全志科技
Allwinner Technology

V40 项目

Lichee 使用说明书 V1.0

文档履历

版本号	日期	制/修订人	内容描述
V1.0	2015-5-30		正式版本

目 录

1. 前言

1.1. 编写目的

本文档用于介绍 V40 carlet 方案的 Linux BSP 的目录结构、固件定制和 Lichee 定制。

1.2. 使用范围

Allwinner V40 平台。

1.3. 阅读对象

V40 平台负责人、版本集成人员、SDK 开发人员。

2. 目录结构

```
├── brandy
├── buildroot
├── build.sh
├── linux-3.10
├── out
├── README
└── tools
```

2.1. Brandy

存放 boot0 和 u-boot 源码，其目录结构为

```
├── build.sh
├── arm-trusted-firmware-1.0
├── armv8_toolchain
├── extern-libs
├── gcc-linaro
├── pack_tools
├── toolchain
├── u-boot-2014.07
```

brandy 源码一共包含以下几个部分：用于烧录的 fes1，用于启动的 boot0，用于烧写和启动的 uboot，其中 fes1 和 boot0 代码生成的 bin 文件体积必须控制在 24K 以内。编译命令(没有修改该部分源码，就不需要编译 brandy)：

```
$ cd brandy
方法 1：一次性编译 boot 需要的所有文件
$ ./build.sh -p sun8iw11p1(生成 boot0、uboot、fes、atf)

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 sun8iw11p1_config            --配置 sun8iw11p1 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：u-boot、boot0、sboot 和 fes 源码，包括启动引导、量产烧写的代码。

pack_tools：打包时使用的打包工具的源码

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-3.10

Linux 内核源码目录，结构如下

```
├── android
├── arch
├── block
├── COPYING
├── CREDITS
├── crypto
├── Documentation
├── drivers
```

```
├── 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 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。

2.4. 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/sun8iw11p1/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\
  sun8iw11p1smp_android_magton_defconfig
  sun8iw11p1smp_defconfig
  sun8iw11p1smp_eyeseelinux_magton_defconfig
  sun8iw11p1smp_defconfig
  sun8iw11p1smp_min_defconfig
```

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

lichee\buildroot\scripts\mkrule

sun8iw11p1_android	**sun8i_defconfig**	**sun8iw11p1smp_android_magton_defconfig**
sun8iw11p1_dragonboard	sun8i_defconfig	sun8iw11p1smp_android_magton_defconfig
sun8iw11p1_linux	sun8i_defconfig	sun8iw11p1smp_defconfig
|	|	|
chip_platform[_business]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 sun8iw11p1_android 中，sun8iw11p1 表示芯片平台，Android 表示项目平台。	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

例如：

sun8iw11p1_android　　sun8i_defconfig　　sun8iw11p1smp_android_magton_defconfig

表示编译 sun8iw11p1 芯片的 Android 项目时，内核将会使用配置文件：**sun8iw11p1smp_android_magton_defconfig**

3.2. 编译方式

3.2.1. 编译方式 1： ./build.sh lunch 命令 （建议使用）

```
lichee$  ./build.sh lunch (执行命令)
  All available lichee lunch:
   0. sun50iw1p1-android
   1. sun50iw2p1-android
   2. sun8iw11p1-android    (sun8iw11p1：表示芯片平台， android：表示项目平台)
   3. sun8iw6p1-android-eagle
   4. sun8iw6p1-android-secure
   5. sun8iw6p1-android-aston
   6. sun8iw7p1-android-dolphin
   7. sun8iw7p1-android-secure
   8. sun8iw7p1-android-karaok
   9. sun8iw8p1-android
  10. sun9iw1p1-android-jaws
  11. sun9iw1p1-android-secure
  12. sun9iw1p1-android-optimus
Choice: 2 (选择编译 sun8iw11p1 的 Android)
```

3.2.2. 编译方式 2： ./build.sh config 命令

```
lichee$  ./build.sh config (执行命令)
Welcome to mkscript setup progress
All available chips:
   0. sun50iw1p1
   1. sun50iw2p1
   2. sun8iw11p1    (sun8iw11p1：芯片项目)
   3. sun8iw6p1
   4. sun8iw7p1
   5. sun8iw8p1
   6. sun9iw1p1
Choice: 2
All available platforms:
   0. Android      （Android： 项目平台）
   1. dragonboard
   2. linux
   3. eyeseelinux
Choice: 0
not set business, to use default!
```

```
LICHEE_BUSINESS=
using kernel 'linux-3.10':
select arch by kernel version and chip
==================================
INFO: ----------------------------------------
INFO: build lichee ...      //选择的编译项目信息
INFO: chip: sun8iw11p1
INFO: platform: android
INFO: business:
INFO: kernel: linux-3.10
INFO: arch: arm
INFO: board:
INFO: output: out/sun8iw11p1/android/
INFO: ----------------------------------------
INFO: build buildroot ...
```

提示：新增业务选择项，前提是配置对应选择的芯片平台，mkbusiness 文件

V40： 对应 sun8iw11p1

3.2.3. 编译方式 3： ./build.sh -p [chip]_[platform] -k [kernel]

```
lichee$ ./build.sh -p sun8iw11p1_android -k linux-3.10
not set business, to use default!
not set arch, to use default by kernel version
INFO: ----------------------------------------
INFO: build lichee ...    //编译信息显示
INFO: chip: sun8iw11p1
INFO: platform: android
INFO: business:
INFO: kernel: linux-3.10
INFO: arch: arm
INFO: board:
INFO: output: out/sun8iw11p1/android/
INFO: ----------------------------------------
INFO: build buildroot ...
installing external toolchain
```

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/sun8iw11p1/linux/common/buildroot/
```

目录结构如下

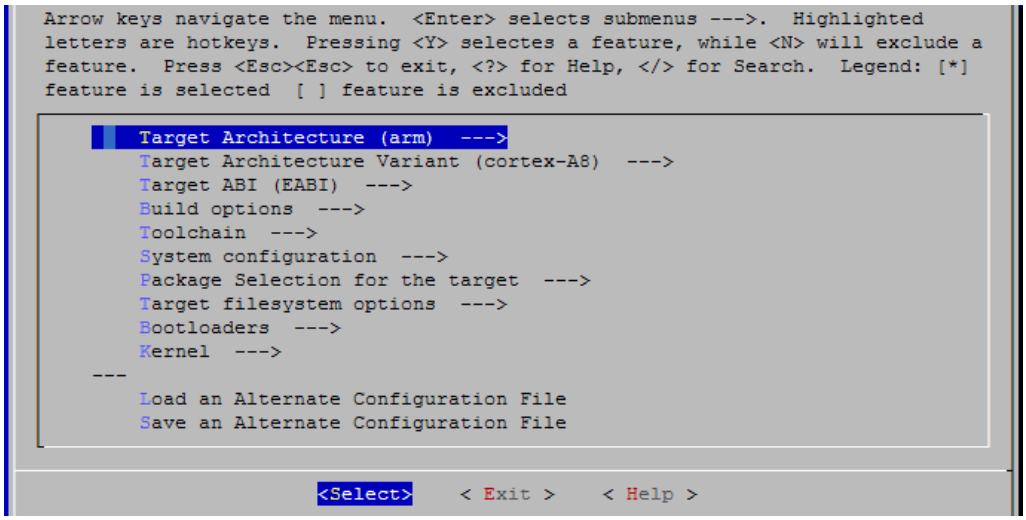
```
├── build
├── external-toolchain
├── host
├── images
├── Makefile
├── staging -> host/usr/arm-unknown-linux-gnueabi/sysroot
├── stamps
├── target
├── toolchain
└── toolchainfile.cmake
```

target 目录即 rootfs 的内容。

添加应用软件步骤：

1. $ cd out/sun8iw11p1/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/sun8iw11p1/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

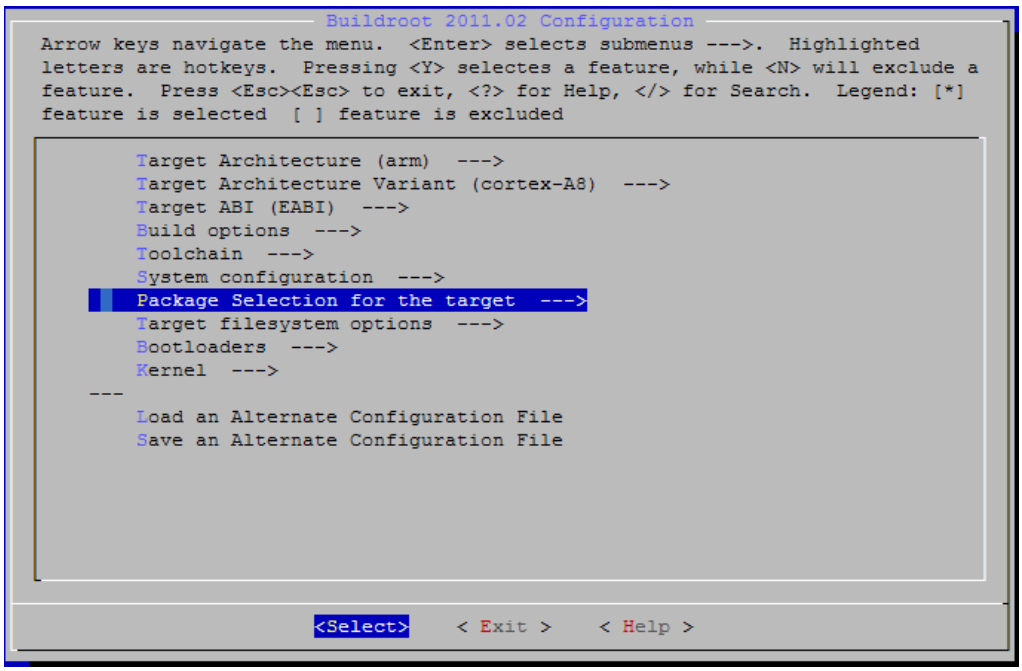


图 5.4 Buildroot make menuconfig 界面

做如图的选择，按下 enter 建

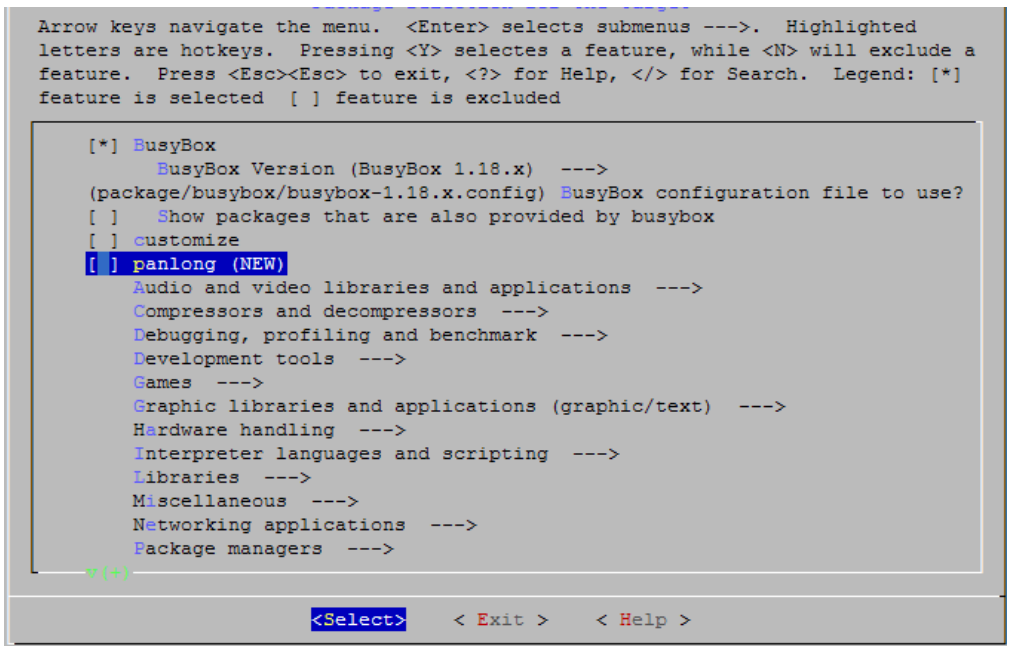


图 5.5 package selection for the target 子菜单界面

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}KERNEL_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