Nvidia Tegra 4上安装Ubuntu 14.10

前一阵子在忙一款11.6寸平板，Nvidia Tegra 4主芯片，2G内存，大屏幕，全尺寸吸磁键盘，加上融合了Windows的桌面确实有点平板，笔记本二和一的感觉，即然是笔记本就不能单纯跑android这怎么简单了，项目差不多接近尾声的时候就在Nvidia Tegra 4运行了一把Ubuntu14.10，总体来说在tegra4上跑ubuntu14.10还算流畅，本打算如果跑不起来就求其次移植一个lubuntu在tegra4上，现在来看效果还算很好，完成没必要，只是时间比较仓促，手上只有一块平板，ubuntu14.10运行起来以后还要再烧回android版本继续调BUG，苦啊，所以在ubuntu上的触摸屏，声卡就没有实现，声卡驱动是正常的，触摸屏网上有相关的教程，有兴趣可以自己研究下，好，此处略去一万字，开始重点写一下移植ubuntu14.10的过程。

1，分区

NV是通过nvflash工具烧写系统静像，通过flash.cfg来对mmc分区，这里我们需要分BCT，PT，EBT，LIX，APP，DTB这几个分区：

BCT：这个分区是用来存放DDR参数，bct.cfg文件，大小8M
PT　：用来存放分区表flash.cfg文件，大小2M
EBT：放我们的bootloader.bin，大小8M
LIX　：用来放我们的内核，大小8M
APP：用来存放我们的Ubuntu系统，这里我们分配8G的空间
DTB：用来存放tegra114-macallan.dtb文件，大小4M

2，制件bootloader.bin及boot.img文件

boot.img包含我们的内核及ubuntu启动时需要的initrd文件，bootloader不解释。

Bootloader 一般不用修改，除非你需要制件双系统需要修改bootloader，制作双系统时需要flash.cfg中另外添加两个分区，一个我们boot.img及我们的ubuntu分区，在bootloader在载入内核时通过：
LoadKernel(RamBase, KERNEL_PARTITION, NULL, 0));
去指定可以引导我们ubuntu的boot.img分区即可。

boot.img配制内核部分：
（1）去把触摸屏的驱动
（2）支持LCD console，如果没有这个支持，屏幕是没有任何图像显示的，内核配制如下

```
@@ -1228,8 +1228,14 @@ CONFIG_FB_MXC_SYNC_PANEL=y
 # Console display driver support
 #
 # CONFIG_VGA_CONSOLE is not set
+# CONFIG_VGACON_SOFT_SCROLLBACK is not set
 CONFIG_DUMMY_CONSOLE=y
```

-# CONFIG_FRAMEBUFFER_CONSOLE is not set
+CONFIG_FRAMEBUFFER_CONSOLE=y
+# CONFIG_FRAMEBUFFER_CONSOLE_DETECT_PRIMARY is not set
+# CONFIG_FRAMEBUFFER_CONSOLE_ROTATION is not set
+# CONFIG_FONTS is not set
+CONFIG_FONT_8x8=y
+CONFIG_FONT_8x16=y

(3) 修改启动参数，支持serial console

console=ttyS0,115200n8 debug_uartport=lsport,3 console=tty1
console=ttyS0表示可能使用串口得到控制台，如果你的板子不支持OTG，这个选项就显得尤其重要。

OK内核配制这些基本就已经够了，make zImage编译内核，生成zImage
boot.img还缺少一个initrd文件，ubuntu启动时会找这个文件，但并不关心他的内容是什么，所以我们 # touch initrd
生成一个空的initrd文件即可.

最后一步生成boot.img，需要下载一个mkbootimg制件boot的工具，在编译的android输出目录下也可以找个这个工具，生成boot.img的命令：

mkbootimg --kernel ./zImage --ramdisk ./initrd –board mmcblk0p1 --output boot.img

3，制件ubuntu文件系统

分区，bootloader，boot都已经好了，下面就开始制件ubuntu文件系统。

(1) 下载ubuntu core rootfs

http://cdimage.ubuntu.com/ubuntu-core/releases/
上面的网站有ubuntu12.04.4到ubuntu 14.10的各种版本的ubuntu静像。
这里我下载的是arm ubuntu 14.10版本

http://cdimage.ubuntu.com/ubuntu-core/releases/14.10/release/ubuntu-core-14.10-core-armhf.tar.gz

解压：

```
mkdir ubuntu-14.10
cd ubuntu-14.10
sudo tar xpf ubuntu-core-14.10-core-armhf.tar.gz
```

(2)，删除root 密码

sudo vi etc/shadow #切记，etc前面不要有斜线，不然你将删除的是你本机的root密码

(3)，serial console 支持

前面内核启动时已经添加console=ttyS0，还需要创建/etc/init/ttyS0.conf文件，内容如下：
ttyS0 - getty
#
This service maintains a getty on ttyS0

description "Get a getty on ttyS0"

start on runlevel [2345]
stop on runlevel [016]

respawn

exec /sbin/getty -L 115200 ttyS0

如果你下载的是14.04或者14.10版本，你会发现即便这两个地方已经修改过还是无法通过串口登陆ubuntu，这是udev的原因，没有创建/dev/ttyS0节点，需要使udev自动去创建/dev/ttyS0节点才行，ls /dev/inpt同样你也会发现没有生成/dev/input下的字符节点，及/dev/fb0，这就意味着你将不能使用鼠标和键盘以及屏幕没有输出，这里解决办法可以从13.04的/sbin/udevd替换当前sbin目录下，并修改
/etc/init/udev.con文件，将/lib/systemd/systemd-udevd替换为/sbin/udevd
-exec /lib/systemd/systemd-udevd --daemon
+exec /sbin/udevd –daemon
这时候udev就会去自动创建设备节点了。

(4)，添加wifi驱动，我的板子上wifi使用的型号是bcm43341，找到bcm43341.ko cfg80211.ko两个驱动模块，放在/lib/modules目录下，修改/etc/rc.local文件使ubuntu启动时自动加载这两个模块
#!/bin/sh -e
#
rc.local
#
This script is executed at the end of each multiuser runlevel.
Make sure that the script will "exit 0" on success or any other
value on error.
#
In order to enable or disable this script just change the execution
bits.
#

```
# By default this script does nothing.
/sbin/insmod /lib/modules/cfg80211.ko
/sbin/insmod /lib/modules/bcm43341.ko
exit 0
```

将wifi所需要的固件放入下面目录 data/misc/wifi/firmware/
执行ifconfig -a 如果有wlan0说明wifi驱动已经正常加载.

(5)，移植WPA及iwlist

iwlist用来扫描网络，查看wifi是否可用，wpa用来连接网络。
网上下载wpa_supplicant源码，编译时加上static选项。得到下面这几个文件：
wpa_cli
wpa_passphrase
wpa_supplicant
wpa_supplicant.conf

将，wpa_cli，wpa_passphrase，wpa_supplicant这三个工具放入usr/local/sbin/目录
将，wpa_supplicant.conf放入/etc目录

基础的ubuntu系统配制修改已经差不多了，下面就是制件文件系统
这里我生成了一个8G文件系统，因为之前我们给ubuntu分了8G的空间，所以不要超过8G大小
生成文件系统的方法，主要通过下面几条命令，文档的最后会给出详细的制件文件系统的脚本。

```
#这里生成一个8G的system.img

dd if=/dev/zero of=system.img bs=512 count=$bcnt > /dev/null 2>&1;
#将system.img挂载成一个块设备
losetup /dev/loop0 system.img > /dev/null 2>&1;
#我在这里使用的是ext3格式
mkfs -t ${rootfs_type} /dev/loop0 > /dev/null 2>&1;
#将system.img当成一个块设备挂载
#将我们刚才修改的最基本的14.10 ubuntu系统拷贝到system.img
mount /dev/loop0 mnt > /dev/null 2>&1;
(cd /mnt; tar cf - *) | tar xf - ;
```

最后卸载system.img，这时候我们的ubuntu 14.10的静像就算生成了，

4，烧录静像到平板

system.img也有了，万事具备，只欠东风了，平板开机进入烧写模式通过下面命令烧写。

../bin/nvflash --bct bct.cfg --setbct --configfile flash.cfg --create --bl bootloader.bin --odmdata 0x80498000 –go

5，连接网络，安装ubuntu-desktop

(1)，烧录完成以后会平板自动开机，如果出现如下提示，表示ubuntu14.10已经可以运行了

```
localhost login:
Ubuntu 14.10 localhost.localdomain ttyS0

localhost login: root
Last login: Wed Dec 24 06:01:17 UTC 2014 on ttyS0
Welcome to Ubuntu 14.10 (GNU/Linux 3.4.66 armv7l)

 * Documentation:  https://help.ubuntu.com/
root@localhost:~# ls /dev/
```

(2)，配制wifi，将要连接的wifi网络和密码写入wpa_supplicant.conf

如：

```
root@localhost:~# wpa_passphrase  TP-LINK_2.4GHz_FAF2AE Qucii-699 >
/etc/wpa_supplicant.conf

root@localhost:~# cat /etc/wpa_supplicant.conf
network={
        ssid="TP-LINK_2.4GHz_FAF2AE"
        #psk="Qucii-699"
        psk=22bb534f9e5bec0465cbc91c9a13141c6f06f0e7bfd75bb0d0c2014768efa7d3
}
root@localhost:~#
```

(3)，连接wifi

执行：wpa_supplicant -B -i wlan0 -c /etc/wpa_supplicant.conf 连接无线网络

ping 8.8.8.8 #ping 可以ping通，连上网就可以发挥你的想像力了，想做什么就大胆的去尝试吧。

(4)，更新软件源

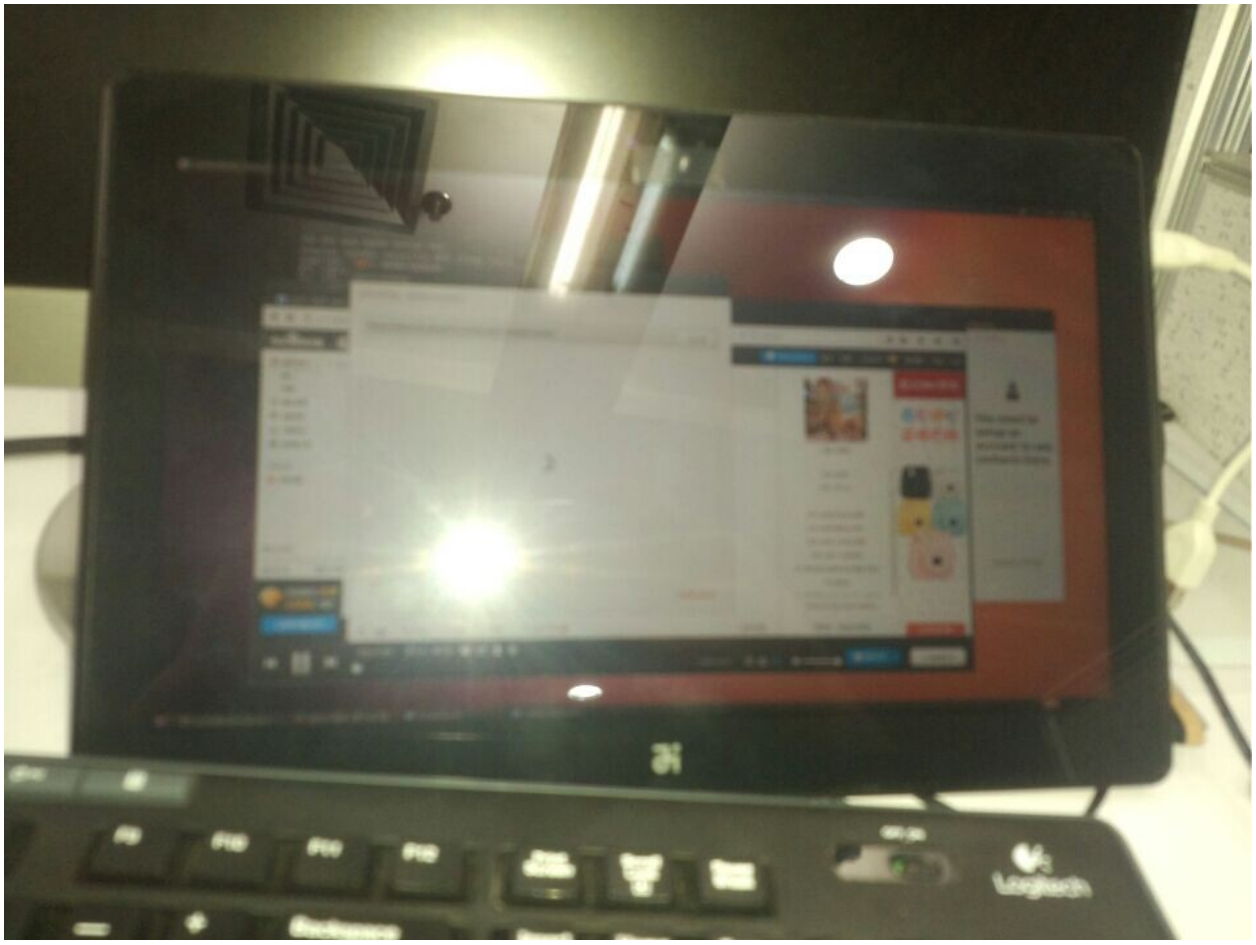
apt-get update

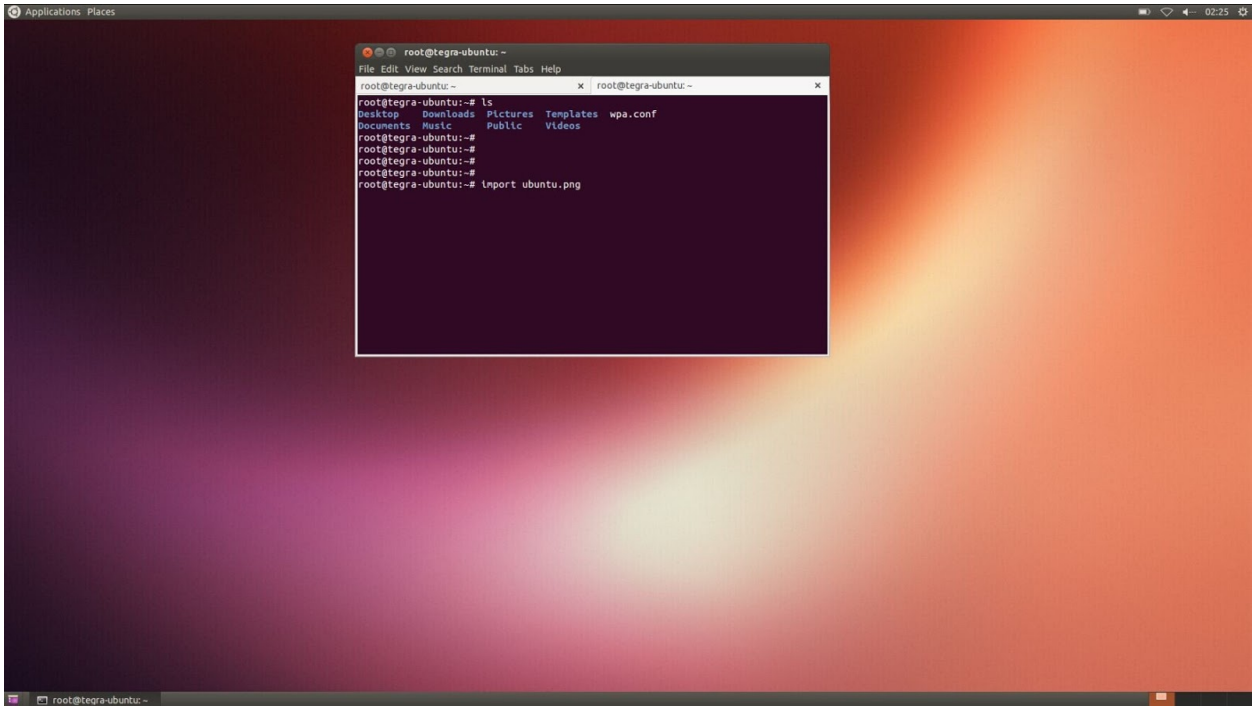
(5)，安装ubuntu-desktop

apt-get install ubuntu-desktop

reboot重启便会出现ubnuntu登陆窗口。

至此，ubuntu14.10已经安装完成，手机像素挺差的，不过不要太在意这些细节，来，上个图。





安装桌面时有时候你可能还需要这些软件

2 apt-get install x-window-system-core
3 apt-get install gnome-core
4 apt-get install metacity
5 apt-get install gnome-desktop-environment (optional)

安装完桌面以后剩下还有一个问题，就是备份刚刚安装的 ubuntu 14.10

可参考下面的脚本：

```
june@:~/ubuntu_img$ cat ~/Dropbox/codes/system_to_img/system_to_img.sh
#!/bin/bash
#########################################################################
# File Name: system_to_img.sh
# Author: chengdong
# mail: zchengdongdong@gmail.com
# Created Time: Wed 03 Dec 2014 02:39:39 PM CST
#########################################################################

image=./image/ubuntu.img

ssh root@172.0.0.116 'dd if=/dev/mmcblk0p1' | dd of=$image

e2fsck -yf $image

resize2fs -M $image
```

==========

最后把制件文件系统的脚本贴出来。

```
#!/bin/bash
rootfs_type="ext3"
#268435456,6442450944,8589934592
rootfs_size=8589934592
LDK_DIR=$(cd `dirname $0` && pwd)
rootfs=${ROOTFS-${LDK_DIR}/ubuntu-14.10};
echo "Making system.img... ";
umount /dev/loop0 > /dev/null 2>&1;
losetup -d /dev/loop0 > /dev/null 2>&1;
rm -f system.img;
if [ $? -ne 0 ]; then
	echo "clearing system.img failed.";
	popd > /dev/null 2>&1;
	exit 1;
fi;
bcnt=$(( ${rootfs_size} / 512 ));
bcntdiv=$(( ${rootfs_size} % 512 ));
if [ $bcnt -eq 0 -o $bcntdiv -ne 0 ]; then
	echo "Error: root file system size has to be 512 bytes allign.";
	popd > /dev/null 2>&1;
	exit 1;
fi
dd if=/dev/zero of=system.img bs=512 count=$bcnt > /dev/null 2>&1;
if [ $? -ne 0 ]; then
	echo "making system.img failed.";
	popd > /dev/null 2>&1;
	exit 1;
fi;
losetup /dev/loop0 system.img > /dev/null 2>&1;
if [ $? -ne 0 ]; then
	echo "mapping system.img to loop device failed.";
	popd > /dev/null 2>&1;
	exit 1;
fi;
mkfs -t ${rootfs_type} /dev/loop0 > /dev/null 2>&1;
if [ $? -ne 0 ]; then
	echo "formating filesystem on system.img failed.";
	popd > /dev/null 2>&1;
```

```
        exit 1;
fi;
sync;
rm -rf mnt;
if [ $? -ne 0 ]; then
        echo "clearing mount point failed.";
        popd > /dev/null 2>&1;
        exit 1;
fi;
mkdir -p mnt;
if [ $? -ne 0 ]; then
        echo "making mount point failed.";
        popd > /dev/null 2>&1;
        exit 1;
fi;
mount /dev/loop0 mnt > /dev/null 2>&1;
if [ $? -ne 0 ]; then
        echo "mounting system.img failed.";
        popd > /dev/null 2>&1;
        exit 1;
fi;
pushd mnt > /dev/null 2>&1;
echo -n -e "\tpopulating rootfs from ${rootfs}... ";
(cd ${rootfs}; tar cf - *) | tar xf - ;
if [ $? -ne 0 ]; then
        echo "failed.";
        popd > /dev/null 2>&1;
        popd > /dev/null 2>&1;
        exit 1;
fi;
echo "done.";

popd > /dev/null 2>&1;
echo -e -n "\tSync'ing... ";
sync; sync;     # Paranoid.
echo "done.";
umount /dev/loop0 > /dev/null 2>&1;
losetup -d /dev/loop0 > /dev/null 2>&1;
rmdir mnt > /dev/null 2>&1;
mv system.img images/
echo "System.img built successfully. ";
```