A13 Android OLinuxino Jelly Bean

To compile Jelly Bean for the OLinuxino board, you need to broadly go through the following steps:

1. Setup host OS
2. Setup environment on host OS
3. Get the sources
4. Configure the sources (should you wish to change something)
5. Compile
6. Flash the image

Now I am going to step by step go through each step.

Host OS

Host OS setup is generally easy. You either start with a brand new machine, or a VM on which you are going to compile your Android. Minimum settings are:

- Quad core processor
- 32 GB RAM (swap + physical)
- 100 GB disk space
- Network card

Once you have assembled your machine (or created the VM), you need to install Ubuntu 12.04 LTS 64 bit on it. I used the minimal iso (as it installs quickly and does not add other stuff that will take resources and also boots quickly). We do not need a GUI on this machine. A CLI is enough. A good practice is to install SSH server on the machine and from then on ssh into the machine from your desktop.

Setup Environment on Host OS

Once you have installed the minimal iso, you need to follow the Android initializing build environment instructions with a few modifications. Do the following:

JAVA (Oracle)

To compile Jelly Bean, you need Oracle Java. This is no longer available via apt repositories. You need java version 1.6. I downloaded the package jdk-6u45-linux-x64.bin.

Install Sun Java 6:

chmod a+x jdk-6u45-linux-x64.bin

sudo /bin/bash jdk-6u45-linux-x64.bin

Then utilize the following commands to actually switch to Sun Java 6.

```
sudo update-alternatives --config java
```

```
sudo update-alternatives --config javac
```

If you are not on at least this version, then it is not going to compile Jelly Bean. (Instructions are same for ICS as well).

Other essentials

Before installing any other software, upgrade your installation to the latest.

```
sudo apt-get install aptitude wget

sudo aptitude update

sudo aptitude safe-upgrade
```

You need to install some other software as well.

```
sudo apt-get install git gnupg flex bison gperf build-essential \
  zip curl libc6-dev libncurses5-dev:i386 x11proto-core-dev \
  libx11-dev:i386 libreadline6-dev:i386 libgl1-mesa-glx:i386 \
  libgl1-mesa-dev g++-multilib mingw32 tofrodos \
  python-markdown libxml2-utils xsltproc zlib1g-dev:i386
```

The next command is optional.

```
sudo ln -s /usr/lib/i386-linux-gnu/mesa/libGL.so.1 /usr/lib/i386-linux-
gnu/libGL.so
```

You also need to configure USB access to actually be able to adb shell into the board.

Create a file called `51-android.rules` with the following content:

```
# adb protocol on passion (Nexus One)
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4e12",
MODE="0600", OWNER="<username>"
# fastboot protocol on passion (Nexus One)
SUBSYSTEM=="usb", ATTR{idVendor}=="0bb4", ATTR{idProduct}=="0fff",
MODE="0600", OWNER="<username>"
# adb protocol on crespo/crespo4g (Nexus S)
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4e22",
MODE="0600", OWNER="<username>"
# fastboot protocol on crespo/crespo4g (Nexus S)
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4e20",
MODE="0600", OWNER="<username>"
# adb protocol on stingray/wingray (Xoom)
SUBSYSTEM=="usb", ATTR{idVendor}=="22b8", ATTR{idProduct}=="70a9",
MODE="0600", OWNER="<username>"
# fastboot protocol on stingray/wingray (Xoom)
```

```
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="708c",
MODE="0600", OWNER="<username>"
# adb protocol on maguro/toro (Galaxy Nexus)
SUBSYSTEM=="usb", ATTR{idVendor}=="04e8", ATTR{idProduct}=="6860",
MODE="0600", OWNER="<username>"
# fastboot protocol on maguro/toro (Galaxy Nexus)
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4e30",
MODE="0600", OWNER="<username>"
# adb protocol on panda (PandaBoard)
SUBSYSTEM=="usb", ATTR{idVendor}=="0451", ATTR{idProduct}=="d101",
MODE="0600", OWNER="<username>"
# adb protocol on panda (PandaBoard ES)
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="d002",
MODE="0600", OWNER="<username>"
# fastboot protocol on panda (PandaBoard)
SUBSYSTEM=="usb", ATTR{idVendor}=="0451", ATTR{idProduct}=="d022",
MODE="0600", OWNER="<username>"
# usbboot protocol on panda (PandaBoard)
SUBSYSTEM=="usb", ATTR{idVendor}=="0451", ATTR{idProduct}=="d00f",
MODE="0600", OWNER="<username>"
# usbboot protocol on panda (PandaBoard ES)
SUBSYSTEM=="usb", ATTR{idVendor}=="0451", ATTR{idProduct}=="d010",
MODE="0600", OWNER="<username>"
# adb protocol on grouper/tilapia (Nexus 7)
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4e42",
MODE="0600", OWNER="<username>"
# fastboot protocol on grouper/tilapia (Nexus 7)
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4e40",
MODE="0600", OWNER="<username>"
# adb protocol on manta (Nexus 10)
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4ee2",
MODE="0600", OWNER="<username>"
# fastboot protocol on manta (Nexus 10)
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4ee0",
MODE="0600", OWNER="<username>"
```

Copy this file into `/etc/udev/rules.d/`

Essentially:

```
sudo cp 51-android.rules /etc/udev/rules.d/51-android.rules
```

Reboot. The process of setting up the host is now complete.

You may optionally also use ccache.

```
echo "export USE_CCACHE=1" >> ~/.bashrc

export CCACHE_DIR=~/ccache
```

SOURCES

Create a directory with the name jb.

```
mkdir -p ~/jb/sources/
```

Get the sources:

```
cd ~/jb/sources/

wget http://emicrotec.at/public/ANDROID/android4.1-v1.2.tar.gz

wget http://emicrotec.at/public/ANDROID/lichee-4.1-v1.2.tar.gz

wget http://emicrotec.at/public/ANDROID/A13.txt

ls
```

Unpack them.

```
cd ~/jb/

mkdir exdroid

cd exdroid

tar xpf ~/jb/sources/android4.1-v1.2.tar.gz

tar xpf ~/jb/sources/lichee-4.1-v1.2.tar.gz
```

In folder ~/jb/exdroid, you will now find two folders: android4.1 & lichee.

Configure

The directory android4.1 contains the android code that you might is wish to change. Configurations are in devices/softwinner/nuclear-evb/ directory.

lichee contains the kernel, where you might want to add other modules. To configure:

```
cd ~/jb/exdroid/lichee/linux-3.0

make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- menuconfig
```

Compile

Execute the following commands in order.

```
cd ~/jb/exdroid/android4.1

source build/envsetup.sh

cd ../lichee/

./build.sh -p a13_nuclear -k 3.0

lunch

#(choose nuclear_evb, option 10)

extract-bsp

make -i -j4

pack
```

The image file will now be at:

~/jb/exdroid/lichee/tools/pack/sun5i_android_a13-evb.img

Congratulations, you have successfully compiled Jelly Bean for A13-OLinuxino

Flash

Flash the image using Live Suit v1.09 for Windows. Prefer Windows 7 for this job. If you are stuck with Windows 8 (and can't revert to Windows 7), reboot Windows 8 into Test Mode. Otherwise, Live Suit will not install or work. The Linux Live Suit does not work and cannot flash the image. The Live Suit available from linux-sunxi is old and will not even recognize the image as correct. Download Live Suit from cubieboard site. Live Suit on Linux **will not** work.