

Lichee2.6.36_Linux User Manual

Revision History

<i>Version</i>	<i>Date</i>	<i>Author</i>	<i>Change Description</i>
0.1	2011-5-31	Benn	初稿
0.2	2011-07-19	Benn	根据实际对文档作了调整

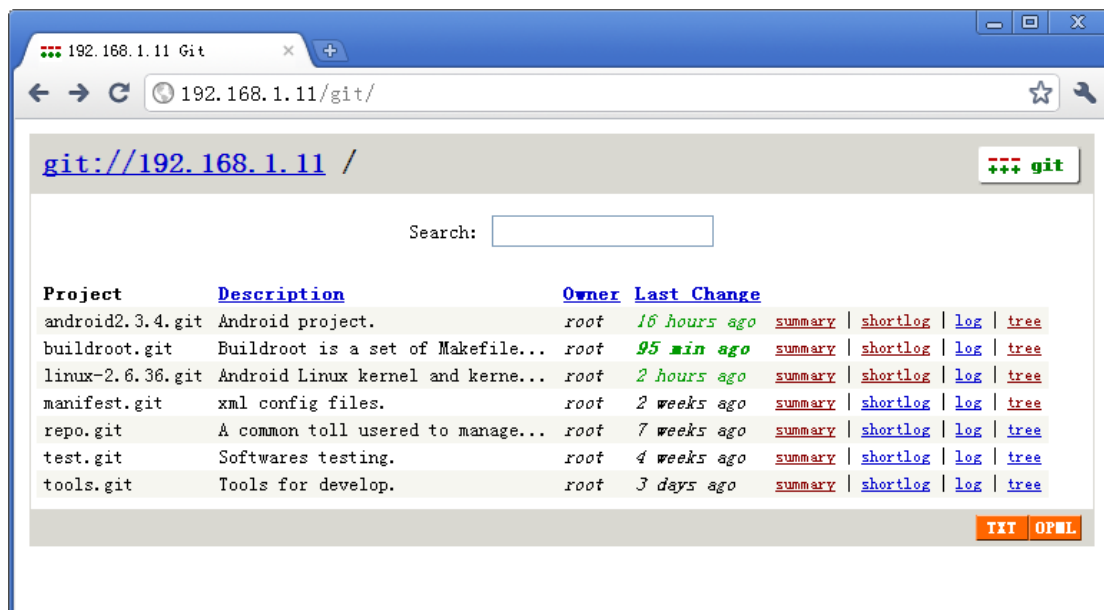
1. 概述

本文档用于介绍 Lichee2.6.36 Linux BSP（后面简称 Linux BSP）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 Linux BSP。

2. 目录结构介绍

Lichee2.6.36 Linux BSP 主要由 Buildroot, Linux kernel 两大部分组成。其中 Buildroot 负责 ARM 工具链、U-Boot、应用程序软件包、Linux 根文件系统和固件包的生成；Linux Kernel 是 Linux BSP 的核心部分。

Linux BSP 采用 Git 版本控制系统，并结合了 Android 的 repo 辅助工具。在 Git 版本库上，我们有多组并列的 Git 仓库，我们可以通过 git-web 界面直接访问。如下图所示



上图中，列举了 android2.3.4.git, buildroot.git, linux-2.6.36.git, manifest.git, repo.git, test.git 和 tools.git 总共 7 个独立的 git 仓库。我们这里只涉及到 buildroot.git, linux-2.6.36.git, manifest.git, repo.git 和 test.git，另外 2 个属于 Android 系统的。以下是各个 git 仓库的介绍。

2.1. buildroot.git

它的主要作用是

- (1) 管理包之间的依赖关系
- (2) 生成 ARM 交叉工具链
- (3) 生成 U-Boot
- (4) 制作根文件系统，可以包含 strace, directfb, oprofile 等非常丰富的应用软件和测试软件
- (5) 生成最终用于烧写的固件包
- (6)

它的目录结构如下

```
[git@itlab test]$ tree -L 1 buildroot/
buildroot/
|-- board
|-- boot
|-- CHANGES
|-- Config.in
|-- configs
```

```
|-- COPYING
|-- dl
|-- docs
|-- fs
|-- linux
|-- Makefile
|-- package
|-- readme.txt
|-- scripts
|-- target
|-- tools
`-- toolchain
```

其中，boot 目录里存放 Boot 代码，config 目录里存放预定义好的配置文件，比如我们的 sun4i_defconfig，dl 目录里存放已经下载好的软件包，scripts 目录里存放 buildroot 运作的代码，target 目录里存放用于生成根文件系统的一些规则文件，在 tools 目录中存放的是用于打包和量产的 pc 工具。对于我们来说最为重要的是 package 目录，里面存放了将近 3000 个软件包的生成规则，我们可以在里面添加我们自己的软件包或者是中间件。更多关于 buildroot 的介绍，可以到 buildroot 的官方网站 <http://buildroot.uclibc.org/> 获取。

2.2. linux-2.6.36.git

目录结构如下，

```
[benn@LServer lichee]$ tree -L 1 linux-2.6.36/
linux-2.6.36/
|-- arch
|-- block
|-- build
|-- COPYING
|-- CREDITS
|-- crypto
|-- Documentation
|-- drivers
|-- firmware
|-- fs
|-- include
|-- init
|-- ipc
|-- Kbuild
```

```
|-- kernel
|-- lib
|-- MAINTAINERS
|-- Makefile
|-- mm
|-- modules
|-- net
|-- output
|-- README
|-- REPORTING-BUGS
|-- rootfs
|-- samples
|-- scripts
|-- security
|-- sound
|-- tools
|-- usr
`-- virt
```

以上目录结构跟标准的 Linux 内核是一致的，除了多一个 modules 目录。**modules 目录**是我们扩展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 example, mali 和 wifi 这 3 个外部模块，其中 example 是示例用的，mali 是我们的 3D GPU 驱动。

```
[benn@LServer lichée]$ tree -L 1 linux-2.6.36/modules/
linux-2.6.36/modules/
|-- example
|-- mali
`-- wifi
```

2.3. manifest.git (可选)

manifest 和 repo 是 Google 管理 Android 代码时必备的 2 个部分。对于我们的 Linux BSP 系统来说，是可选的。用户可以只适用 git 去维护它的代码，也可以选择使用 git/repo + manifest 去管理它的代码。普通用户建议只用 git 维护代码，Linux BSP 维护人员建议使用 git/repo + manifest 的方式。

在 manifest.git 里面只包含一个个单独的 xml 文件，比如 sun3i.xml, sun4i.xml, sun4i-0.83.xml, sun5i.xml, default.xml 等。每一个 xml 文件对应一个版本，里面规定我们需要从那些 git 中取出哪些版本（branch, tag 或者某次 commit）的代码，以及怎样组织新的目录结构。

一个形象的理解是，1 个 xml 文件对应一份菜单，repo 工具根据这个菜单从版本库中

取得相应版本的代码构造出指定的工作目录。这其实也是 Android 代码管理的核心工作方式之一。

以下是 sun4i.xml 的一个配置文件的例子

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest>
  <remote name="korg" fetch="git://192.168.1.11/" />
  <default revision="master" remote="korg" />

  <project path="lichee/buildroot" name="buildroot">
    <copyfile src="scripts/top_build.sh" dest="lichee/build.sh" />
  </project>
  <project path="lichee/linux-2.6.36" name="linux-2.6.36"
    revision="b444d0a920b65e33bc4ad3fe8204c92f5a4e3b68" >
  </project>
</manifest>
```

2.4.repo.git(可选)

Google 为 Android 开发的工具，我们只有少量的修改。去掉了一些 GPG 签名 verify 的部分。Repo 是建立在 git 上面的一套命令集，用于辅助管理多个 Git 仓库。一般用户无需了解。

2.5.test.git

包含 Linux BSP 的测试代码。这里不做描述

3. 主要特点

(1) 分布式

采用 git 分布式版本控制系统

(2) 与 Android 代码管理方式兼容

git / repo 工具相结合，manifest 配置脚本定义版本

(3) 可扩展

可集成

(4) 傻瓜式

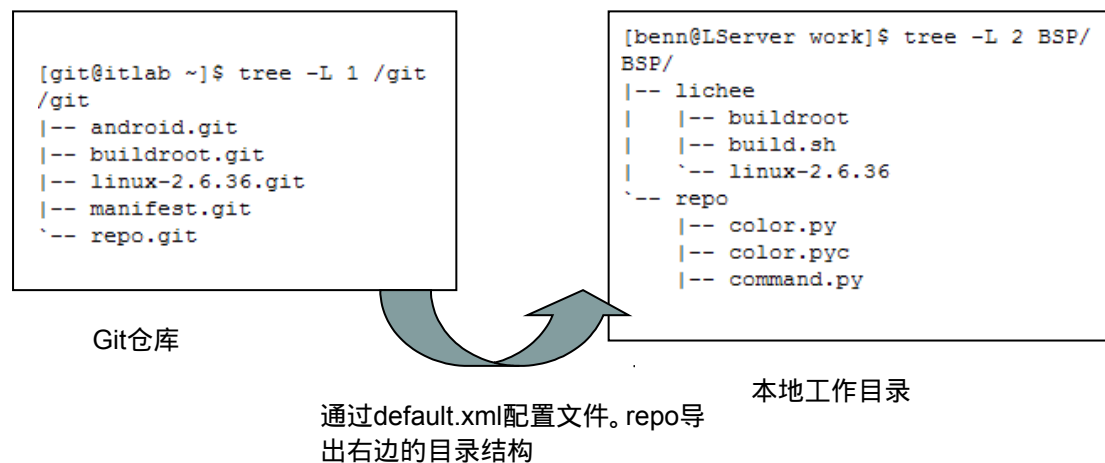
自动编译和生成固件包

4. 内部工作机制

4.1. Git 仓库与工作目录的映射关系

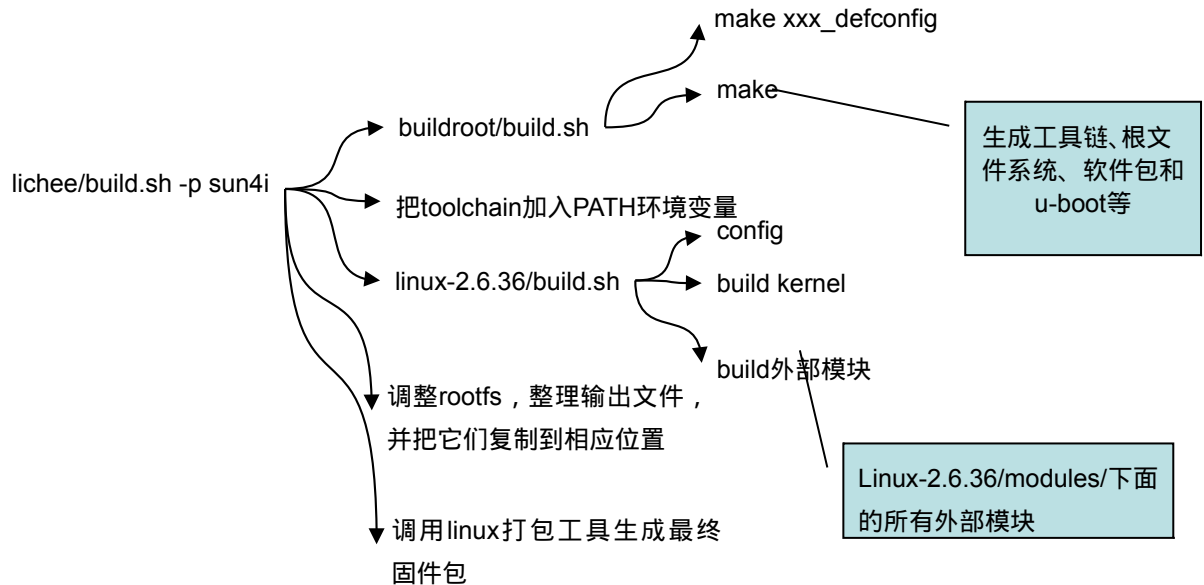
```
$mkdir BSP
$cd BSP
$git clone git://192.168.1.11/repo.git
$repo/repo init -u git://192.168.1.11/manifest.git
$repo/repo sync
```

执行完上述命令后，本地工作目录如下图所示



4.2. 自动化编译流程

以 sun4i 为例子



注意：

- (1) lichee/build.sh 是由 repo 生成的，他从 buildroot/scripts/top_build.sh 自动复制过来的，具体看 default.xml 即清楚
- (2) 在执行 build.sh 脚本时需要指定参数，具体可以参考 ./build.sh -h 输出

5. 获取和编译代码

现在假设要在 192.168.1.37 上获取最新的 Linux BSP 代码，并编译。以下内容中蓝色字体部分是需要用户输入的命令。如果您是第一次使用 git，则需要输入下面的命令把自己邮箱地址和用户名配置到 git 中

```
$git config --global user.email "you@allwinnertech.com"
$git config --global user.name "Your Name"
```

5.1. 只用 git 的方式

(1) ssh 登陆到 192.168.1.37 服务器上

需要在 192.168.1.37 上有自己的账号，建议使用 putty 登陆。Putty 的下载地址为 <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>

(2) 初始化工作目录并获取代码

```
$mkdir -pv workdir/lichee
$cd workdir/lichee
$git clone git://192.168.1.11/buildroot.git
$git clone git://192.168.1.11/linux-2.6.36.git
$cd ../
$cp buildroot/scripts/top_build.sh build.sh
```

执行完上面的命令，你的工作目录已经包含了最新的 Linux BSP 代码。

(3) 切换到用户指定的版本(可选)

比如此时用户想切换到 release-0.85 这个版本上，则做如下操作

```
$cd buildroot
$git checkout -b release-0.85 release-0.85
$cd ../
$cd linux-2.6.36
$git checkout -b release-0.85 release-0.85
```

(4) 编译代码

比如要编译的一个精简版的 Linux BSP 固件，则输入

```
$cd workdir
$./build.sh -p sun4i-lite
```

要编译一个完整版的 Linux BSP 固件，则输入

```
$cd workdir
$./build.sh -p sun4i
```

编译完代码，系统会把生成的用户可能有用的二进制文件复制到 lichee/out 目录下，同时还会把生成的固件包复制到 buildroot/tools/pack 下面的打包工具的工作目录下，并启动 Linux 打包工具生成最终的固件包，比如 buildroot/tools/pack/sun4i_pack_ddr3_lin/wboot/ePDKv100.img。用户拿到这个固件包后，直接使用 LiveSuit 工具烧写 Nand Flash 即可。

5.2. git/repo + manifest 的方式

(1) ssh 登陆到 192.168.1.37 服务器上

需要在 192.168.1.37 上有自己的账号，建议使用 putty 登陆。Putty 的下载地址为 <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>

(2) 初始化工作目录并获取代码

```
$mkdir -pv workdir/lichee  
$cd workdir  
git clone git://192.168.1.11/repo.git  
$repo/repo init -u git://192.168.1.11/manifest.git -m sun4i-0.85.xml  
$repo/repo sync
```

执行完上面的命令，你的工作目录已经包含了 release-0.85 的 Linux BSP 代码。

后面同上

6. 烧写固件

6.1. Nand 启动

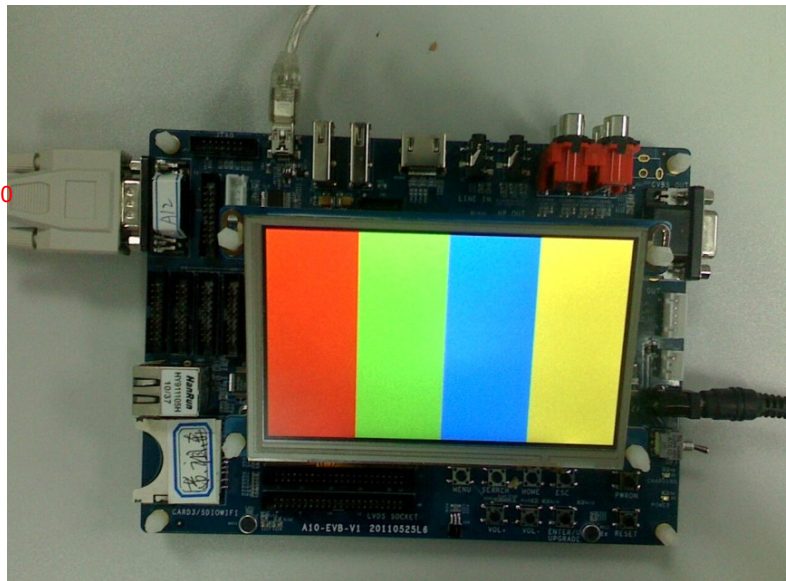
这里假设烧写的是 A10-EVB 板，下面是烧写固件包的步骤（从 Nand Flash 启动）

(1) 按照下图连接好开发板

串口、USB 和电源

OTG口上接PC端

串口
波特率：115200
数据流控制：
无



12V电源

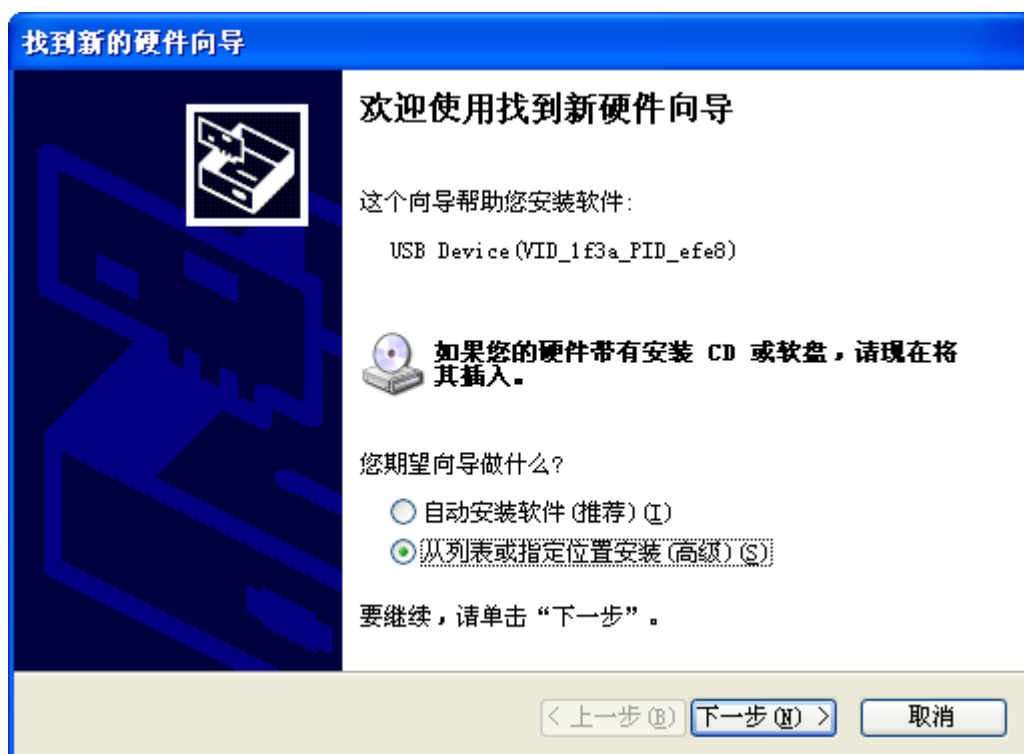
(2) 准备好固件包

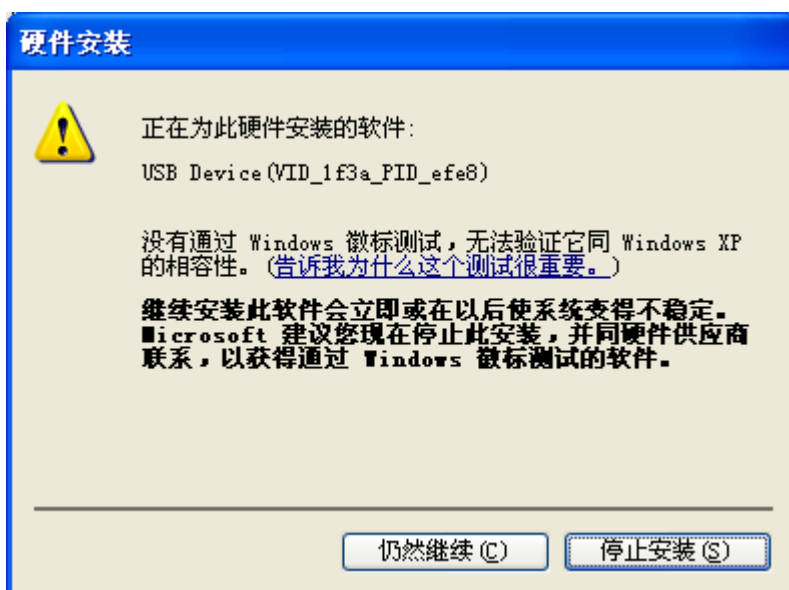
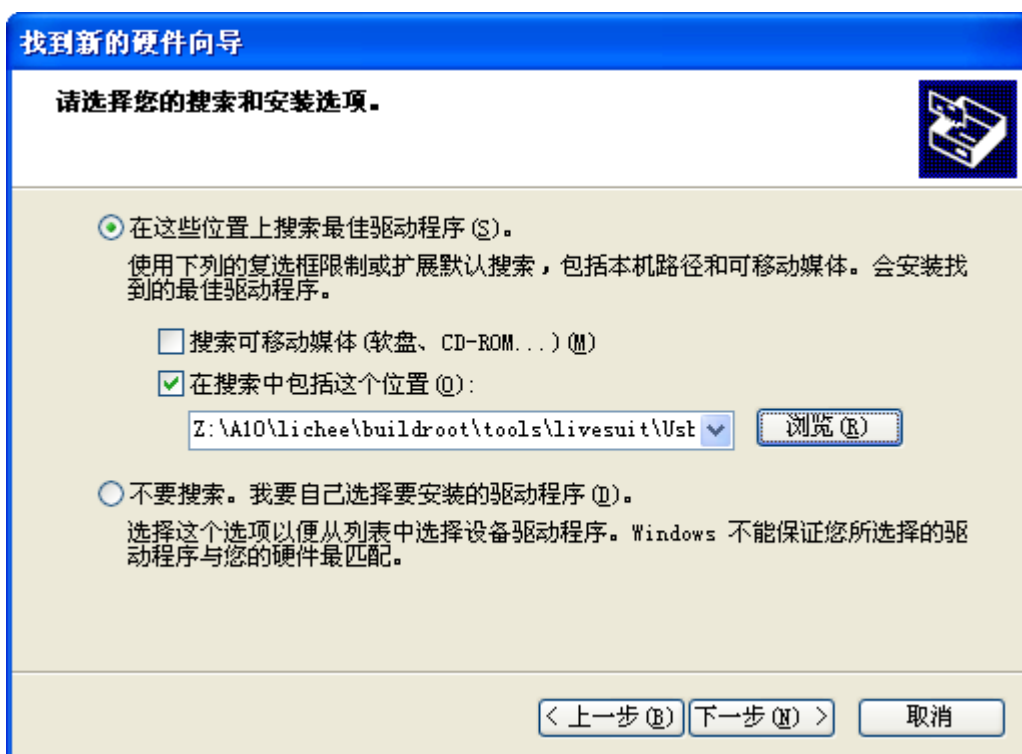
在 5.1 章节的最后，我们得到了 Linux BSP 生成的固件包。也可以从 b-server\ShareForAll\A10\Release 下面获得已经编译好的固件包。

(3) 运行 LiveSuit 工具

在 buildroot/tools/下面有这个工具，可以从下面把该工具复制到 Windows XP 上。线连接好以后，按 ENTER/UPGRADE 键 + RESET 键(具体是按住 ENTER/UPGRADE 键后再按 RESET 键，先松开 RESET 键，后松开 ENTER/UPGRADE 键)。然后会进入升级模式。

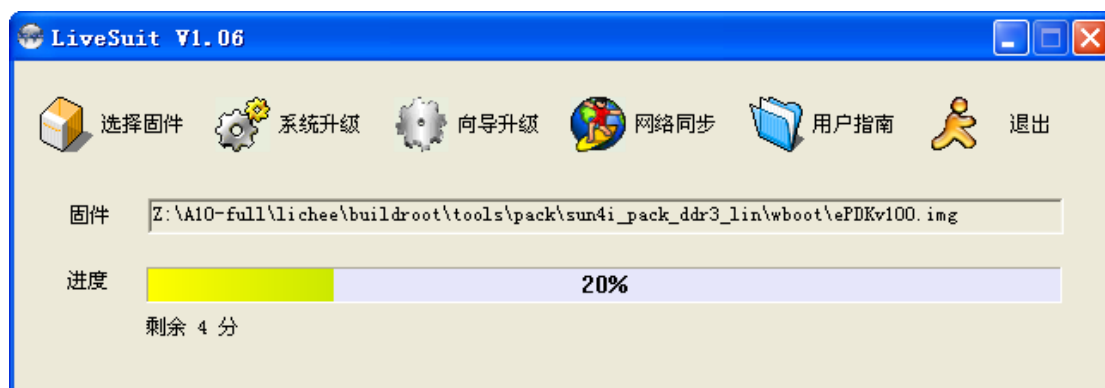
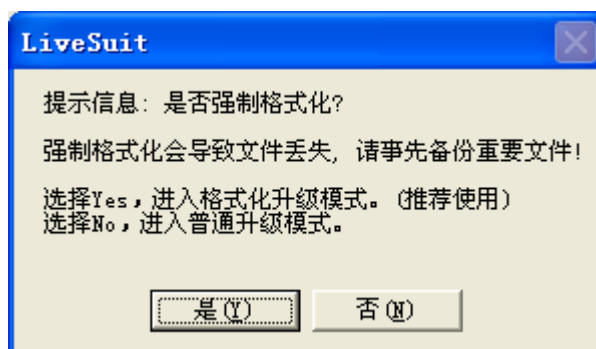
如果第一次使用 LiveSuit，则会提示需要安装 USB 驱动，驱动也放在 buildroot/tools/livesuit/UsbDriver 下面。按照提示安装 USB 驱动，如下图所示



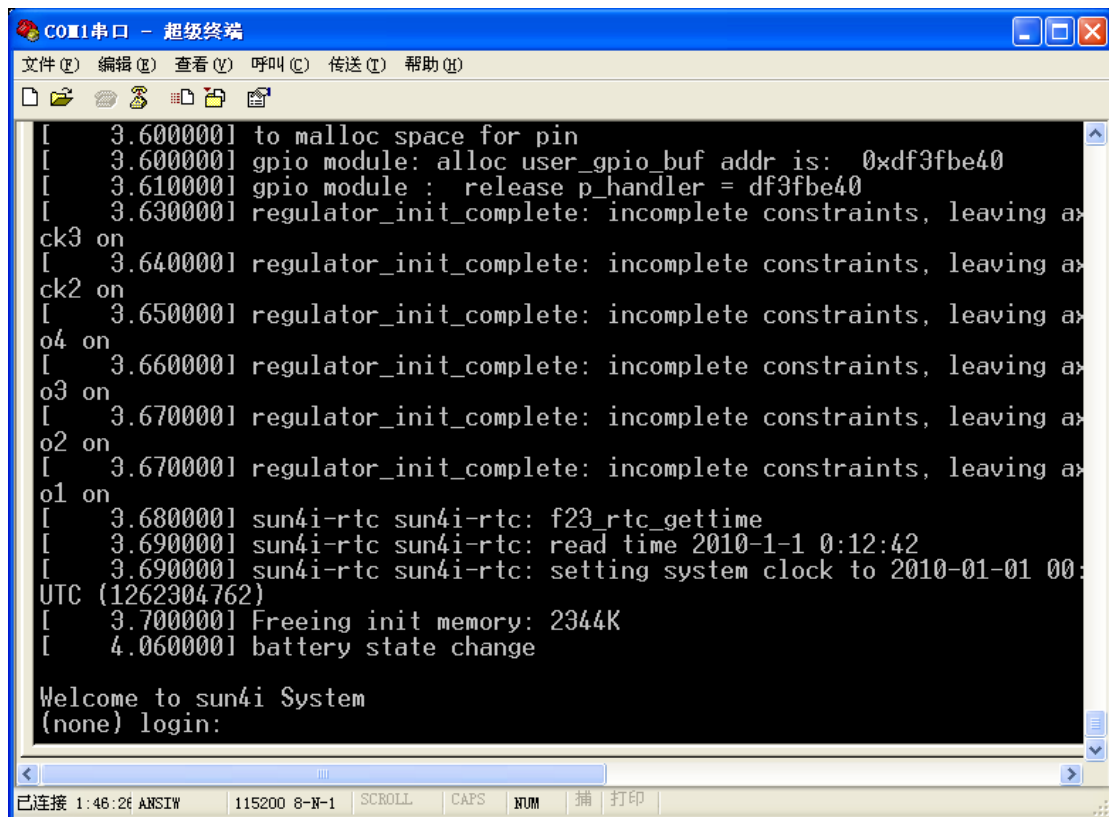




安装完驱动后就进入烧写流程。



烧写完，按 RESET



```
[ 3.600000] to malloc space for pin
[ 3.600000] gpio module: alloc user_gpio_buf addr is: 0xdf3fbe40
[ 3.610000] gpio module : release p_handler = df3fbe40
[ 3.630000] regulator_init_complete: incomplete constraints, leaving ax
ck3 on
[ 3.640000] regulator_init_complete: incomplete constraints, leaving ax
ck2 on
[ 3.650000] regulator_init_complete: incomplete constraints, leaving ax
o4 on
[ 3.660000] regulator_init_complete: incomplete constraints, leaving ax
o3 on
[ 3.670000] regulator_init_complete: incomplete constraints, leaving ax
o2 on
[ 3.670000] regulator_init_complete: incomplete constraints, leaving ax
o1 on
[ 3.680000] sun4i-rtc sun4i-rtc: f23_rtc_gettime
[ 3.690000] sun4i-rtc sun4i-rtc: read time 2010-1-1 0:12:42
[ 3.690000] sun4i-rtc sun4i-rtc: setting system clock to 2010-01-01 00:
UTC (1262304762)
[ 3.700000] Freeing init memory: 2344K
[ 4.060000] battery state change

Welcome to sun4i System
(none) login:
```

用 root 用户登陆，密码为空。

6.2. U-Boot 启动

6.3. SD 卡启动

7. 定制根文件系统

当前 Linux BSP 中使用了 2 级根文件系统。第一级根文件系统和 Linux 内核编译在一起，第二级根文件系统一般烧写到 Nand Flash 的分区中。

7.1. 修改 built-in rootfs

```
$cd /lichee/linux-2.6.36/rootfs
```

```
$./build.sh e sun4i_rootfs.cpio.gz
```

此时该目录下多了 skel 目录，里面是 built-in 的根文件系统，直接修改即可。

修改完后

```
$./build.sh c sun4i_rootfs.cpio.gz
```

更新完 rootfs 后记得需要重新编译 Linux Kernel

7.2. 修改 Nand Flash 的 rootfs

(1) 复制一份现有的配置文件

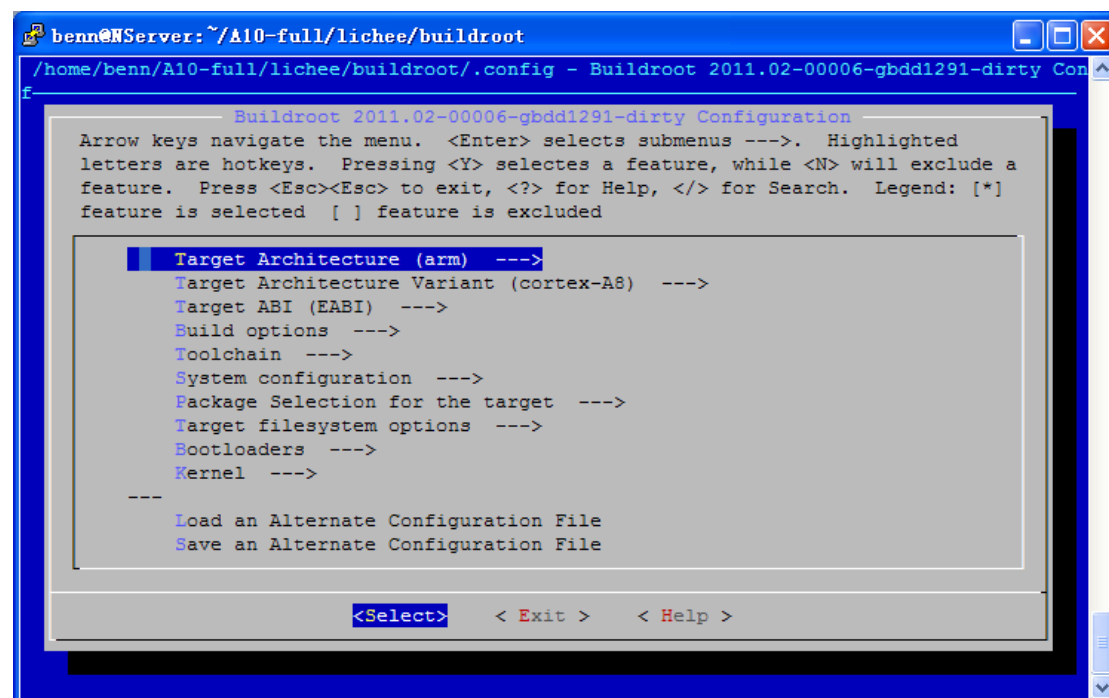
```
$cd /lichee/buildroot
```

```
$cp configs/sun4i_defconfig .config
```

(2) 进入 buildroot 界面进行配置

```
$make menuconfig
```

上述命令后，显示下面的界面



配置完后保存，然后到 lichee 目录下重新运行 build.sh 脚本。

编译过程中，如果有软件包缺失，则 buildroot 会自动从网上下载，而此时如果编译机器无法连接网络，则需要把软件包复制到 buildroot/dl 目录下面。

8. 定制内核

略

9. 集成软件包

9.1. 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>。如果还存在问题请联系本文档作者获得帮助。

对于内核驱动，应该尽量考虑放到 linux-2.6.36/drivers 下面，如果无法直接跟 kernel 的 menuconfig 集成，则应该放在 linux-2.6.36/modules 下面。

9.2. 二进制包

同上，只是忽略掉编译过程。可以参考 buildroot/packages/mali-3d 和 linux-2.6.36/modules/mali

10. 新增平台

大致步骤如下

1.1. 添加 buildroot config 文件

参考 buildroot/configs/sun4i_defconfig

1.2. 添加 buildroot 的 build.sh 脚本

参考 buildroot/scripts/build_sun4i.sh。根据需要修改 scripts/common.sh 脚本

1.3. 添加 kernel 的 config 文件

自己定义

1.4. 添加 kernel 的 build.sh 脚本

参考 linux-2.6.36/scripts/build_sun4i.sh

1.5. 添加平台的 manifest 文件

(可选)

参考 manifest.git 中的 sun4i.xml 文件

11. 运行 Android

参考 \\b-server\ShareForAll\benn\linux-android-20110712-debug

12. 附录

1.6. 附录 1 相关资源

git 帮助文档

<http://www.kernel.org/pub/software/scm/git/docs/user-manual.html>

repo 帮助文档

<http://source.android.com/source/version-control.html>

makefile 帮助文档

<http://www.gnu.org/software/make/manual/make.html>

buildroot 帮助文档

<http://buildroot.uclibc.org/downloads/buildroot.html>

Git 仓库

<http://android.git.kernel.org/>

<http://git.kernel.org/>