

T7 项目

Lichee 使用说明书 V1.0

目录

T7 项目.....	1
Lichee 使用说明书 V1.0.....	1
1. 目录.....	2
2. 前言.....	3
2.1. 编写目的.....	3
2.2. 使用范围.....	3
2.3. 阅读对象.....	3
3. 目录结构.....	4
3.1. Bootloader.....	4
3.2. Brandy.....	4
3.3. buildroot.....	5
3.4. linux-3.10.....	6
3.5. tools.....	7
4. 编译系统.....	8
4.1. 二次开发.....	8
4.2. 编译方式.....	8
4.2.1. 编译方式: <code>./build.sh config</code> 命令.....	8
5. Lichee 定制.....	10
5.1. 根文件系统定制.....	10
5.2. 集成软件包.....	11
5.2.1. 源代码包.....	11
5.2.2. 二进制包.....	13
5.2.3. 可执行文件.....	13
6. Declaration.....	14

1. 前言

1.1. 编写目的

本文档用于介绍全志科技 T7 芯片的 Linux BSP(Lichee 为开发代号)的结构、内部机制以及简单用法。该文档的目的用于指导用户如何定制和使用该 SDK。Linux BSP 可以联系全志科技的销售人员获取。

1.2. 使用范围

Allwinner T7 平台。

1.3. 阅读对象

T7 平台负责人、版本集成人员、SDK 开发人员。

2. 目录结构

```

├── bootloader
├── brandy
├── buildroot
├── build.sh
├── doc
├── linux-3.10
├── README
└── tools

```

2.1. Bootloader

存放 boot0、sboot 和 fes 源码，其目录结构为

```

├── tools
├── uboot_2011_sunxi_spl
└── uboot_2014_sunxi_spl

```

包括启动引导、量产烧写的代码。用于烧录的 fes1，用于启动的 boot0，需要在 u-boot-2014.07 中编译。其中 fes1 和 boot0 代码生成的 bin 文件体积必须控制在 24K 以内。

2.2. Brandy

存放 u-boot 源码，其目录结构为

```

├── add_hash
├── add_hash.sh
├── arm-trusted-firmware-1.0
├── armv8_toolchain
├── basic_loader
├── b.sh
├── build.sh
├── extern-libs
├── gcc-linaro
├── pack_tools
├── scripts
├── SUNXI_README
├── toolchain
├── u-boot-2011.09
└── u-boot-2014.07

```

brandy 包含用于烧写和启动的 uboot 源码。编译命令(没有修改该部分源码,就不需要编译 brandy):

```

$ cd brandy
方法 1: 编译 uboot
$ ./build.sh -p sun8iw17p1(生成 uboot)

```

方法 2: 单独编译 boot 各部分 bin 文件

```
$ cd brandy/u-boot-2014.07
```

```
$ make distclean --清理临时文件
```

```
$ make sun8iw17p1_config --配置 sun8iw17p1 平台, 只需要执行一次, 如果执行了 make distclean, 则需要再运行一次
```

```
$ make -j8 --生成 uboot
```

```
$ make boot0 --生成 boot0
```

```
$ make fes --生成 fes
```

```
$ make sboot --生成 sboot
```

以上编译命令要在配置平台后才能运行, 生成的 bin 文件会自动拷贝到相应的目录。

gcc-linaro: u-boot 和 boot0 交叉编译工具链。

toolchain: arm-trusted-firmware 交叉编译工具链

u-boot-2014.07: 只包括 u-boot 源码。boot0、sboot 和 fes 源码在 lichee/bootloader。

pack tools: 打包时使用的打包工具的源码

2.3. 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
|— build.sh
|— CHANGES
|— Config.in
|— configs
|— COPYING
|— dl
|— docs
|— external-packages
|— fs
|— linux
|— Makefile
|— output
|— 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。

2.4. linux-3.10

Linux 内核源码目录，结构如下

```
|— android
|— arch
|— block
|— build.sh
|— COPYING
|— CREDITS
|— crypto
|— Documentation
|— drivers
|— firmware
|— fs
|— include
|— init
|— ipc
|— Kbuild
|— Kconfig
|— kernel
|— lib
|— linaro
|— MAINTAINERS
|— Makefile
|— mm
|— modules
|— Module.symvers
|— net
|— output
|— README
|— REPORTING-BUGS
|— rootfs_32bit_arm9.cpio.gz
|— rootfs_32bit.cpio.gz
|— rootfs.cpio.gz
|— samples
|— scripts
|— security
|— sound
|— 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 gpu 和 nand 这 2 个外部模块，gpu 目录存放的是 GPU 驱动，nand 是 nand 驱动。

2.5. tools

目录结构如下

—	build
—	codecheck
—	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/sun8iw17p1/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. 二次开发

增加新的内核配置的步骤如下:

步骤 1: 在 lichee/linux-3.10/arch/arm/configs/目录下新增内核配置文件, 如下

```
lichee/linux-3.10/arch/arm/configs/
sun8iw17p1smp_android_defconfig
sun8iw17p1smp_defconfig
sun8iw17p1smp_min_defconfig
sun8iw17p1smp_standby_min_defconfig
```

步骤 2: 在 lichee/buildroot/scripts/mkrule 文件中新增

```
buildroot/scripts/mkrule
sun8iw17p1_android          sun8i_defconfig          sun8iw17p1smp_android_defconfig
sun8iw17p1_linux           sun8i_defconfig          sun8iw17p1smp_defconfig
```

chip_platform[_business] 制作 buildroot 配置 配置文件名称, 对应内核 (步骤 1), 增加的文件名称的格式进行命名, [_business] 非必须, chip 与 platform 是必须与芯片平台保持一致, 如 sun8iw17p1_android 中, sun8iw17p1 表示芯片平台, Android 表示项目平台。

例如:

```
sun8iw17p1_android          sun8i_defconfig          sun8iw17p1smp_android_defconfig
表示编译 sun8iw17p1 芯片的 Android 项目时, 内核将会使用配置文件: sun8iw17p1smp_android_defconfig
```

3.2. 编译方式

3.2.1. 编译方式: ./build.sh config 命令

```
lichee$ ./build.sh config
Welcome to mkscrip setup progress
All available platforms:
  0. android (Android: 项目平台)
  1. dragonboard
  2. linux
```

3. camdroid

Choice: 0

All available chips:

0. sun3iw1p1

1. sun5iw1p1

2. sun5iw2p1

3. sun5iw3p1

4. sun5iw6p1

5. sun8iw10p1

6. sun8iw11p1

7. sun8iw12p1

8. sun8iw15p1

9. sun8iw17p1 (sun8iw17p1: 芯片项目)

10. sun8iw1p1

11. sun8iw3p1

12. sun8iw5p1

13. sun8iw6p1

14. sun8iw7p1

15. sun8iw8p1

16. sun8iw9p1

17. sun9iw1p1

Choice: 9

All available kernel:

0. linux-3.10

Choice: 0

All available boards:

0. fpga

1. t7-p1 (t7-p1: 开发板)

2. t7-perf1

3. t7-perf2

4. t7-perf4

5. t7-ver

Choice: 1

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/sun8iw17p1/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/sun8iw17p1/linux/common/buildroot/
2. \$ make menuconfig

上面命令执行完会显示以下界面，

```
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted
letters are hotkeys. Pressing <Y> selects a feature, while <N> will exclude a
feature. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*]
feature is selected [ ] feature is excluded

Target Architecture (arm) --->
Target Architecture Variant (cortex-A8) --->
Target ABI (EABI) --->
Build options --->
Toolchain --->
System configuration --->
Package Selection for the target --->
Target filesystem options --->
Bootloaders --->
Kernel --->
---
Load an Alternate Configuration File
Save an Alternate Configuration File

<Select> < Exit > Help >
```

3. 根据需要配置应用软件
4. 退出并保存

5. 备份 config

```
$ cp out/sun8iw17p1/linux/common/buildroot/.config buildroot/configs/sun8i_defconfig
```

可以参照 3.1 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、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

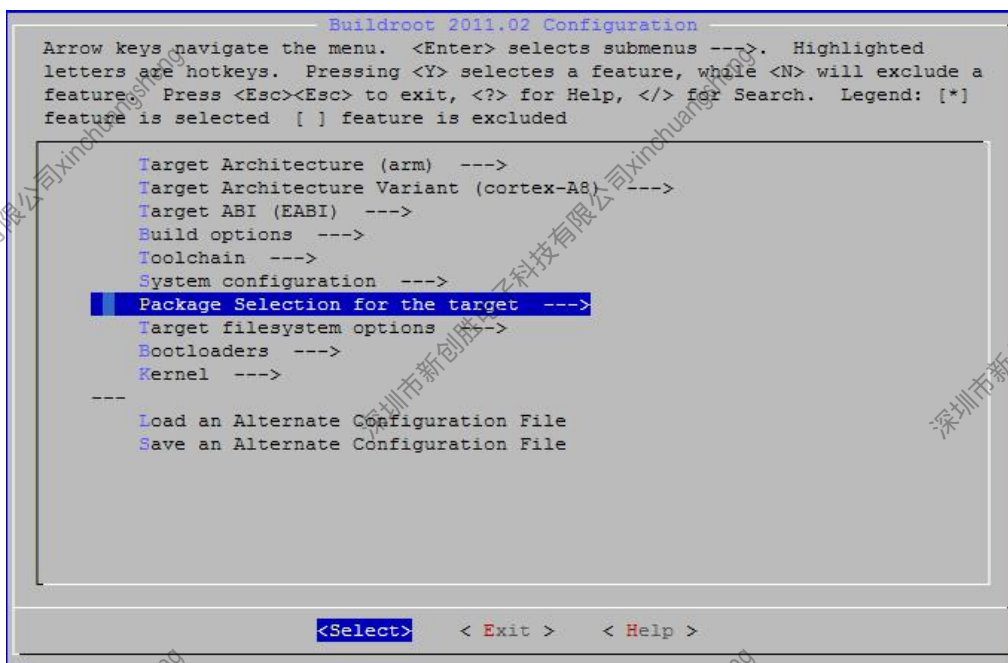


图 5.4 Buildroot make menuconfig 界面

做如图的选择，按下 enter 建

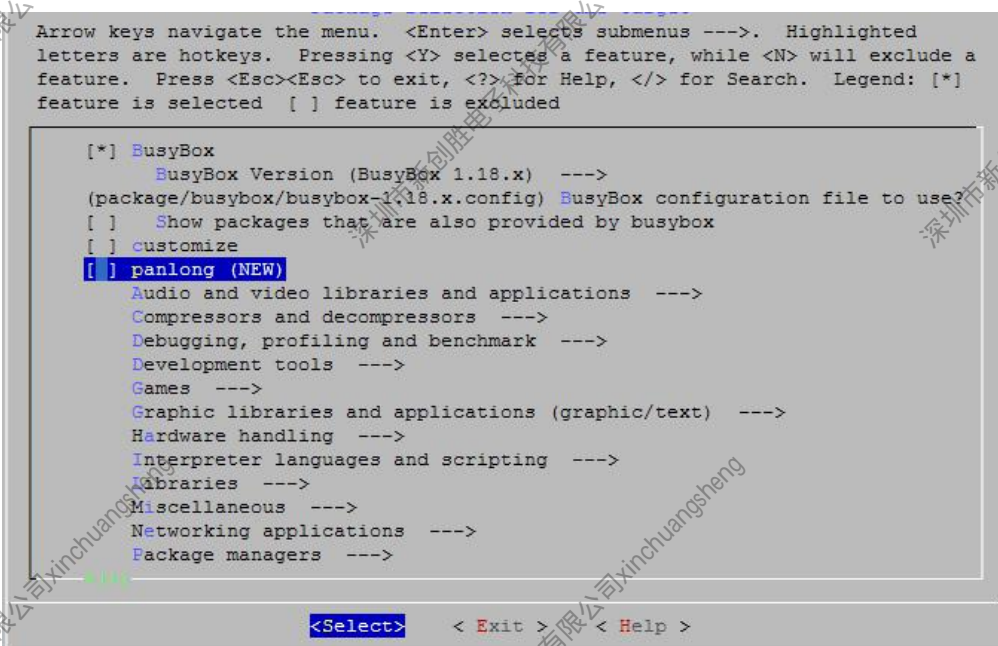


图 5.5 package selection for the target 子菜单界面

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-3.10/drivers 下面，如果无法直接跟 kernel 的 menuconfig 集成，则应该放在 linux-3.10/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-3.10 所在的那一层目录
LICHEE_MOD_DIR==${LICHEE_KDIR}/output/lib/modules/${KERNEL_VERSION}KERN
L_VERSION= 3.10
CROSS_COMPILE= arm-linux-gnueabi-
ARCH=arm
```

这些参数的定义都在 linux-3.10/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-3.10/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 sun8i 的，我们就可以把 buildroot 下的 .config 重命名为 sun8i_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. Declaration

This document is the original work and copyrighted property of Allwinner Technology (“Allwinner”). Reproduction in whole or in part must obtain the written approval of Allwinner and give clear acknowledgement to the copyright owner.

The information furnished by Allwinner is believed to be accurate and reliable. Allwinner reserves the right to make changes in circuit design and/or specifications at any time without notice. Allwinner does not assume any responsibility and liability for its use. Nor for any infringements of patents or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Allwinner. This datasheet neither states nor implies warranty of any kind, including fitness for any particular application.