全志科技
Allwinner Technology

Tina

Linux SDK Quick Start Guide

文档履历

版本号	日期	制/修订人	制/修订记录
V1.0	2017/11/05		更新文档名称
V1.1	2018/06/08		加入移植软件包的内容

目　录

1. 开发环境准备

1.1. 硬件资源

1) 适用 R 系列或者 F 系列开发板(R6、R16、R18、R40、R30、R11、R7)
2) 电源适配器、usb 转串口线、jtag 小板、micro-usb 线
3) PC 或编译服务器

1.2. 软件资源

1) 编译服务器只能使用 Linux 64bit 的系统，建议安装 ubuntu12.04 64bit
2) 需要安装的软件包如下：
   gcc, binutils, bzip2, flex, python, perl, make,ia32-libs, find, grep, diff, unzip, gawk, getopt, subversion, libz-dev, libc headers
   ubuntu 可直接执行以下命令安装：
   sudo apt-get install build-essential subversion git-core libncurses5-dev zlib1g-dev gawk flex quilt libssl-dev xsltproc libxml-parser-perl mercurial bzr ecj cvs unzip ia32-libs -y
   详细信息参考：https://wiki.openwrt.org/zh-cn/doc/howto/buildroot.exigence
3) 固件烧录工具: LiveSuit (linux) 或 PhonixSuit (window)

2. SDK 下载指南

2.1. 代码下载方式

方式：从全志服务器 git 下载
电信网络地址：**61.143.53.198**
联通网络地址：**221.4.213.95**
方法：请参照《**SDK 仓库下载说明**》
示例：客户提供 ssh 公钥给 Allwinner 后，通过 git clone 下载
如下载 linux-4.4 内核，username 替换成客户下载账号的用户名。
$ git clone ssh://username@61.143.53.198/git_repo/R18_SDK/linux-4.4.git

3. 目录结构介绍

Tina Linux SDK 主要由构建系统、配置工具、工具链、host 工具包、目标机功能包、文档、脚本、linux 内核、bootloader 部分组成，下文按照目录顺序介绍相关的组成组件.

```
.
├── build
├── config
├── Config.in
├── dl
├── docs
├── lichee
├── Makefile
├── out
├── package
├── prebuilt
├── rules.mk
├── scripts
├── target
├── toolchain
└── tools
```

目录结构 1 根目录

3.1. build 目录

build 目录存放 Tina Linux 的构建系统文件，此目录结构下主要是一系列基于 Makefile 规格编写的 mk 文件。主要的功能是

1. 检测当前的编译环境是否满足 Tina Linux 的构建需求
2. 生成 host 包编译规则
3. 生成工具链的编译规则
4. 生成 target 包的编译规则
5. 生成 linux kernel 的编译规则
6. 生成系统固件的生成规则

```
.
├── autotools.mk
├── aw-upgrade.mk
├── board.mk
├── cmake.mk
├── config.mk
├── debug.mk
├── depends.mk
├── device.mk
├── device_table.txt
├── download.mk
├── dumpvar.mk
├── envsetup.sh
.....
```

目录结构 2 build 目录

3.2. config 目录

config 目录主要存放 Tina Linux 中配置菜单的界面以及一些固定的配置项，该配置菜单基于内核的 mconf 规格书写。

```
.
├── Config-build.in
├── Config-devel.in
├── Config-images.in
├── Config-kernel.in
└── top_config.in
```

目录结构 3 config 目录

3.3. docs 目录

docs 目录组要存放用于开发的文档，以 markdown 格式书写。

```
.
├── build.md
├── config.md
├── init-scripts.md
├── Makefile
├── network.md
├── tina.md
├── wireless.md
└── working.md
```

目录结构 4 docs 目录

3.4. lichee 目录

```
.
├── bootloader
│   ├── uboot_2011_sunxi_spl
│   └── uboot_2014_sunxi_spl
├── brandy
│   ├── add_hash
│   ├── add_hash.sh
│   ├── arm-trusted-firmware-1.0
│   ├── armv8_toolchain
│   ├── basic_loader
│   ├─
│   ├── build.sh
│   ├── extern-libs
│   ├── gcc-linaro
│   ├── pack_tools
│   ├── SUNXI_README
│   ├── toolchain
│   ├── u-boot-2011.09
│   └── u-boot-2014.07
├── linux-3.10
└── linux-3.4
└── linux-4.4
```

brandy/bootloader 目录用于存放 bootloader，其中 u-boot-2011.09 用于存放 R7/R11/R8/R16/R58 的 bootloder，u-boot-2014.07 用于存放 R6/R18/R40 的 bootloader；

linux-3.10 是 R6/R40 使用的内核源码目录；linux-3.4 是 R7/R11/R8/R16/R58 使用的内核源码目录

Linux-4.4 是 R18、R30 对应的内核源码目录；

3.5. package 目录

package 目录存放 target 机器上的软件包源码和编译规则。目录按照目标软件包的功能进行分类。

```
.
├── allwinner
├── base-files
├── devel
├── dragonst
├── firmware
├── kernel
├── lang
├── libs
├── luci
├── luci.mk
├── Makefile
├── minigui
├── multimedia
├── network
├── routing
├── system
└── utils
```

目录结构 5 package 目录

3.6. prebuilt 目录

prebuilt 目录是交叉编译器保存目录，详细请参考文档《Tina Linux 交叉编译器介绍_v1.1》，目录结构如下：

```
.
└── gcc
    └── linux-x86
        ├── aarch64
        │   ├── aarch64-toolchain.txt
        │   └── toolchain-sunxi
        ├── arm
        │   ├── arm-toolchain.txt
        │   └── toolchain-sunxi
        └── host
            └── host-toolchain.txt
```

目录结构 7 prebuilt 目录

3.7. scripts 目录

scripts 目录用于存放 pc 端或小机端使用的一些脚本。

```
.
├── arm-magic.sh
├── brcmImage.pl
├── bundle-libraries.sh
├── checkpatch.pl
├── clang-gcc-wrapper
├── cleanfile
├── clean-package.sh
```

```
├──── cleanpatch
├──── combined-ext-image.sh
├──── combined-image.sh
├──── config
├──── config.guess
├──── config.rpath
.....
```

目录结构 6 scripts 目录

3.8. target 目录

target 目录用于存放目标板相关的配置以及 sdk 和 toolchain 生成的规格

```
.
├──── allwinner
├──── Config.in
├──── imagebuilder
├──── Makefile
├──── sdk
└──── toolchain
```

目录结构 7 target 目录

3.9. toolchain 目录

```
.
├──── binutils
├──── Config.in
├──── fortify-headers
├──── gcc
├──── gdb
├──── glibc
├──── info.mk
├──── insight
├──── kernel-headers
├──── Makefile
├──── musl
└──── wrapper
```

目录结构 8 toochain 目录

工具链相关软件包，其中 gcc 目录为 gcc 编译器的编译规则目录，gdb 为 c / c++调试器编译规则目录，glibc 目前没有启用，默认使用的是 musl 下的 musl-libc。

3.10. tools 目录

tools 目录用于存放 host 端工具的编译规则。

```
.
├──── autoconf
├──── automake
├──── aw_tools
├──── b43-tools
├──── bc
├──── bison
├──── ccache
├──── cloog
```

```
├─── cmake
├─── dosfstools
├─── e2fsprogs
├─── elftosb
.....
```

目录结构 9 tools 目录

3.11. out 目录

out 目录用于保存编译相关的临时和最终镜像文件，编译后自动生成此目录。以编译 R18 为例，out 目录如下：

```
.
├── tulip-d1
└── host
```

目录结构 10 out 目录

其中 host 目录用于存放 host 端的工具以及一些开发相关的文件。其他的目录均为方案对应的目录，R18 对应的方案为 tulip-d1。

以 tulip-d1 为例说明方案目录下的结构：

```
.
├── tulip-d1-uImage
├── boot.img
├── compile_dir
├── image
├── md5sums
├── packages
├── rootfs.img
├── sha256sums
├── staging_dir
├── tina_tulip-d1_card0.img
└── tina_tulip-d1_uart0.img
```

目录结构 11 out/tulip-d1 目录

boot.img 为最终烧写到系统 boot 分区的数据，该分区默认为 vfat 格式。
rootfs.img 为最终烧写到系统 rootfs 分区的数据，该分区默认为 squashfs 或者 ext4fs 格式。
tulip-uImage 为内核最终的镜像，会打包到 boot.img 中。
compile_dir 为 sdk 编译 host，target 和 toolchain 的临时文件目录，存有各个软件包的源码。
staging_dir 为 sdk 编译过程中保存各个目录结果的目录。
packages 目录保存的是最终生成的 ipk 软件包。
tina_tulip-d1_card0.img 为最终固件包(系统镜像)，串口信息转递到 tf 卡座输出。
tina_tulip-d1_uart0.img 为最终固件包(系统镜像)，串口信息通过串口 0 输出。

4. 编译系统的内部工作原理

具体流程省略，这里简单介绍一下构建系统的工作流程

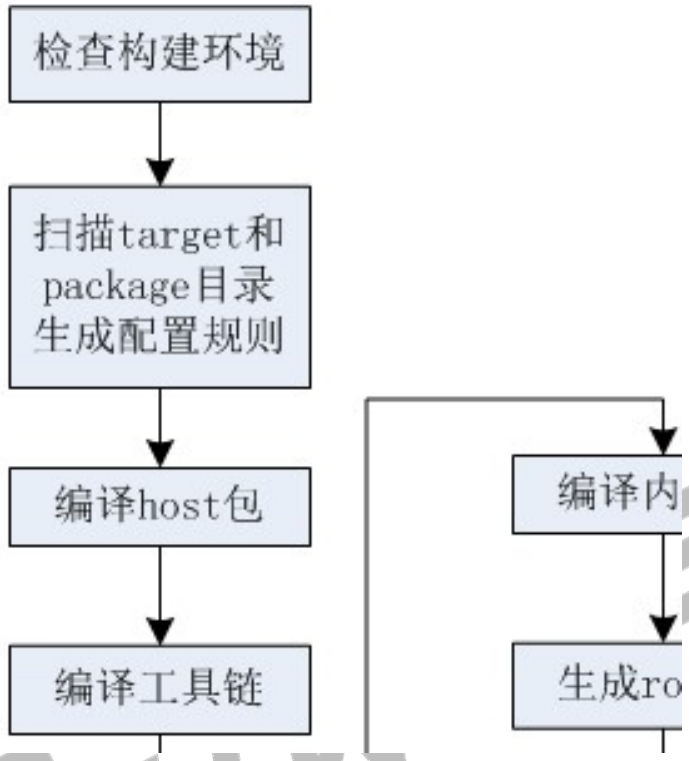

图 4-1 构建系统工作流程

5. 代码的编译与打包

```
$ source build/envsetup.sh
    => 设置环境变量
$ lunch
    => 选择方案，详见下文
$ make [-jN]
    => 编译, -jN 参数选择并行编译进程数量
$ pack [-d]
    => 打包, -d 参数使生成的固件包串口信息转到 tf 卡座输出. 例见 out 目录 章节
```

方案选择说明:
R40 开发方案 => azalea_* (以 azalea 开头)
R16 开发方案 => astar_*
R8 开发方案 => nuclear_*
R58 开发方案 => octopus_*
R18 开发方案 => tulip_*
R7/R11 开发方案 => banjo_*
编译完成后系统镜像会打包在 out/<board>/ 目录下，参考 out 目录章节。

5.1. SDK 的配置

Tina Linux SDK 的根目录下，执行 make menuconfig 命令进入 Tina Linux 的配置界面。该配置界面和 Linux Kernel 的配置界面相似，配置选项中有 y/m/n 三种配置选择，配置的意义如下:
<*> (按 y)：表示该软件包将包含在固件中
<M>(按 m)：表示该软件将会被编译,但不会包含在固件中
< >(按 n)：表示该软件不会被编译
配置完成后就可以按照上面的方法编译 SDK

5.2. SDK 中内置的命令

SDK 中默认集成了很多好用的命令，大部分命令必须在执行完以下两部后才能使用：

```
$ source build/envsetup.sh
$ lunch
```

命令列表:

命令	命令有效目录	作用
make	tina 根目录	编译整个 sdk，参考 3.3 节
make menuconfig	tina 根目录	启动软件包配置界面
make kernel_menuconfig	tina 根目录	启动内核配置界面
printfconfig	tina 下任意目录	打印当前 SDK 的配置
croot	tina 下任意目录	cd 到 tina 根目录
cconfigs	tina 下任意目录	cd 到方案的 bsp 配置目录
cdevice	tina 下任意目录	cd 到方案配置目录
cgeneric	tina 下任意目录	cd 到 generic 目录
cout	tina 下任意目录	cd 到方案的输出目录
cgrep	tina 下任意目录	在 c / c++ / h 文件中查找字符串
minstall path/to/package/	tina 根目录	编译并安装软件包
mclean path/to/package/	tina 根目录	clean 软件包
mm [-B]	软件包的目录	编译软件包, -B 参数: 在编译前先 clean 软件包

5.3. SDK 中预定义的变量

在 SDK 中默认会定义一些常用的目录或变量，这些定义的变量可以在 SDK 的 Makefile 中使用。

5.3.1. SDK 中导出的环境变量

变量名	含义
TARGET_PRODUCT	目标板的名称，定义在 target/allwinner/<方案>/vendorsetup.sh 中。其中<方案>的目录名称和 TARGET_PRODUCT 必须相同
TARGET_BUILD_VARIANT	SDK 构建的类型，有 tina 和 dragonboard 可选。 tina 即为 tina linux 的选项 dragonboard 为针对该板子的板卡测试的选项。
TARGET_BUILD_TYPE	SDK 的开发进展 devel：开发中的固件 release：发布的固件
TARGET_PLATFORM	平台名称 nuclear，astar，octopus，tulip，azalea，banjo 等
TARGET_BUILD_ARCH	平台的名称，有 arm 和 arm64 可选。
TARGET_BUILD_CPU	CPU 的类型，根据平台不同有 cortex-a7，cortex-a8，cortex-a57 等。
TARGET_PLATFORM_VERSION	目前 SDK 的版本

5.3.2. SDK 中定义的一些常用的变量

变量名	含义
ARCH	当前构建目标机器的 ARCH，该值和 TARGET_BUILD_ARCH 环境变量相同。
BOARD	目标板的名称，和 TARGET_PRODUCT 环境变量相同
DL_DIR	下载目录，即为 dl 目录
TARGET_OUT_DIR	方案的编译输出目录，即为 out/<方案>
BUILD_DIR	SDK 编译系统目录，即为 build
SCRIPT_DIR	SDK 存放脚本的目录，即为 tina/scripts
COMPILE_DIR_BASE	SDK 中编译目录，即在 TARGET_OUT_DIR 下的 compile_dir 目录
COMPILE_DIR	SDK 中目标软件包的编译目录，即在 COMPILE_DIR_BASE 下的 target 目录
COMPILE_DIR_TOOLCHAIN	SDK 中编译工具链的目录，即在 COMPILE_DIR_BASE 下的 toolchain 目录
STAGING_DIR	SDK 中存放目标软件包编译结果的目录，即在 TARGET_OUT_DIR 目录下的 staging_dir/tareget 目录
TOOLCHAIN_DIR	SDK 中存放工具链的目录，即在 TARGET_OUT_DIR 目录下的 staging_dir/toolchain 目录
TARGET_DIR	存放目标根文件系统的目录，即为 COMPILE_DIR 下的 rootfs 目录
STAGING_DIR_ROOT	存放完整文件系统的目录，即为 STAGING_DIR 目录下的 roofs 目录

5.3.3. Makfile 中常用的几个命令

命令变量名称	命令含义
TARGE_CC	目标机器 c 语言编译器
TARGET_CXX	目标机器 c++语言编译器
SED	等同于 sed -i -e
CP	等同于 cp -fpR
LN	等同于 ln -sf

XARGS	等同于 xargs -r
BASH	等同于 bash
TAR	等同于 tar
FIND	等同于 find
PATCH	等同于 patch
PYTHON	等同于 python
INSTALL_BIN	等同于 install -m0755
INSTALL_DIR	等同于 install -d m0755
INSTALL_DATA	等同于 install -m0644
INSTALL_CONF	等同于 install -m0600

5.4. 创建和移植软件包

5.4.1. 概述

在 Tina Linux SDK 中一个软件包目录下通常包含如下两个目录和一个文件:
package/<name>/Makefile
package/<name>/patches/ [可选]
package/<name>/files/ [可选]
其中: patches 保存补丁文件，在编译前会自动给源码打上所有补丁
files 保存软件包的源码，在编译时会对应源码覆盖源码中的源文件
Makefile 编译规则文件，例见下一节

5.4.2. Makefile 范例

（此处的 Makefile 仅提供一个简单的范例，更为详细的说明请参考 OpenWrt 的 Wiki：Creating Packages：https://openwrt.org/docs/guide-developer/packages）

该 Makefile 的功能是软件源码的准备,编译和安装的过程,提供给 Tina Linux 识别和管理软件包的接口,软件的编译逻辑是由软件自身的 Makefile 决定,理论上和该 Makefile(该 Makefile 只执行 make 命令和相关参数)无实质关系.

```
include $(TOPDIR)/rules.mk

PKG_NAME:=bridge                                        #软件包的名字
PKG_VERSION:=1.0.6                                      #软件包版本(详见①)
PKG_RELEASE:=1                                          #该 Makefile 的版本

PKG_SOURCE:=bridge-utils-$(PKG_VERSION).tar.gz          #软件包文件名(dl 目录下)
PKG_SOURCE_URL:=@SF/bridge                              #该包的下载地址(详见②)
PKG_MD5SUM:=9b7dc52656f5cbec846a7ba3299f73bd            #该包的 md5 值(详见③)
PKG_CAT:=zcat                                           #该软件包的解压方法
PKG_BUILD_DIR:=$(COMPILE_DIR)/bridge-utils-$(PKG_VERSION) #该软件包的编译目录

include $(BUILD_DIR)/package.mk

define Package/bridge                                   #Package/<包名>(详见④)
    SECTION:=net                                        #该包的类别,目前没有使用
    CATEGORY:=Base system                               #该包在配置系统中的分类
    TITLE:=Ethernet bridging configuration utility      #该包的简单的描述信息
    URL:=http://bridge.sourceforge.net/                 #该包的原始下载连接
```

```
        [可选]MAINTAINER:=                          #该包的维护人员的联系方式
        [可选]DEPENDS:=                             #该包的依赖关系(详见⑤)
        [可选]BUILDONLY:=                           #固定编译(详见⑥)
    endef

    [可选]
    define Package/<name>/conffiles:
            #指定软件包依赖的配置文件,一个配置文件一行
    endef

    [可选]
    define Build/Prepare:
            #准备源码和给源码打补丁
    endef

    [可选]
    define Package/bridge/description
            #该软件包的文字描述
        Manage ethernet bridging:
        a way to connect networks together to form a larger network.
    endef

    [可选]
    define Build/Configure
        #配置软件包(详见⑦)
        $(call Build/Configure/Default, \
        --with-linux-headers="$(LINUX_DIR)" \
    )
    endef

    [可选]
    define Build/Compile
            #执行软件编译的动作(详见⑧)
    endef

    define Package/bridge/install                   #执行软件包的安装动作(详见⑨)
        $(INSTALL_DIR) $(1)/usr/sbin
        $(INSTALL_BIN) $(PKG_BUILD_DIR)/brctl/brctl $(1)/usr/sbin/
    endef

    $(eval $(call BuildPackage,bridge))             #BuildPackage 的宏(详见⑩)
```

详注:

①如果是开源软件,软件包版本建议与下载软件包的版本一致

②以 PKG 开头的变量主要告诉编译系统去哪里下载软件包.其中@SF 是一个内置的特殊关键字,表示去 sourceforge 网站去下载软件包,类似的关键字还有@GNU 表示从 GNU 下载软件源码包.

③md5sum 用于校验下载下来的软件包是否正确,如果正确,在编译该软件的时候,就会在 PKG_BUILD_DIR 下找到该软件包的源码.

④Package/<name>: <name>用来指定该 Package 的名字,该名字会在配置系统中显示

⑤使用依赖包的名字<name>来指定依赖关系,具体如下:

+foo: 若软件包<name>选上了，则另一个软件包 foo 也会被选上

foo: 若软件包 foo 没有被选上，则软件包<name>不可见（也无法被选上）

@FOO: 若符号（symbol）CONFIG_FOO 没有被选上（没有被设为 y），则软件包<name>不可见（也无法被选上）

⑥如果该值为 1,该包将不会出现在配置菜单中,但会作为固定编译,可选
⑦在开源软件中一般用来生成 Makefile,其中参数可以通过 CONFIGURE_VARS 来传递
⑧在开源软件中一般相当于执行 make,其中有两个参数可以使用:MAKE_FLAGS 和 MAKE_VARS
⑨内置的几个关键字如下:
INSTALL_DIR 相当于 install -d m0755
INSTALL_BIN 相当于 install -m0755
INSTALL_DATA 相当于 install -m0644
INSTALL_CONF 相当于 install -m0600
⑩该 Makefile 的所有 define 部分都是为该宏的参数做的定义.上层 Makefile 通过调用此宏进行编译

5.5. 软件的调试

在编译过程中如果出错,可以使用如下命令查看更多信息:

1. 当配置菜单中没有显示软件包时,可以使用如下命令分析出错的原因:
   $ TOPDIR=$PWD make -C package/<name> DUMP=1 V=s
2. 当编译时出错,相关的命令: [参考“SDK 中内置的命令”章节]
   $ make package/<name>/clean 单独清除编译软件包的中间文件
   $ make package/<name>/install 单独编译软件包
   也可进入软件包目录后执行: $ mm -B

6. 固件的烧写

使用 PC 端的工具烧写固件.
windows 使用 PhoenixSuit，使用参考《PhoenixSuit 使用说明文档.doc》
Linux 使用 LiveSuit，使用请参考 LiveSuitV306_For_Linux64.zip 解压后 PhoenixSuit 说明文的《Readme》

7. 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 acknowledgment 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.