全 志 科 技
Allwinner Technology

R16

lichee 使用手册

文档履历

版本号	日期	制/修订人	制/修订记录
V1.0	2015-02-28		初始版本

目录

1. 简介

本文档用于介绍 R16 芯片的 Linux BSP 的目录结构，固件定制和 Lichee 定制。

2. 目录结构

```
├─── brandy
├─── buildroot
├─── build.sh
├─── linux-3.4
├─── README
└─── tools
```

2.1. brandy

存放 boot0 和 u-boot 源码，其目录结构为

```
├─── basic_loader
├─── build.sh
├─── extern-libs
├─── gcc-linaro
├─── pack
├─── pack_tools
├─── u-boot-2011.09
└─── u-boot-2013.01
```

baisc_loader：boot0 源码，编译器使用 arm-cc，搭配 cygwin。boot0 代码体积必须控制在 24K 以内。编译命令：

```
$ cd basic_loader
$ make -f make_a67
```

生成 boot0_nand_sun8iw5p1.bin， boot0_sdcard_sun8iw5p1.bin， fes1_sun8iw5p1.bin 。其中 boot0_nand_sun8iw5p1.bin 是 Nand 的 boot0， boot0_sdcard_sun8iw5p1.bin 是 eMMC 的 boot0，fes1_sun8iw5p1.bin 是烧写引导程序。

gcc-linaro：u-boot 交叉编译工具链。

u-boot-2011.09：u-boot 源码，包括启动引导、量产烧写的代码。编译器使用 gcc-linaro，编译命令：

```
$ cd u-boot-2011.09
$ make distclean
$ make sun8iw5p1
$ make -j
```

生成的 u-boot-sun8iw5p1.bin 会自动拷贝到 lichee/tools/pack/chips/sun8iw5p1/bin 目录下。

2.2. buildroot

buildroot 的主要作用是

- 管理编译脚本和交叉编译工具链
- 定制开发 DragonBoard 测试用例
- 制作 Linux 固件的根文件系统，
  可以包含 strace, directfb, oprofile 等非常丰富的应用软件和测试软件。

目录结构如下

```
├─── board
├─── boot
├─── CHANGES
├─── Config.in
├─── configs
├─── COPYING
├─── dl
├─── docs
├─── external-packages
├─── fs
├─── linux
├─── Makefile
├─── package
├─── README
├─── scripts
├─── target
└─── toolchain
```

scripts：Lichee 编译脚本，主要包含 mkcmd.sh，mkcommon.sh，mkrule 和 mksetup.sh。
target/dragonboard：dragonboard 定制开发根目录。

2.3. linux-3.4

Linux 内核源码目录，结构如下

```
├─── android
├─── arch
├─── arisc_sun9iw1p1.bin
├─── block
├─── COPYING
├─── CREDITS
├─── crypto
├─── Documentation
├─── drivers
```

```
├── e_build.sh
├── firmware
├── fs
├── include
├── init
├── ipc
├── Kbuild
├── Kconfig
├── kernel
├── lib
├── MAINTAINERS
├── Makefile
├── mm
├── modules
├── net
├── output
├── README
├── REPORTING-BUGS
├── rootfs.cpio.gz
├── samples
├── scripts
├── security
├── sound
├── tinyandroid.tar.gz
├── tools
├── usr
└── virt
```

以上目录结构跟标准的 Linux 内核一致，除了 modules 目录。modules 目录是我们用来存放没有跟内核的 menuconfig 集成的外部模块的地方。我们目前放了 aw_schw,mali，rogue_km 和 nand 这 4 个外部模块，其中 aw_schw 是密钥模块：存储 key，rogue_km 和 mali 是 GPU 驱动，nand 是 nand 驱动。

2.4. tools

目录结构如下

```
├── daily_build
├── doc
├── pack
└── tools_win
```

该目录存放方案系统配置、打包脚本和工具，以及部分平台相关的工具。

用途	位置
方案配置	pack/common/

	pack/chips/sun8iw5p1/configs/
打包脚本和工具	pack/pack
	pack/pctools/
平台相关的工具	tools_win/

3. 编译系统

3.1. 使用说明

参考《R16_Android SDK Quick Start Guide》。

3.2. 二次开发

Lichee 编译脚本目前支持编译 buildroot 和 linux 内核，主要包括以下文件，

build.sh

buildroot/scripts/mkcmd.sh

buildroot/scripts/mkcommon.sh

buildroot/scripts/mkrule

buildroot/scripts/mksetup.sh

tools/pack/pack

3.2.1. 指定配置文件

当开发新的平台或者方案时，需要指定 buildroot 和内核的配置文件，修改 buildroot/scripts/mkrule 文件，文件格式如下，

```
<芯片编号>_<系统平台>    <buildroot 配置文件>   <内核配置文件>
```

或者：

```
<芯片编号>_<系统平台>_<方案>          <buildroot 配置文件>   <内核配置文件>
```

芯片编号：例如 sun9iw1p1

系统平台：例如 android

方案：例如 p1

buildroot 配置文件：不需要用 xxx 表示

内核配置文件：例如 sun9iw1p1smp_android_defconfig

将新的芯片平台以上信息汇成一行添加到文件中即可。

3.2.2. 添加系统平台

目前默认支持 3 个系统平台，分别是 android，dragonboard，linux。如需添加新的平台，修改 buildroot/scripts/mkcmd.sh 文件的 platforms 数组，例如添加 firefox 平台：

```
platforms=(
```

```
"android"
"dragonboard"
"linux"
"firefox"
)
```

3.2.3. 打包脚本说明

打包时需要拷贝若干文件到 tools/pack/out 目前，目前脚本对其进行了分类，分别是 tools_file_list, configs_file_list，boot_resource_list 和 boot_file_list，新增文件可以归入其中一类或者创建新类。

目前打包脚本分为 4 个阶段，分别是 do_prepare，do_common，do_pack_<platform>和 do_finish。

do_prepare：完成文件拷贝和预处理动作。

do_common：完成所有系统平台通用的文件解析，分区打包。

do_pack_<platform>：完成当前系统平台特有的工作。

do_finish：完成打包。

4. Lichee 定制

本章节主要介绍如何定制 Linux 固件根文件系统。

4.1. 根文件系统定制

Linux 固件根文件系统由 buildroot 制作，编译生成的文件和程序位于

```
out/sun8iw5p1/linux/common/buildroot/
```

目录结构如下

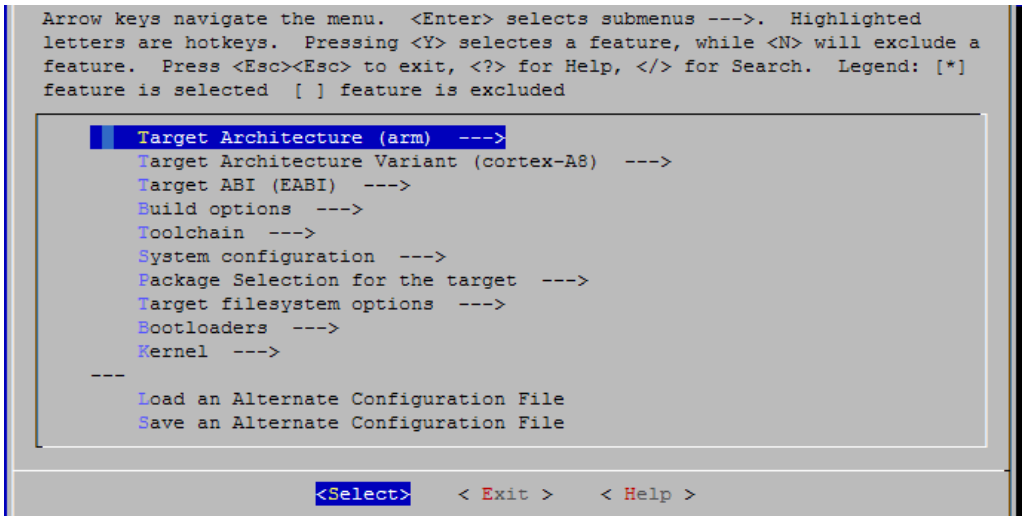
```
├── build
├── external-toolchain
├── host
├── images
├── Makefile
├── staging -> host/usr/arm-unknown-linux-gnueabi/sysroot
├── stamps
├── target
├── toolchain
└── toolchainfile.cmake
```

target 目录即 rootfs 的内容。

添加应用软件步骤。

1. $ cd out/sun8iw5p1/linux/common/buildroot/
2. $ make menuconfig

上面命令执行完会显示以下界面，



3. 根据需要配置应用软件
4. 退出并保存
5. 备份 config

   $ cp out/sun8iw5p1/linux/common/buildroot/.config buildroot/configs/sun8i_defconfig

   可以参照 3.2 添加新的配置。

4.2. 集成软件包

4.2.1. 源代码包

对于用户态的应用程序、动态库和静态库应该集成到 buildroot 中，在 buildroot/packages 下面 1 个目录对应一个包。关于如何在 buildroot 中集成软件包的说明，请参考 http://buildroot.uclibc.org/docs.html。

举一个简单的例子：

要在 buildroot 下添加一个源码包，首先要在 buildroot/package 目录下新建一个目录，目录名称为软件包的名称，目录中，再在目录中添加一个 config.in 文件和一个 xxxx.mk 文件（xxxx 为软件包的名称）。这 2 个文件的具体写法，参见 buildroot/package 目录下的其他的软件包，或者官方网站（软件源码包分为网上的官方软件包和自己编写的源码包，这 2 类包的 config.in 文件形式是一致的，但是.mk 文件的书写会有较大区别，假如是后者，请参见 fsck-msdos 包中的.mk，前者请参见 argus 包中的.mk）。做完以上操作以后，还需要在 buildroot/package 目录下的 config.in 文件中添加

```
source "package/panlong/Config.in"
```

注意：假设要添加的软件包的名称为 ***panlong*** *的话。至于段代码添加的位置由具体情况而定，添加位置影响执行* ***make menuconfig*** *是软件包对应选项的位置。*

示例：

```
menu "Package Selection for the target"

source "package/busybox/Config.in"
source "package/customize/Config.in"
```

```
#source "package/lcd-test/Config.in"
#source "package/tp-test/Config.in"
#source "package/kernel-header/Config.in"
#source "package/sw-tools/Config.in"
#source "package/ext4-utils/Config.in"
#source "package/tiobench/Config.in"
#source "package/fsck_msdos/Config.in"
#source "package/mali-3d/Config.in"
#source "package/cedar/Config.in"
source "package/panlong/Config.in"
# Audio and video applications
source "package/multimedia/Config.in"
```

这里“#”开头的行在执行 make menuconfig 时是看不到的。这里，我们将 source "package/panlong/Config.in"添加到了 menu "Package Selection for the target"菜单下，所以在我们执行 make menuconfig 后

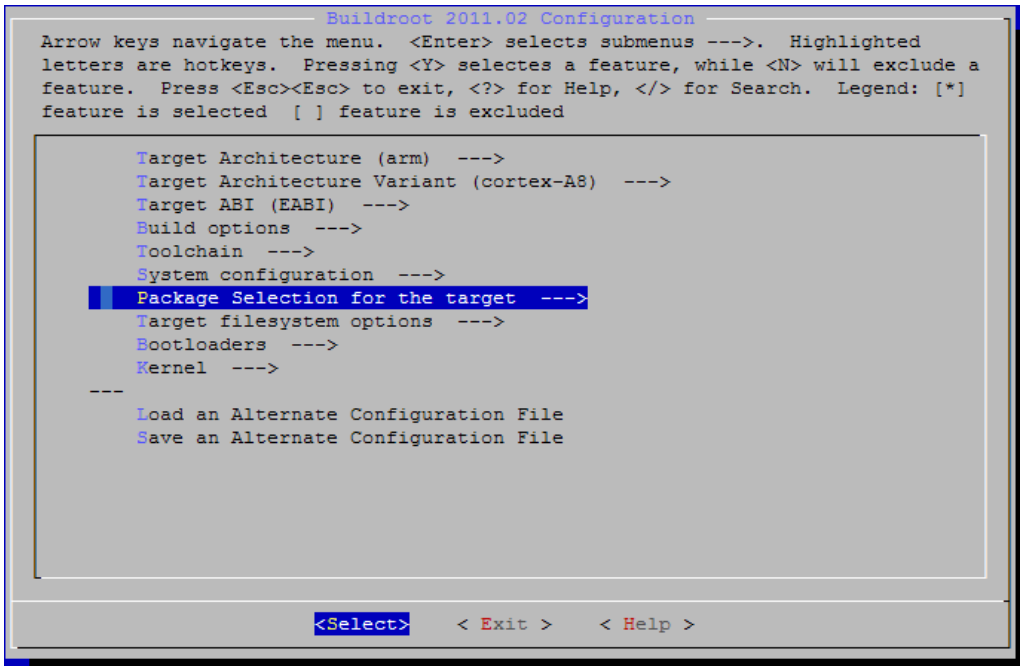


图 5.4 Buildroot make menuconfig 界面

做如图的选择，按下 enter 建

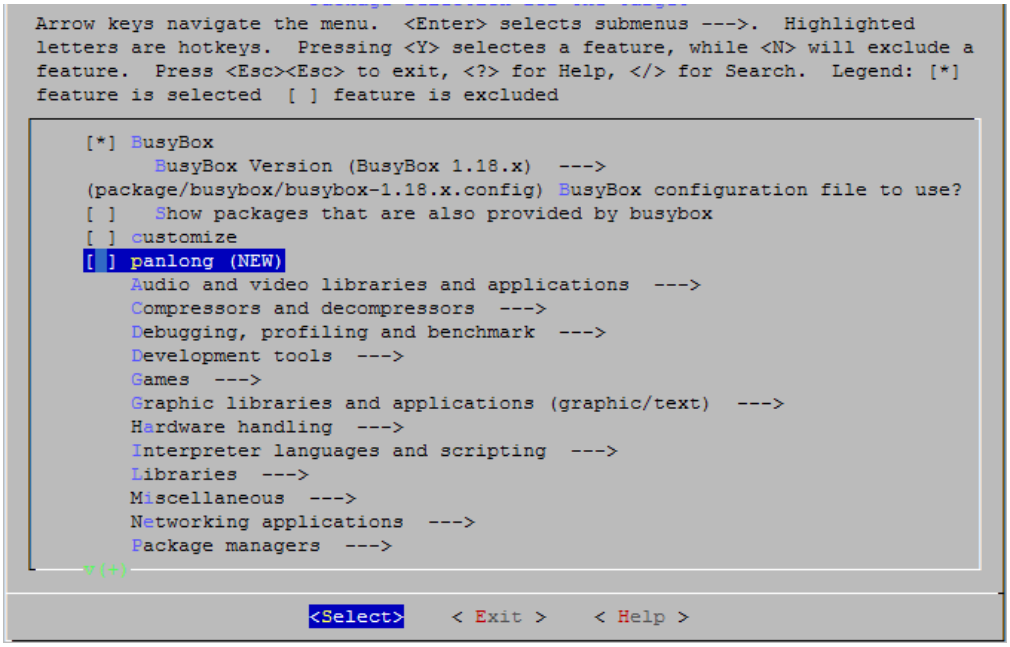


图 5.5 package selection for the target 子菜单界面

就可以看到我们添加的软件包了。

注意：***以上只是演示，实际添加时尽可能添加到子菜单中，以便于软件包的管理。***

对于内核驱动，应该尽量考虑放到 linux-3.4/drivers 下面，如果无法直接跟 kernel 的 menuconfig 集成，则应该放在 linux-3.4/modules 下面。

可以和 menuconfig 集成的软件包，添加方法参见 kconfig 相关资料。

无法与 menuconfig 集成的软件包，用 modules 下的 mali 来进行添加举例：

首先，在 modules 目录下建立 mali 包的子目录，然后为这个包编辑一个总的 makefile，

这里可能会用到 4 个参数：

LICHEE_KDIR：就是 buildroot 和 linux-3.4 所在的那一层目录 LICHEE_MOD_DIR==${LICHEE_KDIR}/output/lib/modules/${KERNEL_VERSION}KERNEL_VERSION= 3.4 CROSS_COMPILE= arm-linux-gnueabi- ARCH=arm

这些参数的定义都在 linux-3.4/scripts/build.sh 中定义。

完成 makefile 的编辑后，为了让系统整体编译时让其被编译进去，还需在 linux-3.4/scripts/build.sh 文件的 build_modules()函数中添加对 nand，wifi，eurasia_km gpu 软件包的编译规则，以及在 clean_modules()函数中添加清除规则。（具体写法可以仿照 nand）

假如添加的项目是默认打开的，那么就需要用编辑好的.config 文件替换掉对应的 defconfig。如 sun9i 的，我们就可以把 buildroot 下的.config 重命名为 sun9i_defconfig，然后保存到 buildroot\configs 文件夹下。

4.2.2. 二进制包

同上，只是忽略掉编译过程。

4.2.3. 可执行文件

直接添加到 lichee\out\linux\common\buildroot\output\target 中（前提是已经完全编译过一次），指令直接添加到 bin、sbin 或者 usr 下的 bin、sbin 中，其他可执行文件可以添加在希望指定的任意文件夹下。

5. 固件定制

固件主要包括方案配置文件、启动引导文件、烧写引导文件和分区镜像等。其中分区镜像由方案的 sys_partition.fex 文件配置，其余由 image.cfg 文件配置。两个文件的默认存放路径如下：

	sys_partition.fex 文件	image.cfg 文件
Android	tools/pack/chips/sun8iw5p1/xxx/	tools/pack/chips/sun8iw5p1/default/image.cfg
DragonBoard	tools/pack/chips/sun8iw5p1/default/	tools/pack/chips/sun8iw5p1/default/image.cfg
Linux	tools/pack/chips/sun8iw5p1/default/	tools/pack/common/imagecfg/ image_linux.cfg

如需使用自定义的 sys_partition.fex 和 image.cfg 文件，只需在方案的目录下添加文件名相同的文件即可。

下面主要介绍分区属性和如何规划分区。

5.1. 分区属性

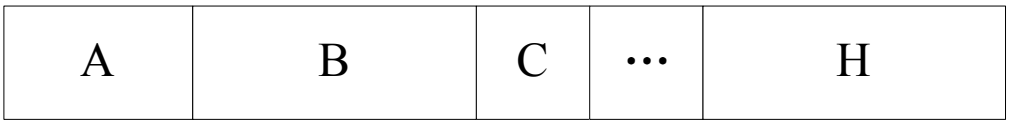
分区定义示例：

```
[partition]
    name             = system
    size             = 1572864
    downloadfile = "system.fex"
    user_type        = 0x8000
```

属性	描述
name	分区名称，最大 12 个字符
size	分区大小，单位是扇区（512Bytes）；Nand 方案为了安全和效率考虑，分区大小最好保证为 16M 字节的整数倍。
downloadfile	分区镜像文件
keydata	标志是否为私有数据分区，1 表示是，0 表示否
encrypt	采用加密方式烧录，将提供数据加密，但损失烧录速度，1 表示采用默认加密方式，? 表示私有用法
verify	量产完成后校验是否正确，1 表示校验，0 表示不做校验

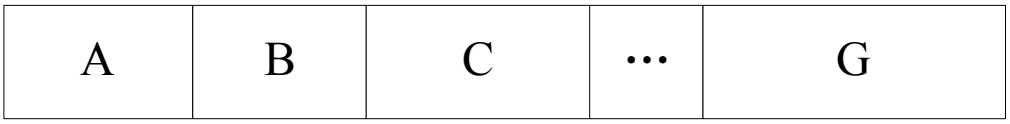
5.2. 规划分区

分区，是指存储设备(通常是 Nand 或者 eMMC)上，根据逻辑关系划分的空间。习惯上，分区的编号从 0 开始，代表第一个分区，1 代表第二个分区，以此类推。这项规则类似于 PC 上的硬盘分区，如下图所示。



上图存储设备上一共有 A-H 共 8 个分区，其中，分区 A 的起始位置从存储设备的头部开始，是第一个分区，分区 H 占用了尾部，是最后一个分区。

规划分区，是指在固件包中指明存储设备上的分区个数，并由用户自己定义分区属性。当烧写固件包后，存储设备上就会存在这样由用户定义的分区。用户可以通过下图的方式规划分区。



通过上图可以看出，B 分区的容量减小，C 分区容量增大，同时增加了 G 分区。

在 Lichee 中，最后一个分区的大小无需指定，是存储设备的容量减去前面分区大小的总和，该分区通常称为 UDISK 分区，在 Android 中用作 sdcard 分区（单用户）或者 data 分区（多用户）。因此，添加的分区必须在 UDISK 分区前面。例如，下面是 Android 系统（单用户）标准的分区划分，

分区名称	大小	说明
bootloader	16M	存放启动 logo 等资源文件
env	16M	存放 u-boot 引导参数
boot	16M	存放 kernel 和 ramdisk
system	768M	
data	2G	
misc	16M	
recovery	32M	
cache	640M	
metadata	16M	存放全盘加密 KEY
private	16M	存放 SN 号
UDISK		sdcard 分区

为了使用固件修改工具的克隆功能，需要添加一个 databk 目录，大小为 256M，添加后新的分区划分如下,

分区名称	大小	说明
bootloader	16M	存放启动 logo 等资源文件
env	16M	存放 u-boot 引导参数
boot	16M	存放 kernel 和 ramdisk
system	768M	
data	2G	
misc	16M	
recovery	32M	
cache	640M	
metadata	16M	存放全盘加密 KEY

private	16M	存放 SN 号
databk	256M	
UDISK		sdcard 分区

具体操作是修改方案目录下的 sys_partition.fex 文件，在 UDISK 分区前加入

```
[partition]
    name            = databk
    size            = 524288
    ro              = 0
    user_type       = 0x8000
```

6. Declaration