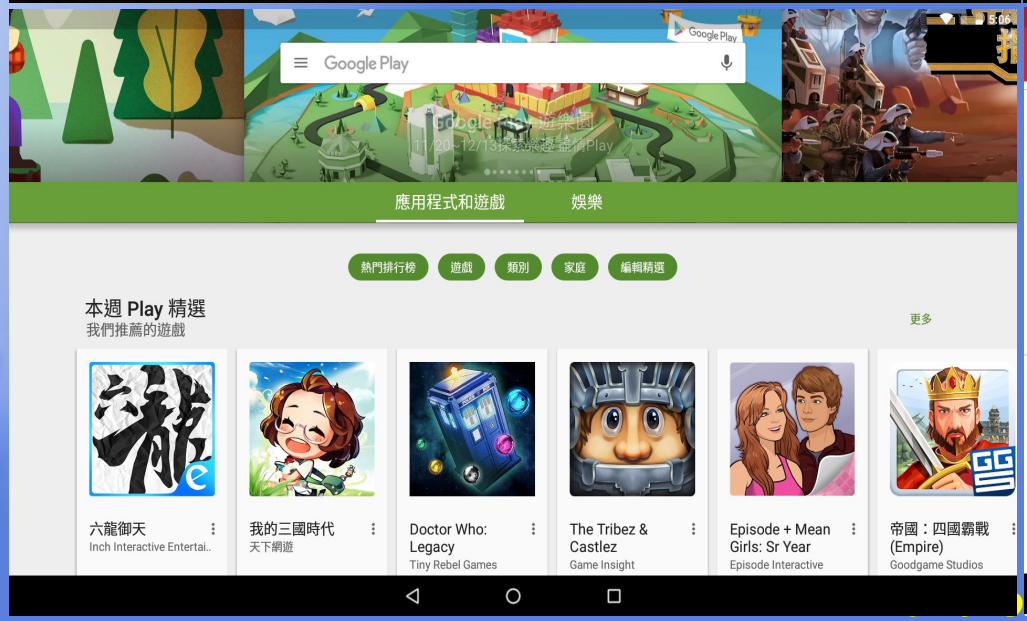
Porting status

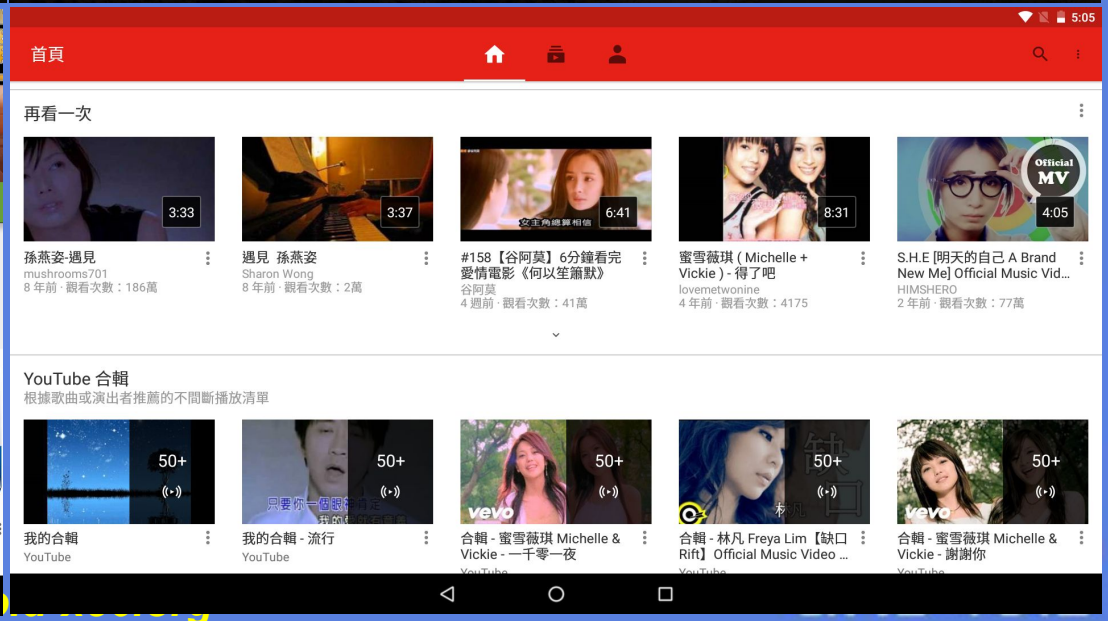
	Marshmallow-x86	Lollipop-x86	Kitkat-x86
Display	OK	OK	OK
HW Acceleration	Partial	OK	OK
Wifi	OK	OK	OK
Touchscreen	OK	OK	OK
Audio	OK	OK	OK
Bluetooth	Partial	OK	OK
Sensors	Partial	OK	OK
Camera	OK	OK	OK
Ethernet	OK	OK	OK
Backlight	OK	OK	OK
Suspend/Resume	Partial	OK	OK

Screenshots









Porting procedure (1/2)

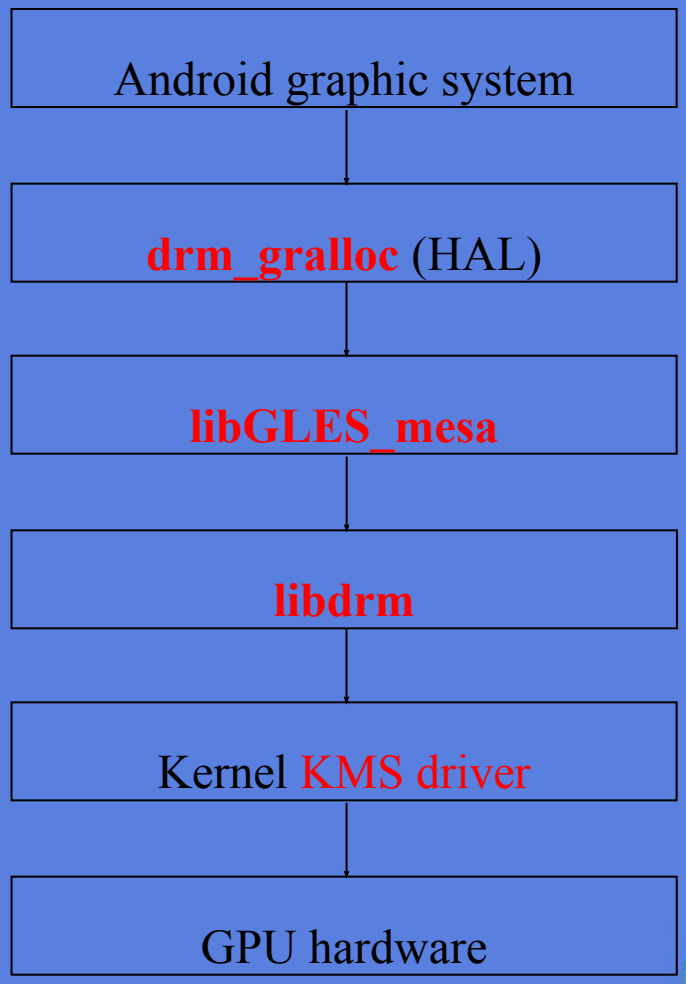
- Prepare the codebase by manifest.xml
  - AOSP git projects
  - Modified AOSP git projects
    - device/generic/x86
    - device/generic/common
    - frameworks/base
    - hardware/libhardware_legacy
    - system/core
    - ...
  - Added x86 specific git projects
    - bootable/newinstaller
    - external/libdrm
    - external/mesa
    - ...

Porting procedure (2/2)

- Apply x86 patches (git rebase) to AOSP
- Try to build and fix compiling errors
- Try to boot and fix problems
  - Display is the critical part
    - Removing the workaround 10194508 breaks Mesa
      - Solved by adding back the workaround
  - Log (logcat / dmesg) is the best friend

Mesa porting for HW acceletion

- Version 11.0.6
- ~ 60 fps
- Supported GPUs
  - Intel integrated graphic chip
  - AMD radeon/~~radeonsi~~
  - Nvidia (nouveau)
  - Vmware? (vmwgfx)
- Not support
  - Intel Atom Zxxx, E6xx (PowerVR)



Develop android-x86

- Get the source tree
- ~~Create a target (optional)~~
- Customize kernel (optional)
- Build the image
- Test on target device

Get source

- The git server
  - ~~git.androidx86.org~~
  - android-x86.git.sf.net

```
$ repo init -u git://gitscm.sf.net/gitroot/android-x86/manifest \
  -b marshmallow-x86
$ repo sync
```

Build an iso / efi image

- $ source build/envsetup.sh
- $ lunch android_x86-userdebug
- $ make -jN iso_img (efi_img)
  - out/target/product/$target/$target.iso

  {kernel,initrd.img,installer.img,ramdisk.img,system.sfs}
- Compress filesystem if mksquashfs is available

  (Disable it by USE_SQUASHFS=0)

Test the iso image

- Hybrid iso format
- Create a bootable USB
  - dd if=target.iso of=/dev/sdX
  - Unetbootin
  - Linux Live USB Creator (LiLi)
- Boot from the USB disk
  - Live mode
  - VESA mode
  - Debug
  - Installer

Unified image

- The image is designed to work on most devices as possible
- Why?
  - Android image is hardware specific
  - Originally we follow the design to create many targets, but...
    - confused the users
    - hard to maintain
    - a nightmare to make release

Make the image be generic

- Leverage Linux open source projects
  - Mesa / libdrm
  - Bluez
  - ffmpeg
- Auto detect and load drivers
  - initrd scripts
  - ueventd
- Determine the HAL to load at runtime
  - /etc/init.sh sets the properties ro. hardware.$hal_name

Limitations

- No optimization for a particular platform / CPU
- Unable to customize overlay at runtime
- No customization of kernel boot parameters

Future plans

- Supports GPU acceleration in VM
  - VMware - workable but buggy
  - Qemu - possible with the latest kernel & qemu
  - GPU passthrough - should work but untested
  - VirtualBox - ?
- Enable hardware video / audio codecs
- Improve the installation
- Better UX
  - CyanogenMod on Android-x86
  - Remix OS on Android-x86

Open source activities

- www.android-x86.org
- ~~code.google.com/p/android-x86~~
- sf.net/projects/android-x86
- twitter.com/android_x86
- IRC at irc.freenode.net #android-x86
- groups.google.com/group/android-x86
  - android-x86@googlegroups.com
  - Over 8700 subscribers
  - > 1000 emails/month

Call for developers

- Google group: Android-x86
  http://groups.google.com/group/android-x86

- IRC channel at irc.freenode.net
  #android-x86

Thank you!