



Hi3520D/Hi3515A/Hi3515C 系统小型化 使用指南

文档版本 01

发布日期 2013-07-31

版权所有 © 深圳市海思半导体有限公司 2013。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HISILICON、海思和其他海思商标均为深圳市海思半导体有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

深圳市海思半导体有限公司

地址： 深圳市龙岗区坂田华为基地华为电气生产中心 邮编：518129

网址： <http://www.hisilicon.com>

客户服务电话： +86-755-28788858

客户服务传真： +86-755-28357515

客户服务邮箱： support@hisilicon.com



前 言

概述

在保证系统性能和业务运行要求的前提下，为满足系统小型化要求，Hi3520D/Hi3515A/Hi3515C 的内核和文件系统需要做适当的裁减。参考 Hi3518 的业务需求，Hi3520D/Hi3515A/Hi3515C 发布包给出了一套通用的小型化配置文件。本文首先介绍了如何使用这些配置文件编译小型化版本镜像文件；部分用户可能不会直接使用这套默认配置文件，而是根据其需求自行裁剪。有鉴于此，本文还介绍了 Hi3520D/Hi3515A/Hi3515C 内核和文件系统裁剪的一般方法。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
Hi3520D	V100
Hi3515A	V100
Hi3515C	V100

读者对象






本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 单板硬件开发工程师

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。



符号	说明
 危险	表示有高度潜在危险，如果不能避免，会导致人员死亡或严重伤害。
 警告	表示有中度或低度潜在危险，如果不能避免，可能导致人员轻微或中等伤害。
 注意	表示有潜在风险，如果忽视这些文本，可能导致设备损坏、数据丢失、设备性能降低或不可预知的结果。
 窍门	表示能帮助您解决某个问题或节省您的时间。
 说明	表示是正文的附加信息，是对正文的强调和补充。

文档版本 01(2013-07-31)

1.1 编译小型化版本 u-boot

修改 u-boot 小型化的相关命令。

2.1 uboot 裁剪

配置文件 hi3520d.h 中，增加部分修改代码；增加注意事项。

文档版本 00B03(2013-06-21)

新增 Hi3515C 的相关描述。

文档版本 00B02(2013-05-09)

1.1 编译小型化版本 u-boot 增加命令。

文档版本 00B01 (2013-04-03)

初稿。



目 录

前 言.....	i
1 如何使用发布包中给出的小型化配置文件编译、制作 Hi3520D/Hi3515A/Hi3515C 小型化版本的内核及文件系统？	1
1.1 编译小型化版本 u-boot	1
1.2 编译小型化版本的内核.....	1
1.3 在 Hi3520D/Hi3515A/Hi3515C 发布包中制作小型化版本的文件系统.....	2
2 Uboot、内核及文件系统裁剪的一般方法	4
2.1 uboot 裁剪.....	4
2.2 内核裁减.....	5
2.2.1 一些目前还处在开发或者完善过程中的模块	5
2.2.2 一些特殊功能和特性的支持.....	5
2.2.3 网络支持	6
2.2.4 设备驱动支持.....	7
2.2.5 文件系统类型支持.....	8
2.2.6 内核镜像文件的压缩方式.....	10
2.2.7 打印和调试信息.....	11
2.3 文件系统裁剪.....	11
2.3.1 busybox 配置选项	12
2.3.2 删除文件系统的可执行文件、相关的库文件中多余的调式信息和符号信息（即 strip elf）	14
2.3.3 使用 squashfs 根文件系统.....	14
2.3.4 剔除没有被使用的库文件.....	15



1 如何使用发布包中给出的小型化配置文件编译、制作 Hi3520D/Hi3515A/Hi3515C 小型化版本的内核及文件系统？

1.1 编译小型化版本 u-boot

如果要单独编译小型化版本，请使用修改后的配置文件 hi3520d_mini.h（修改方法请参见 [2.1 u-boot 裁剪](#)）。具体操作：

拷贝对应的芯片表格到 u-boot 目录

```
cp tools/pc_tools/u-boot_tools/reg_info_Hi3520D-  
bvt_No1_660_330_660_ddr_innerFEPHY.bin u-boot/u-boot-2010.06/.reg1  
cp tools/pc_tools/u-boot_tools/reg_info_Hi3515A-  
bvt_No1_600_300_600_ddr_innerFEPHY.bin u-boot/u-boot-2010.06/.reg2  
cp tools/pc_tools/u-boot_tools/mkboot-hi3520d.sh u-boot/u-boot-2010.06/
```

进入 Hi3520D/Hi3515A/Hi3515C u-boot 源码目录

```
make ARCH=arm CROSS_COMPILE=arm-hisiv100nptl-linux- hi3520d_config  
make ARCH=arm CROSS_COMPILE=arm-hisiv100nptl-linux-  
chmod 777 mkboot-hi3520d.sh  
./mkboot-hi3520d.sh .reg1 .reg2 full-boot.bin  
make ARCH=arm CROSS_COMPILE=arm-hisiv100nptl-linux- mini-boot.bin
```

1.2 编译小型化版本的内核

如果要单独编译小型化版本，请使用配置文件 hi3520d_mini_defconfig。具体操作：

进入 Hi3520D/Hi3515A/Hi3515C 内核源代码目录

```
cp arch/arm/configs/ hi3520d_mini_defconfig .config  
make ARCH=arm CROSS_COMPILE=arm-hisiv100nptl-linux- menuconfig  
make ARCH=arm CROSS_COMPILE=arm-hisiv100nptl-linux- uImage
```



1.3 在 Hi3520D/Hi3515A/Hi3515C 发布包中制作小型化版本的文件系统

- 创建根文件系统根目录下的子目录和文件

进入 osdrv 目录，执行如下操作：

```
tar xzf rootfs_scripts/rootfs.tgz -C pub/
```

在根文件系统中创建相应的库文件

```
tar xzf toolchain/arm-hisiv100nptl-linux/runtime_lib/runtime_lib/  
armv7a_soft/lib.uClibc.tgz -C pub/rootfs
```

- 编译 busybox

进入 osdrv 目录，执行如下操作：

```
tar xzf busybox/busybox-1.16.1.tgz -C busybox  
find busybox/busybox-1.16.1 | xargs touch  
cp busybox/busybox-1.16.1/ busybox_cfg_hi3520d_nptl_mini  
busybox/busybox-1.16.1/.config  
make  
make -C busybox/busybox-1.16.1 install  
cp -af busybox/busybox-1.16.1/_install/* pub/rootfs
```

- 制作根文件系统镜像

进入 osdrv 目录，执行如下操作：

```
cd pub/bin/pc
```

制作 squashfs 镜像

```
./mksquashfs pub/rootfs pub/rootfs_hi3520d_256k.squashfs -b 256K -  
comp xz
```

裁减前大小：

- u-boot 的大小为 157KB
- 内核的大小为 3.1MB
- 文件系统大小为 4.1MB

裁减后大小：

- u-boot 的大小为 76KB
- 内核的大小为 1.5MB
- 文件系统大小为 1.1MB



注意

如果要在 osdrv 目录下编译整套小型化版本，具体操作如下：

进入 Hi3520D/Hi3515A/Hi3515C osdrv 目录

```
make OSDRV_CROSS=arm-hisiv100nptl-linux CHIP=hi3520d OSDRV_SIZE=mini all
```



2 Uboot、内核及文件系统裁剪的一般方法

用户如果觉得发布包中给出的小型化配置不适合自己的业务需求,可以在发布包中的全规格配置文件的基础上,按照下文给出的裁剪方法,根据自己的实际需要,自行裁剪。

2.1 uboot 裁剪

uboot 的裁剪主要从两方面下手:

1. 使用 lzma 压缩算法压缩 uboot 二进制文件本身的尺寸;



注意

在 Hi3520D/Hi3515A/Hi3515C 小型化版本中使用了 lzma 算法对 uboot 二进制文件进行压缩,需要用户在编译代码的服务器上安装 lzma 压缩工具。对应的 lzma 工具已经放在 SDK 发布包中。

2. 在配置文件中将环境变量的起始地址提前。环境变量的起始地址和大小由 SPI flash 的块大小决定。例如, SPI flash 的块大小为 64KB, u-boot 二进制文件大小不超过 128KB, 环境变量的起始地址可以设置为 0x20000, 环境变量的大小可以设置为 0x10000。



说明

由此可见,使用块大小越小的 SPI flash 器件会达到更好的小型化效果,同时考虑到性能问题,用户在使用小型化版本时最好采用块大小为 64KB 的 SPI flash 器件。

修改文件如下:

在配置文件 hi3520d.h 中,修改如下代码:

- 将宏定义#define CONFIG_ENV_OFFSET 的值由 0x80000 改为 0x20000
- 将宏定义#define CONFIG_ENV_SIZE 的值由 0x40000 改为 0x10000
- 注释 DDRT 部分

```
#define CONFIG_DDR_TRAINING_V200
```



```
#ifdef CONFIG_DDR_TRAINING_V200  
  
#define DDRT_ENABLE_BYTE_TRAINING  
  
#define DDRT_ENABLE_BIT_TRAINING  
  
#endif
```



注意

由于 uboot 小型化的解压算法需要在 DDR 初始化后运行，而 uboot 的 DDRT 程序需要在 DDR 初始化前运行。如果想要实现 uboot 小型化，就必须将 DDRT 的程序加入到小型化的解压算法中，由于 DDRT 程序涉及大量的 uboot 标准接口，在编译 uboot 小型化算法时就需要包含这些接口对应的文件，导致生成的 uboot 小型化镜像很大，并没有达到小型化的效果。因此，如果在 uboot 中需要实现 DDRT，就不能使用小型化的压缩算法。

2.2 内核裁减

内核的裁减主要有两个思路：一是通过控制内核的编译过程，让更多的代码编译进内核，以减少内核目标二进制文件的体积；二是通过使用更高压缩率的压缩方法，让内核镜像文件更小。

内核源代码中除了系统运行所必须的核心代码之外，还包含了各种各样的外部设备驱动、文件系统，以及一些跟某种特性相关的代码。而这些代码，在我们的业务环境中，并不都是必须的。我们将这些不必要的内容，通过配置选项进行筛选。

进入内核源代码目录，运行如下命令，打开内核配置菜单，并对菜单上的选项进行配置：

```
$ cp arch/arm/configs/hi3520d_full_defconfig .config  
$ make ARCH=arm CROSS_COMPILE=arm-hisiv100nptl-linux- menuconfig
```

2.2.1 一些目前还处在开发或者完善过程中的模块

这部分代码还没有通过全面测试，稳定性无法保证，用户也很少会使用，在系统的资源有限的情况下，除非有明确的需求，否则都可以去掉。

```
General setup --->  
[ ] Prompt for development and/or incomplete code/drivers
```

去掉这个选项，那些目前还甚少被使用的、处在开发完善阶段的代码就不会被编译进内核，也不会在内核配置菜单中出现。关闭这个选项，可能减少 85K（gzip）空间。

```
Device Drivers --->  
[ ] Staging drivers ---->
```



这个选项提供了一些内核编制以外的驱动，这部分代码放在这里，更多原因是因为想吸引更多用户去了解和用它。但这些代码没有经过广泛测试，相关接口未来可能还会改变。可以关闭这个选项。

2.2.2 一些特殊功能和特性的支持

- **POSIX 标准的异步 I/O 操作支持**

这主要看用户是否使用了 `aio_read` 之类的函数。POSIX 标准的异步 I/O 有别于 linux 原生的 I/O 接口。linux 原生代码认为，io 的读操作是同步的，写操作是异步的。也就是说读取某 io 的数据，必须等待数据由 io 传送到 buffer 之后才读完成返回。在有些情况下，更高性能的做法是希望读操作发起之后，线程立即返回，去做其他与 buffer 无关的事情。直到 buffer 数据准备好，才处理读结果。于是就有了 aio 接口支持。用户通常都只是使用 linux 原生的 I/O 接口，可以将该选项关闭。

```
General setup --->
[ ] Enable AIO support
```

- **系统的 extended profiling(剖面)的支持**

profiling(剖面图)是一个工具来扫描、统计和测评计算机性能的工具。对多数用户而言，可以关闭该选项。

```
General setup --->
[ ] Profiling support
```

- **用户空间的 thumb 二进制代码的支持**

可以关闭。

```
System Type --->
[ ] Support Thumb user binaries
```

- **Disk quotas 支持**

用于支持在多用户系统下，设置每个用户对硬盘的使用空间。可以关闭。

```
File systems --->
[ ] Quota support
```

- **支持将 panic 和 oops 消息存放到 flash 分区中的循环 buffer 中**

可以关闭。

```
Device Drivers --->
<*> Memory Technology Device (MTD) support --->
< > Log panic/oops to an MTD buffer
```

2.2.3 网络支持

几乎所有的产品，都需要支持网络模块。但并不是都需要支持网络模块中的所有功能。比如无线网络部分，用户就可以根据实际需要，选择支持或者不支持。

去掉网络模块的支持：

关闭对 IEEE802.11 协议公共类库的支持和对 Linux wireless LAN 配置 API 的支持。

```
[*] Networking support --->
```



```
[ ] Wireless --->
```

关闭所有 IEEE802.11 协议相关的无线网络设备驱动。

```
Device Drivers --->
```

```
[*] Network device support --->
```

```
[ ] Wireless LAN --->
```

如果要恢复对无线网络的支持，除了网络相关的基础配置必须选择之外，请按照操作顺序，将如下选项配置上。

```
[*] Networking support --->
```

```
[*] Wireless --->
```

```
<*> cfg80211 - wireless configuration API
```

```
<*> Common routines for IEEE802.11 drivers
```

```
Device Drivers --->
```

```
[*] Network device support --->
```

```
[*] Wireless LAN --->
```

```
<*> IEEE 802.11 for Host AP (Prism2/2.5/3 and WEP/TKIP/CCMP)
```

2.2.4 设备驱动支持

Loopback device 支持

该选项支持把一个普通文件当作块设备文件使用。

```
Device Drivers --->
```

```
Block devices --->
```

```
Loopback device support
```

输入设备驱动相关的支持

可以不选该选项。

```
Device Drivers --->
```

```
Input device support --->
```

```
Hardware I/O ports --->
```

```
< > Serial I/O support
```

```
< > Gameport support
```

多媒体设备的支持

可以不选该选项。

```
Device Drivers --->
```

```
< > Multimedia support --->
```

```
[ ] Backlight & LCD device support --->
```

usb 相关的驱动

```
Device Drivers --->
```



```
[*] USB support --->
< > Enable Wireless USB extensions (EXPERIMENTAL)
```

usb host wifi 的支持，可以不选。

usb 具体的设备驱动，可以根据需要进行筛选。参考配置如下：

```
Device Drivers --->
[*] USB support --->
    *** USB Device Class drivers ***
    < > USB Modem (CDC ACM) support
    < > USB Printer support
    < > USB Wireless Device Management support
    < > USB Test and Measurement Class support
    *** NOTE: USB_STORAGE depends on SCSI but BLK_DEV_SD
may ***
    *** also be needed; see USB_STORAGE Help for more
info ***
    < > Realtek Card Reader support
    < > Support for Rio Karma music player
    < > SAT emulation on Cypress USB/ATA Bridge with ATACB
    < > USB ENE card reader support
    < > USB Attached SCSI
    [ ] The shared table of common (or usual) storage
devices
    *** USB Imaging devices ***
    < > USB Mustek MDC800 Digital Camera support
    < > Microtek X6USB scanner support
    *** USB port drivers ***
    < > USB Serial Converter support --->
    *** USB Miscellaneous drivers ***
    < > EMI 6|2m USB Audio interface support
    < > EMI 2|6 USB Audio interface support
    < > ADU devices from Ontrak Control Systems
    < > USB 7-Segment LED Display
    < > USB Diamond Rio500 support
    < > USB Lego Infrared Tower support
    < > USB LCD driver support
    < > USB LED driver support
    < > Cypress CY7C63xxx USB driver support
    < > Cypress USB thermometer driver support
    < > Siemens ID USB Mouse Fingerprint sensor support
    < > Elan PCMCIA CardBus Adapter USB Client
    < > Apple Cinema Display support
    < > USB 2.0 SVGA dongle support (Net2280/SiS315)
```



```
< > USB LD driver
< > PlayStation 2 Trance Vibrator driver support
< > IO Warrior driver support
< > USB testing driver
< > iSight firmware loading support
< > USB YUREX driver support
< > USB Gadget Support --->
    *** OTG and related infrastructure ***
[ ] Generic ULPI Transceiver Driver
< > NOP USB Transceiver Driver
```

2.2.5 文件系统类型支持

linux 内核中提供了对许多种文件系统的支持。实际应用中，许多文件系统都无需直接编译进内核，除非你明确需要使用某种文件系统，你再把它选择上来。

Ext2 文件系统的支持

默认情况下，内核会自动把它选上。把下面选项关闭，可以去掉对 Ext2 的支持。

```
File systems --->
< > Second extended fs support
```

如果你确实使用使用 Ext2，再把它选上。

Ext3 文件系统的支持

内核在默认情况下，通常都会支持 Ext3。关闭下面选项，可以去掉对 Ext3 的支持。

```
File systems --->
< > Ext3 journalling file system support
```

Ext4 文件系统的支持

内核在默认情况下，通常都会支持 Ext4。关闭下面选项，可以去掉对 Ext4 的支持。

```
File systems --->
< > The Extended 4 (ext4) filesystem
```

XFS 文件系统的支持

内核在默认情况下，通常都会支持 XFS。关闭下面选项，可以去掉对 XFS 的支持。

```
File systems --->
< > XFS filesystem support
```

Journalling Flash File System v2 文件系统(JFFS2)



多数情况下, 我们都需要支持 JFFS2, 只有在明确不需要 JFFS2 的情况下, 才关闭下面选项。

```
File systems --->
[*] Miscellaneous filesystems --->
< > Journalling Flash File System v2 (JFFS2) support
```

Compressed ROM file system(即 Cramfs)

Cramfs 文件系统是专门针对闪存设计的只读压缩的文件系统,其容量上限为 256M,采用 zlib 压缩。Cramfs 以压缩方式存储,在运行时解压缩。所有的应用程序要求被拷到 RAM 里去运行, Cramfs 采用分页压缩的方式存放文件,在读取文件时,不会一下子就耗用过多的内存空间,只针对目前实际读取的部分分配内存,尚没有读取的部分不分配内存空间,当我们读取的内容不在内存时, Cramfs 文件系统自动计算压缩后的资料所存的位置,再即时解压缩到 RAM 中。Cramfs 的速度快,效率高,其只读的特点有利于保护文件系统免受破坏,提高了系统的可靠性。由于以上特性, Cramfs 在嵌入式系统中应用广泛。但是它的只读属性同时又是它的一大缺陷,使得用户无法对其内容对进扩充。

Cramfs 的使用和 initrd/initial ram filesystem 密切相关,如果系统不使用 Cramfs,可以同时关闭 initrd/initial ram filesystem 选项:

不支持 cramfs:

```
File systems --->
-*- Miscellaneous filesystems --->
< > Compressed ROM file system support (cramfs)
```

SquashFS

SquashFS 是另一种可用于 flash 设备的 Linux 只读文件系统。squashfs 具有极高的压缩率,数据(data)、节点(inode)和目录(directories)都被压缩。常用于存储介质非常有限的场景。

squashfs 保存了全部的 32 位 UID/GIDS 和文件的创建时间,支持多达 4G 的文件系统, squashfs 使用简单,响应速度快。

默认情况下,内核关闭对 squashfs 的支持。如果选择 SquashFS 作为系统的根文件系统,则应该打开下面选项,增加支持 squashfs 的支持。

```
File systems --->
[*] Miscellaneous filesystems --->
<*> SquashFS 4.0 - Squashed file system support
```

2.2.6 内核镜像文件的压缩方式

linux 内核编译生成二进制文件之后,需要通过某种压缩模式,将庞大的二进制文件压缩成体积更小的镜像文件,以便存放在空间有限的存储介质中;在系统启动时,会首先将压缩的镜像文件解压到 RAM 中,然后再从头执行,完成系统启动。对此, linux 3.0.y 内核提供了多种压缩方式: Gzip, LZMA 和 LZO。而这几种压缩模式,在压缩率和解压速度之间各有千秋。



Gzip: linux 内核镜像默认的、也是最经典的压缩模式。它在压缩率和解压速度上，保持了最佳的平衡。

LZMA: LZMA 是 linux 内核新近才支持的压缩模式，相比另外两种压缩模式，它具有最高的压缩率（同样文件，通过 LZMA 压缩后的体积通常只有 Gzip 的 70%），但是压缩和解压缩的速度要差一些。适用于 spi flash 大小非常有限的场合中。

LZO: 这种压缩方式压缩率最低，但是压缩和解压的速度最快。（目前该算法在 3.0 内核中还不完善，使用该压缩算法会有风险，所以目前并不使用）

具体的选择方法（以选择 LZMA 模式为例）：

```
General setup --->
Kernel compression mode (LZMA) --->
  ( ) Gzip
    (X) LZMA
  ( ) LZO
```



注意

编译内核镜像使用的服务器需要支持 LZMA 压缩算法。如果没有，请将下面提供的 LZMA 压缩算法的源码压缩包拷贝到服务器上，并且在 root 权限下将该源码包解压、编译和安装，命令如下：

```
tar -xzf lzma-4.32.7.tgz
cd lzma-4.32.7
./configure
make install
```



lzma-4.32.7.tgz

2.2.7 打印和调试信息

linux 内核中有不少和系统的调试信息相关内容，这部分内容在系统调试定位的时候非常重要，但也占用了一定的空间。在存储资源极度缺乏的环境中，也可以考虑将他们去掉。关闭它们不影响系统的正常运行。

linux 内核的 debug 文件系统

这是一个虚拟的文件系统，用于存放内核开发者使用的 debug 文件。

```
Kernel hacking --->
[ ] Debug Filesystem
```




内核跟踪器

```
Kernel hacking --->
[ ] Tracers --->
```

用户态出错信息

用户态程序出错后崩溃的时候，内核会打印一句简短的信息，告知出错的具体原因。该信息在应用程序调试阶段非常有用。关闭将不打印出错信息。

```
Kernel hacking --->
[ ] Verbose user fault messages
```

2.3 文件系统裁剪

文件系统的裁减，可以从如下三方面着手：一是通过配置 **busybox**，将不需要的功能、命令裁去；二是将文件系统中的可执行文件和库中多余的调试信息、符号信息删除掉，以减少文件系统的容量；三是采用更高压缩率的文件系统。其中，第二种方式简单，而且非常有效。

2.3.1 busybox 配置选项

打开 **busybox** 的配置选项菜单

```
$ cp [busy_cfg_file] .config
$ make menuconfig
```

其中 **busy_cfg_file** 为具体产品 **busybox** 默认的配置文件。

打开后的 **busybox** 配置菜单如下：

Busybox Settings --->	/* 关于 busybox 的基础配置，下面会详细介绍 */
--- Applets	
Archival Utilities --->	/* 与压缩解压文件相关的功能，请根据具体需要选择 */
Coreutils --->	/* busybox 核心命令集，请根据实际需求选择 */
Console Utilities --->	/* 控制台相关的命令 */
Debian Utilities --->	/* Debian 系统相关的功能，基本可以不选 */
Editors --->	/* 编辑器相关的功能，请根据需要选择 */
Finding Utilities --->	/* 与查找相关的功能，请根据需要选择 */
Init Utilities --->	/* 与系统启动相关的配置，必须保留 */



Login/Password Management Utilities ---> /* 登陆和用户密码的管理 */

Linux Ext2 FS Progs ---> /* 与 Ext2 文件系统相关的命令 */

Linux Module Utilities ---> /* 模块加载和卸载的命令 */

Linux System Utilities ---> /* linux 系统中各大模块的支持 */

Miscellaneous Utilities ---> /* 未分类的功能支持 */

Networking Utilities ---> /* 网络相关的支持，请根据实际需求选择 */

Print Utilities ---> /* 打印机相关的支持，如无特殊需求，可放心关闭 */

Mail Utilities ---> /* 和电子邮件相关的支持，如无特殊需求，可放心关闭 */

Process Utilities ---> /* 进程相关的功能，请根据实际需求，谨慎配置 */

Runit Utilities ---> /* 与系统服务相关的支持，可根据实际需求配置 */

Shells ---> /* 各种 shell 解释器配置 */

System Logging Utilities ---> /* 各种记录、日志相关的支持 */

Load an Alternate Configuration File

Save Configuration to an Alternate File

多数选项需要根据实际的业务需求进行筛选。这里仅对 busybox 的基础配置中一些可能裁减的内容进行说明。

```
Busybox Settings --->
General Configuration --->
[ ] Show verbose applet usage messages
```

支持输入命令 +[--help]，以显示该命令更详细的使用说明。
不选可以节省 13 K 空间。

可以不选。

```
[ ] Store applet usage messages in compressed form
```

以压缩的方式保存命令使用说明的信息。

不必选。

```
[ ] Support --install [-s] to install applet links at runtime
```

本命令支持在 busybox 运行之后，可以通过以创建新符号的方式生成命令。

可以不选。



```
Build Options --->
Debugging Options --->
Installation Options --->
Busybox Library Tuning --->
[ ] Support for /etc/networks
```

支持在 `route` 命令中使用网络名字，而不仅仅是 `ip` 等。

很少用，可以不选。

```
[ ] vi-style line editing commands
```

`vi` 的样式设置命令，可以不选。

```
[ ] Fancy shell prompts
```

上面这两个选项还会影响到命令行开头的提示，如果选上了，`shell` 命令行提示符可能会显示：

```
john@bvt-bsp:~/workspace$
```

否则只会显示如下提示符：

```
$
```

可以不选。

```
[ ] Query cursor position from terminal
```

从终端中查询光标的位置，可以不选。

```
[ ] Use clock_gettime(CLOCK_MONOTONIC) syscall
```

如果选上，`time`, `ping`, `traceroute` 等命令调用的将是系统调用 `clock_gettime`;

如果不选，调用的是 `gettimeofday`，可能会不准确。

可以不选。

```
[ ] Use ioctl names rather than hex values in error messages
```

如果选上使能，`ioctl` 中的错误信息将使用 `ioctl` 命令名称，否则使用二进制数。产品版本可以不选上，这样可节省 1K 空间。

其它的选项，除了必须选择的（比如 `Init Utilities`）之外，基本都和应用需求相关，这里就不再一一说明。

2.3.2 删除文件系统的可执行文件、相关的库文件中多余的调式信息和符号信息（即 `strip elf`）

值得注意的是，文件系统的 `*.ko` 文件是不能 `strip` 的，否则 `ko` 文件不可用。

```
find rootfs/ -perm +700 ! -name "*.ko" -exec arm-hisiv100nptl-linux-strip {} \;
```



上面命令可以将根目录下所有的可执行文件、库文件一次 `strip` 完毕。

2.3.3 使用 squashfs 根文件系统

当要创建一个很小嵌入式 linux 系统时, 存储设备(如软盘, FLASH 等)的每个字节都十分重要, 所以必须尽可能压缩每个可能的地方。squashfs 将这些实现带到了新的高度。Squashfs 具有如下特点:

- 数据(data), 节点(inode)和目录(directories)都被压缩;
- 保存了全部的 32 位 UID/GIDS 和文件的创建时间(注: cramfs 是 8 位, 没有创建时间);
- 支持多达 4G 的文件系统(cramfs 是 16M);
- 重复的文件会被检测并删除掉;
- 同时支持 big endian 和 little endian 架构, 可支持将文件系统 mount 到不同的字节顺序(byte-order)的机器上面。

如何制作 squashfs 文件系统?

squashfs 文件系统具有比 jffs2 文件系统更高的压缩率。linux-3.0.y 内核支持 squashfs 文件系统, 但在默认配置中, 支持 squashfs 选项是关闭的。因此, 需要先打开相关的配置选项。

```
File systems --->
[*] Miscellaneous filesystems --->
<*> SquashFS 4.0 - Squashed file system support
[*] Include support for XZ compressed file systems
```

默认情况下 mksquashfs 工具制作的文件系统镜像采用 gzip 算法压缩, mksquashfs 还支持 lzma/xz 压缩算法。相同大小的文件系统, 采用 xz 压缩算法的镜像体积约为 gzip 压缩算法的 4/5。要使 mksquashfs 支持 lzma/xz 压缩算法, 在编译 mksquashfs 工具的时候, 需要打开支持 xz 压缩算法的选项。另外考虑到多数服务器都不支持 lzma 库, 在 mksquashfs 源码包中, 需要增加一个 zlib 库。源码包见附件 squashfs4.2.+xz.tgz。



squashfs4.2.+xz
.tgz

具体操作如下:

```
tar -xvf squashfs4.2.+xz.tgz
cd squashfs4.2/
make
```

在当前目录下生成的 mksquashfs 即为支持 xz 压缩算法的制作文件系统工具。使用方法如下:

```
./mksquashfs rootfs ./ rootfs.squashfs.img -b 256K -comp xz
```

其中, rootfs 是之前已经制作好的根文件系统, rootfs.squashfs.img 是生成的 squashfs 文件系统映像文件。-b 256K 指定 squashfs 文件系统的块大小为 256K。-comp xz 制定文件系统压缩方式为 xz。



2.3.4 剔除没有被使用的库文件

文件系统的裁减，除了以上方式外，在所有开发工作都已经完成的阶段，还可以使用最后一招：找出文件系统中没有被使用的库文件，并删除之。

下面的脚本可以帮我们找出哪些库文件从没被使用过：

```
$vi libsave.sh
```

```
#!/bin/bash
    find rootfs/ -perm +700 -exec arm-hisivl00nptl-linux-readelf -d {}
\;>log1.txt
grep ".so" log1.txt | sort | uniq >log2.txt
```

执行该脚本后，当前目录下会生成./log2.txt 文件，在./log2.txt 中列出的，就是从没被其它程序使用的库文件，可以删除。