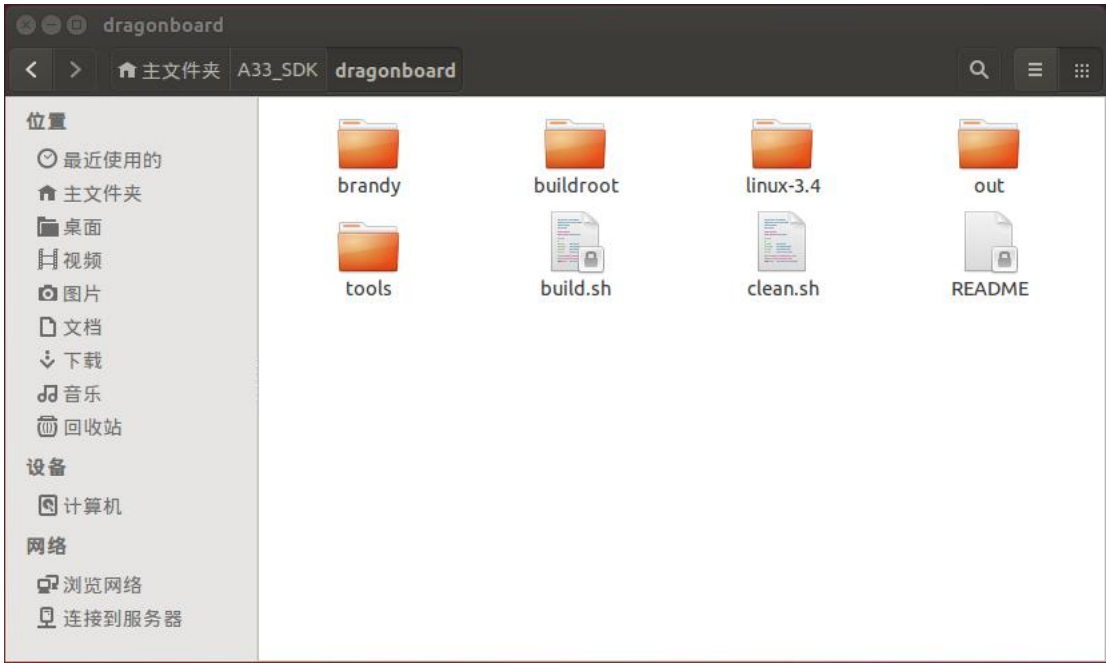
A33-Vstar

Linux SDK 编译及烧写方式说明

Linux_SDK 包名：A33_vstar_linux.tar.bz2
工程名：vstar
平台名：dragonboard
系统配置文件路径：
tools/pack/chips/sun8iw5p1/configs/vstar/sys_config.fex

解压 linuxSDK 源码包，目录结构如下



编译 Linux

首次编译，需要导入 vstar 方案的内核配置，进入 linux3.4 目录，执行：

```
$ cp a33_vstar_defconfig .config
```

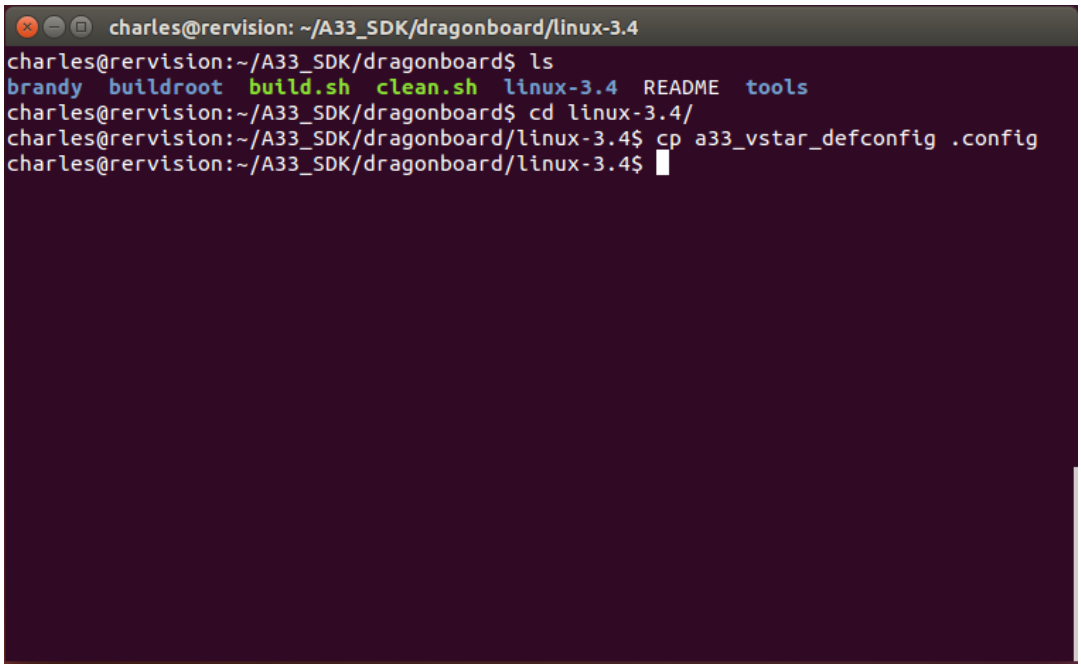
退回到 lichee 目录，再执行

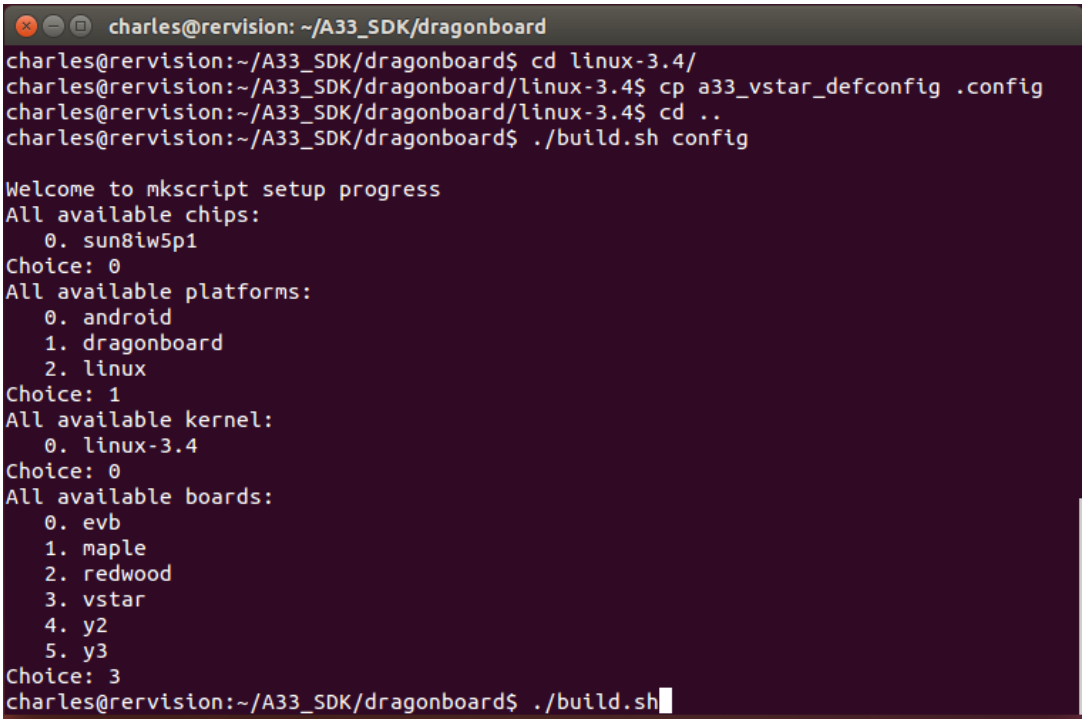
```
$ ./build.sh config
```

依次选择：0 1 3
（仅首次编译需要 config，以后就不需要了）

```
$ ./build.sh
```

即完成一次编译


charles@rervision: ~/A33_SDK/dragonboard/linux-3.4
charles@rervision:~/A33_SDK/dragonboard$ ls
brandy buildroot build.sh clean.sh linux-3.4 README tools
charles@rervision:~/A33_SDK/dragonboard$ cd linux-3.4/
charles@rervision:~/A33_SDK/dragonboard/linux-3.4$ cp a33_vstar_defconfig .config
charles@rervision:~/A33_SDK/dragonboard/linux-3.4$


charles@rervision: ~/A33_SDK/dragonboard
charles@rervision:~/A33_SDK/dragonboard$ cd linux-3.4/
charles@rervision:~/A33_SDK/dragonboard/linux-3.4$ cp a33_vstar_defconfig .config
charles@rervision:~/A33_SDK/dragonboard/linux-3.4$ cd ..
charles@rervision:~/A33_SDK/dragonboard$ ./build.sh config
Welcome to mkscript setup progress
All available chips:
0. sun8iw5p1
Choice: 0
All available platforms:
0. android
1. dragonboard
2. linux
Choice: 1
All available kernel:
0. linux-3.4
Choice: 0
All available boards:
0. evb
1. maple
2. redwood
3. vstar
4. y2
5. y3
Choice: 3
charles@rervision:~/A33_SDK/dragonboard$ ./build.sh

```
charles@rervision: ~/A33_SDK/dragonboard
   2. linux
Choice: 1
All available kernel:
   0. linux-3.4
Choice: 0
All available boards:
   0. evb
   1. maple
   2. redwood
   3. vstar
   4. y2
   5. y3
Choice: 3
charles@rervision:~/A33_SDK/dragonboard$ ./build.sh
INFO: ----------------------------------------
INFO: build lichee ...
INFO: chip: sun8iw5p1
INFO: platform: dragonboard
INFO: kernel: linux-3.4
INFO: board: vstar
INFO: output: out/sun8iw5p1/dragonboard/vstar
INFO: ----------------------------------------
INFO: build buildroot ...
installing external toolchain
please wait for a few minutes ...
```

编译完成：

```
charles@rervision: ~/A33_SDK/dragonboard
make[2]:正在离开目录 `/home/charles/A33_SDK/dragonboard/buildroot/target/dragonboard
/src/testcases/rtk_hciattach'
make[1]:正在离开目录 `/home/charles/A33_SDK/dragonboard/buildroot/target/dragonboard
/src/testcases'
generating rootfs...
blocks: 149M -> 360M
Creating filesystem with parameters:
    Size: 377487360
    Block size: 4096
    Blocks per group: 32768
    Inodes per group: 7680
    Inode size: 256
    Journal blocks: 1440
    Label:
    Blocks: 92160
    Block groups: 3
    Reserved block group size: 23
Created filesystem with 1256/23040 inodes and 40920/92160 blocks
e2fsck 1.42.9 (4-Feb-2014)
success in generating rootfs
Build at: 2016年 11月 07日 星期一 18:03:34 CST
INFO: build rootfs OK.
INFO: ----------------------------------------
INFO: build lichee OK.
INFO: ----------------------------------------
charles@rervision:~/A33_SDK/dragonboard$
```

如要清除编译，执行：

```
$ ./clean.sh
```

打包固件：

```
$ ./build.sh pack
```

```
charles@rervision: ~/A33_SDK/dragonboard
    Label:
    Blocks: 92160
    Block groups: 3
    Reserved block group size: 23
Created filesystem with 1256/23040 inodes and 40920/92160 blocks
e2fsck 1.42.9 (4-Feb-2014)
success in generating rootfs
Build at: 2016年 11月 07日 星期一 18:03:34 CST
INFO: build rootfs OK.
INFO: ----------------------------------------
INFO: build lichee OK.
INFO: ----------------------------------------
charles@rervision:~/A33_SDK/dragonboard$ ./build.sh pack
INFO: packing firmware ...
copying tools file
copying configs file
copying boot resource
copying boot file
support dragonboard test emmc when boot from card0
[mmc2_para]-sdc_used: 0 -> 1
packing for dragonboard
/home/charles/A33_SDK/dragonboard/tools/pack/pctools/linux/eDragonEx/
/home/charles/A33_SDK/dragonboard/tools/pack/out
Begin Parse sys_partion.fex
Add partion boot-resource.fex BOOT-RESOURCE_FEX
Add partion very boot-resource.fex BOOT-RESOURCE_FEX
```

```
charles@rervision: ~/A33_SDK/dragonboard
u-boot.fex Len: 0xbc000
fes1.fex Len: 0x1fc0
usbtool.fex Len: 0x20600
aultools.fex Len: 0x25d92
aultls32.fex Len: 0x2295a
cardtool.fex Len: 0x14000
cardscript.fex Len: 0x6ea
sunxi_mbr.fex Len: 0x10000
dlinfo.fex Len: 0x4000
arisc.fex Len: 0x2ed88
boot-resource.fex Len: 0x4c3400
Vboot-resource.fex Len: 0x4
env.fex Len: 0x20000
Venv.fex Len: 0x4
boot.fex Len: 0xc37800
Vboot.fex Len: 0x4
rootfs.fex Len: 0x16800000
Vrootfs.fex Len: 0x4
BuildImg 0
Dragon execute image.cfg SUCCESS !
----------image is at----------

/home/charles/A33_SDK/dragonboard/tools/pack/sun8iw5p1_dragonboard_vstar.img

pack finish
charles@rervision:~/A33_SDK/dragonboard$
```

固件位置： tools/pack/sun8iw5p1_dragonboard_vstar.img

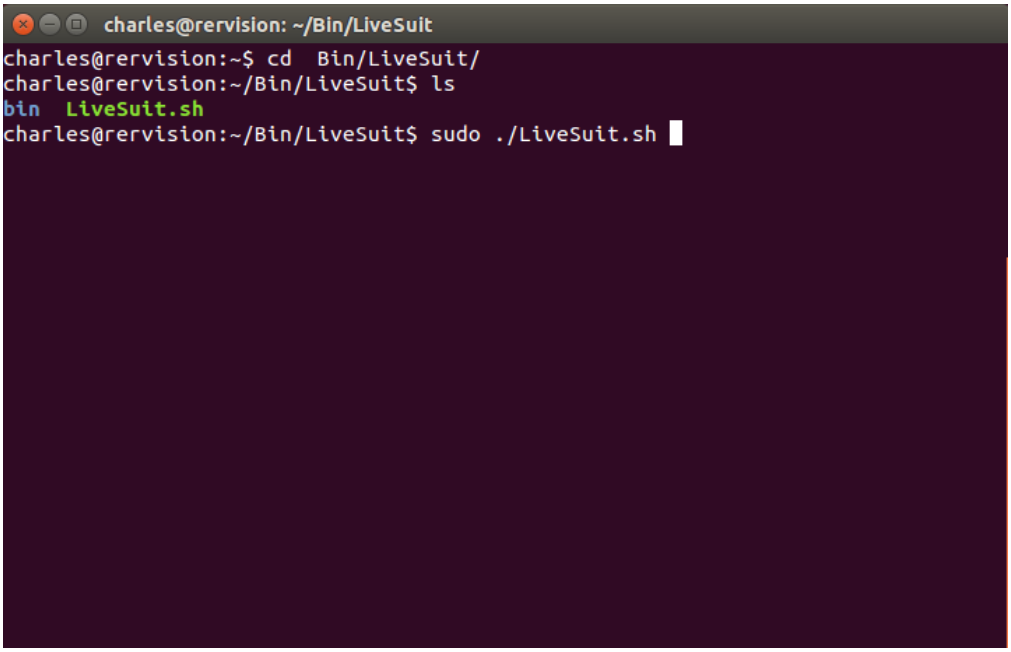
在 Linux 系统烧写固件

按照 LiveSuit_For_Linux 安装包中的教程安装 LiveSuit，根据本机的系统位数选择安装 32 位或 64 位版本。

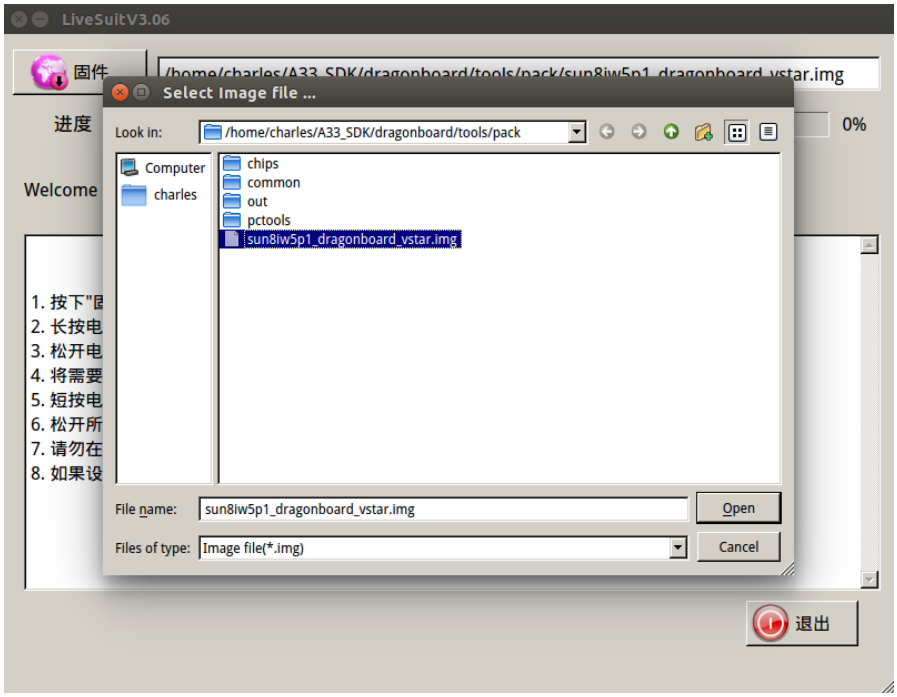
可执行程序路径：~/Bin/LiveSuit/LiveSuit.sh

运行 LiveSuit

```
$ sudo ./LiveSuit.sh
```



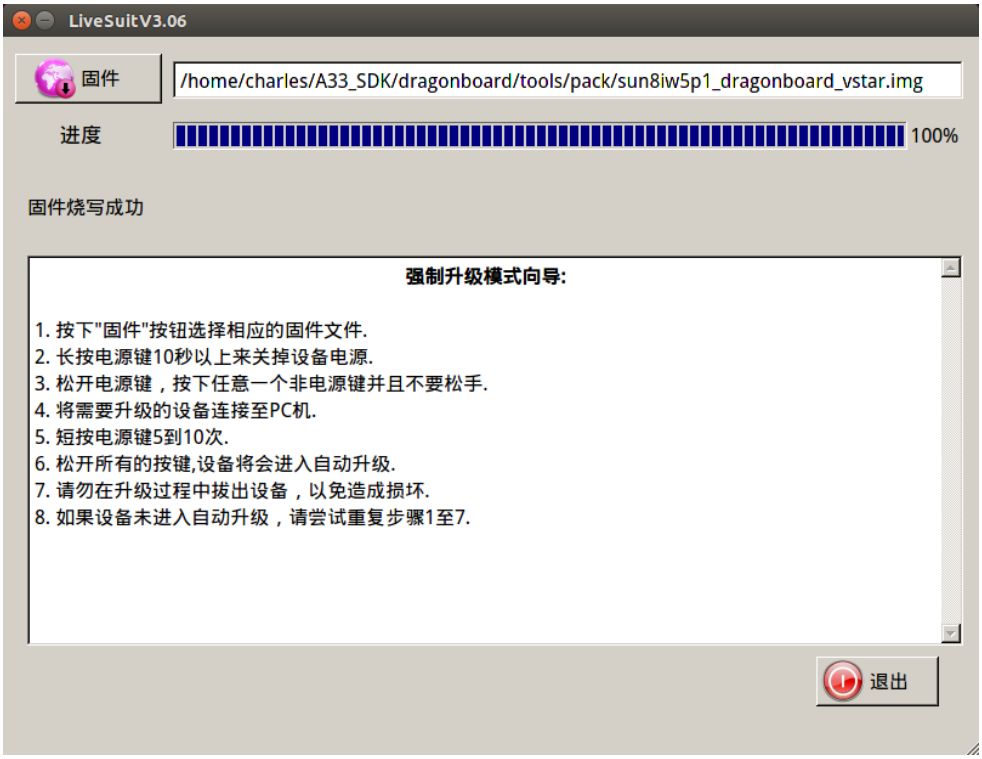
选择打包好的固件



给开发板断电，按住板上方的任意音量按键，将 USB 线连接设备的 MicroUSB 口至 PC 机，**短按电源（POWER）键大概 10 次左右**，会检测到设备（此时可以松开音量键），弹出窗口提示“是否格式化数据分区”：选 Yes，进行格式化烧写，清除 data 目录；选 No，进行非格式化烧写，不清除 data 目录

如果没有检测到设备，请重新尝试上述步骤

烧写的过程需要 3 分钟左右



在 Windows 系统烧写固件

安装 PhonixSuit 软件，打开后选择“一键刷机”，点击“浏览”选择要烧写的固件文件，选中“格式化”进行格式化烧写；如果不想擦除 data 分区，就取消“格式化”

给开发板断电，按住板上方的任意音量按键，将 USB 线连接设备的 MicroUSB 口至 PC 机，**短按电源（POWER）键大概 10 次左右**，会检测到设备（此时可以松开音量键），提示开始烧写固件，如果没有检测到设备，请重新尝试上述步骤

固件烧写成功后，拔掉 USB 线，再给开发板上电

修改 Linux 内核配置

```
$ cd linux-3.4
$ make menuconfig ARCH=arm
```

默认的内核配置文件
linux3.4/vstar_dragonboard_config

还原默认配置

```
$ cp a33_vstar_defconfig .config
```

交叉编译器配置

交叉编译器路径
out/dragonboard/common/buildroot/external-toolchain/bin

将编译器加入环境变量，用于编译 Linux 应用程序
$export PATH=$PATH:/home/rervision/A33_SDK/dragonboard/out/dragonboard/common/buildroot/external-toolchain/bin

导入交叉编译器的环境后，直接使用 arm-linux-gnueabi-gcc 编译应用程序

文件系统定制：

文件系统路径：
buildroot/target/dragonboard/rootfs

编译脚本：
buildroot/target/dragonboard/build.sh
在编译时调用此脚本，主要作用是复制文件到 rootfs 中，或在 rootfs 创建目录或文件，或建议文件连接，修改权限等。

启动脚本：
dragonboard/rootfs/autorun.sh
原路径是 dragonboard/extra/autorun.sh，编译时由脚本拷到 rootfs 中
把要加载驱动模块的命令和需要自启动的程序写在这里

驱动模块路径：
/system/vendor/modules
原路径是 rootfs/lib/modules/3.4.39，运行时由脚本拷到/system/vendor/modules

编译应用程序：

单独编译：
导入交叉编译器的环境后，直接使用 arm-linux-gnueabi-gcc 编译
将生成的程序传到板上即可运行，或手动拷贝到 rootfs 的 bin 目录，编译打包

在工程中编译：
参考 buildroot/target/dragonboard/src/testcase/example

将要编译的软件包（以 myapp 为例）放到 testcase 中，
修改 testcase/Makefile，加入
make -C myapp

修改 myapp 中的 Makefile

```
SRC_ROOT := ../..
LDFLAGS := -LSCRIPT
INCLUDE $(SRC_ROOT)/RULE.MK
MYAPP_OBJS = MYAPP.O

.PHONY: ALL
ALL: MYAPP
MYAPP: $(MYAPP_OBJS)
	$(LINK_MSG)
	$(LINKX)

$(MYAPP_OBJS): %.O: %.C
	$(COMPILE_MSG)
	$(COMPILEX)
```

在编译系统时，会把 myapp 编译到 rootfs/dragonboard/bin