Hi3520D/Hi3515A/Hi3515C

System Tailoring Guide

Issue   01

Date   2013-07-31

HiSilicon Technologies Co., Ltd.

Address: Huawei Industrial Base

            Bantian, Longgang

            Shenzhen 518129

            People's Republic of China

Website: http://www.hisilicon.com

Email: support@hisilicon.com

About This Document

Purpose

The Hi3520D/Hi3515A/Hi3515C kernel and file system can be tailored to minimize the system when the system services and performance are ensured. Based on Hi3520D/Hi3515A/Hi3515C service requirements, a set of common configuration files are provided in the Hi3520D/Hi3515A/Hi3515C software development kit (SDK) for minimizing the system. This document describes how to compile small-sized images by using these configuration files. You can tailor complete configuration files as required. Therefore, this document also describes how to tailor the Hi3520D/Hi3515A/Hi3515C kernel and file system.

Related Versions

The following table lists the product versions related to this document.

Product Name	Version
Hi3520D	V100
Hi3515A	V100
Hi3515C	V100

Change History

Changes between document issues are cumulative. Therefore, the latest document issue contains all changes made in previous issues.

Issue 01 (2013-07-31)

This issue is the fourth draft release, which incorporates the following changes:

Chapter 1 Compiling Small-Sized Images of the Kernel and File System for the Hi3520D/Hi3515A

In section 1.1, the descriptions are updated.

Chapter 2 Tailoring the U-boot, Kernel, and File System

In section 2.1, the descriptions of tailoring the U-boot are updated.

Issue 00B03 (2013-06-21)

This issue is the third draft release, which incorporates the following changes:

The descriptions of the Hi3515C are added.

Issue 00B02 (2013-05-09)

This issue is the second draft release, which incorporates the following changes:

Chapter 1 Compiling Small-Sized Images of the Kernel and File System for the Hi3520D/Hi3515A

In section 1.1, the **commands** are updated.

Issue 00B01(2013-04-03)

This issue is the first draft release.

Contents

1 Compiling Small-Sized Images of the Kernel and File System for the Hi3520D/Hi3515A/Hi3515C

1.1 Compiling a Small-Sized U-boot Image

Use the modified configuration file **hi3520d.h** to separately compile a small-sized U-boot image (for details about how to modify **hi3520d.h**, see section 2.1 "Tailoring the U-boot"). See the following:

```
cp
tools/pc_tools/uboot_tools/reg_info_Hi3520D-bvt_No1_660_330_660_ddr_inner
FEPHY.bin uboot/u-boot-2010.06/.reg1

cp
tools/pc_tools/uboot_tools/reg_info_Hi3515A-bvt_No1_600_300_600_ddr_inner
FEPHY.bin uboot/u-boot-2010.06/.reg2

cp tools/pc_tools/uboot_tools/mkboot-hi3520d.sh  uboot/u-boot-2010.06/
```

Go to the U-boot source code directory of the Hi3520D/Hi3515A/Hi3515C, and run the following commands:

```
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 Compiling a Small-Sized Kernel Image

Use the configuration file **hi3520d_mini_defconfig** to compile only a small-sized kernel. To be specific, go to the directory of the Hi3520D/Hi3515A/Hi3515C kernel source code, and run the following commands:

```
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 Creating a Small-Sized File System Image in the Hi3520D/Hi3515A/Hi3515C SDK

1.3.1 Creating Subdirectories and Files under the Directory of the Root File System

Go to the **osdrv** directory and run the following command:

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

Create library files in the root file system:

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

1.3.2 Compiling the BusyBox

Go to the **osdrv** directory and run the following commands:

```
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
```

1.3.3 Creating the Root File System Image

Go to the **osdrv** directory and run the following command:

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

To create the Squashfs image, run the following command:

```
./mksquashfs pub/rootfs pub/rootfs_ hi3520d_256k.squashfs -b 256K
-comp xz
```

After tailoring, the U-boot size decreases from 157 KB to 76 KB, the kernel size decreases from 3.1 MB to 1.5 MB, and the file system size decreases from 4.1 MB to 1.1 MB.

📖 **NOTE**

To compile a set of small-sized images including the U-boot, kernel, and file system in the **osdrv** directory, go to the Hi3520D/Hi3515A **osdrv** directory, and run **make OSDRV_CROSS=arm-hisiv100nptl-linux CHIP=hi3520d OSDRV_SIZE=mini all**.

2 Tailoring the U-boot, Kernel, and File System

If the configuration files for minimizing the system do not meet your service requirements, you can tailor the complete configuration files provided in the SDK by following the methods described in this chapter.

2.1 Tailoring the U-boot

Tailoring the U-boot involves:

- Compressing the binary U-boot file by using the lzma compression algorithm.

⚠ CAUTION

Before compressing the binary U-boot file by using the lzma compression algorithm in the tailored Hi3520D/Hi3515A/Hi3515C system, you must install the lzma compression tool on the server where the code is compiled. The lzma compression tool is provided in the SDK.

- Setting the start address for environment variables ahead in the configuration file. The block size of the SPI flash determines the start address and size of environment variables. For example, if the block size of the SPI flash is 64 KB and the size of the binary U-boot file is less than or equal to 128 KB, you can set the start address for environment variables to **0x20000** and environment variable size to **0x10000**.

📖 **NOTE**

> If an SPI flash with a smaller block size is used, the system size is also smaller. You are advised to use an SPI flash with 64 KB block size to minimize the system size while ensuring system performance.

Modify the configuration file **hi3520d.h** as follows:

- Change the value of **#define CONFIG_ENV_OFFSET** from **0x80000** to **0x20000**.
- Change the value of **#define CONFIG_ENV_SIZE** from **0x40000** to **0x10000**.
- Comment out the DDRT part.

```
#define CONFIG_DDR_TRAINING_V200

#ifdef CONFIG_DDR_TRAINING_V200
```

```
#define DDRT_ENABLE_BYTE_TRAINING

#define DDRT_ENABLE_BIT_TRAINING

#endif
```

CAUTION

The decompression algorithm for tailoring the U-boot can be used only after the DDR is initialized, whereas the DDRT program of the U-boot must run before the DDR is initialized. If you want to tailor the U-boot, you must add the DDRT program to the decompression algorithm. Because the DDRT program involves lots of standard U-boot interfaces, the files corresponding to these interfaces are required when the decompression algorithm is compiled. As a result, the generated U-boot image is large, and the U-boot is not minimized. Therefore, you cannot use the decompression algorithm for tailoring the U-boot if you want to run the DDRT program under the U-boot.

2.2 Tailoring the Kernel

The kernel can be tailored in the following methods:

- Controlling the compilation process and compiling less code into the kernel to reduce the size of target binary files in the kernel.
- Using a compression method with higher compression rate to reduce the size of images in the kernel.

In addition to core code required for the system running, kernel source code also include the code for various external device drivers, file systems, and certain features. The code is not necessary in the service environment. Therefore, the code is filtered by using configuration options.

Go to the directory of the kernel source code, and run following command to open the kernel configuration menu and configure options on the menu:

```
$ cp arch/arm/configs/hi3520d _full_defconfig .config

$ make ARCH=arm CROSS_COMPILE=arm-hisiv100nptl-linux- menuconfig
```

2.2.1 Modules Under Development or Improvement

The code has not been comprehensively tested. The stability cannot be ensured. Users seldom use the code. Therefore, when system resources are limited, the code can be removed unless otherwise specified.

```
General setup  --->

[ ] Prompt for development and/or incomplete code/drivers
```

Deselect the option. Seldom used and under-development code is not compiled into the kernel and do not appear on the kernel configuration menu. When the option is deselected, 85 KB (Gzip) space may be reduced.

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

```
[ ] Staging drivers --->
```

This option provides drivers beyond the kernel range. The code is stored here for more potential customers. However, the code is not comprehensively tested, and related interfaces may change in future. Therefore, this option can be deselected.

2.2.2 Special Functions and Features

2.2.2.1 Asynchronous I/O Operations in Compliance with POSIX

The AIO depends on the use of the aio_read function. The asynchronous input/output (AIO) in the Portable Operating System Interface Standard (POSIX) is different from the original AIO in the Linux. For Linux original code, the read operation over input/output interfaces is synchronous whereas the write operation is asynchronous. That is, to read data over an input/output interface, users must wait until the data is transferred from the input/output interface to the buffer.

In certain circumstances, better performance is expected. That is, data can be sent back by the thread immediately after the read operation is initiated, and operations independent from the buffer can be complete. When the buffer data is ready, read results are processed. AIO support is introduced for the circumstances. Typically, only original input/output interfaces are used in the Linux. Therefore, this option can be deselected by using the following option:

```
General setup --->
```

```
[ ] Enable AIO support
```

2.2.2.2 Extended Profiling

Profiling is used to scan, collect, and assess computer performance. For most users, this option can be deselected using the following option:

```
General setup --->
```

```
[ ] Profiling support
```

2.2.2.3 Thumb Binary Code of User Space

This option can be deselected using the following option:

```
System Type --->
```

```
[ ] Support Thumb user binaries
```

2.2.2.4 Disk Quotas

This option is used to set the hard disk space for each user in a multi-user system, and can be deselected using the following option:

```
File systems --->
```

```
[ ] Quota support
```

2.2.2.5 Panic and Oops Message Storage to the Cyclic Buffer in the Flash Partition

This option can be deselected using the following option:

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

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

2.2.3 Network Support

Almost all products must support the network module, but all functions of the module do not need to be supported. For example, users can select whether to support the wireless network as required.

Disable the support of the network module.

- Disable the support of public databases of the IEEE802.11 protocol and the support of application programming interfaces (APIs) configured on the Linux wireless LAN using the following option:

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

- Disable all wireless network device drivers related to the IEEE802.11 protocol using the following option:

```
Device Drivers  --->
[*] Network device support  --->
[ ]   Wireless LAN  --->
```

- To support wireless networks again, enable the basic configurations related to networks, and configure the following options in sequence:

```
[*] 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 Device Drivers

2.2.4.1 Loopback Device

This option supports the use of a common file as a block device file.

```
Device Drivers  --->
Block devices  --->
Loopback device support
```

2.2.4.2 Input Device Driver

This option can be deselected.

```
Device Drivers  --->
Input device support  --->
```

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

            < > Serial I/O support

            < > Gameport support
```

2.2.4.3 Multimedia Device

This option can be deselected.

```
Device Drivers --->

Graphics support ->

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

2.2.4.4 USB Drivers

```
Device Drivers --->

[*] USB support --->

< >   Enable Wireless USB extensions (EXPERIMENTAL)
```

The support of USB host Wi-Fi can be disabled.

```
Device Drivers --->

[*] USB support --->

< >   USB Mass Storage support
```

The support of USB flash drives can be disabled.

Select the USB driver as required. The following configurations are for reference:

```
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 Supported File System Types

The Linux kernel supports multiple types of file systems. Actually, many file systems may not be compiled into the kernel, unless otherwise specified.

2.2.5.1 Ext2 File System

The kernel supports the file system by default. To disable the Ext2 support, deselect the following option:

```
File systems  --->

< > Second extended fs support

To use the Ext2 file system, select the option.
```

2.2.5.2 Ext3 File System

The kernel supports the file system by default. To disable the Ext3 support, deselect the following option:

```
File systems  --->

< > Ext3 journaling file system support
```

2.2.5.3 Ext4 File System

The kernel supports the file system by default. To disable the Ext4 support, deselect the following option:

```
File systems  --->

< > The Extended 4 (ext4) filesystem
```

2.2.5.4 XFS

The kernel supports the file system by default. To disable the XFS support, deselect the following option:

```
File systems  --->

< > XFS filesystem support
```

2.2.5.5 JFFS2

In most cases, the Journaling Flash File System v2 (JFFS2) must be supported. Deselect the following option only when JFFS2 is not required:

```
File systems  --->

[*] Miscellaneous filesystems  --->

< >  Journaling Flash File System v2 (JFFS2) support
```

2.2.5.6 Cramfs

The Compressed ROM file system (cramfs) is a read-only compressed file system designed for flash memories. The upper capacity bound is 256 MB and the compression method is zlib. Cramfs stores data in compression mode and decompresses data during running. All applications must be copied to the RAM for running. Cramfs compresses files and stores them by page, and does not occupy too much RAM space when reading files. Cramfs allocates RAM only for contents to be read currently. If the contents to be read are not stored in the RAM, cramfs automatically calculates the location where the compressed material is stored and decompresses the material to the RAM.

Cramfs features high speed and high efficiency. In addition, files are read-only in cramfs. This prevents the file system from being corrupted and improves system reliability. With these features, cramfs is widely used in embedded systems. However, the read-only attribute is also a great disadvantage, because users cannot extend contents in the system.

The use of cramfs is closely related to the initrd/initial ram filesystem. If the system does not use cramfs, the initrd/initial ram filesystem option can be deselected at the same time.

Cramfs is not supported:

```
File systems  --->

-*- Miscellaneous filesystems  --->

< >   Compressed ROM file system support (cramfs)
```

2.2.5.7 Squashfs

Squashfs is another type of Linux read-only file system used for flash devices. Squashfs features high compression rate and can compress data, inodes, and directories. Squashfs is applicable to scenarios where storage media are extremely limited.

Squashfs stores all 32 bits of user ID (UID) and group IDs (GIDs), and file creation time. Squashfs supports up to 4 GB file system, and is easy to use and rapid in response.

The kernel does not support squashfs by default. If squashfs is selected as the root file system, enable the following option to support squashfs:

```
File systems  --->

[*] Miscellaneous filesystems  --->

<*>   SquashFS 4.0 - Squashed file system support
```

2.2.6 Methods of Compressing Kernel Images

After the Linux kernel is compiled to generate binary files, the files need to be compressed as smaller images, so that files can be stored in limited storage media. When the system starts, images are decompressed to the RAM and executed from the beginning. The Linux 3.0.y kernel provides multiple compression methods at various compression rates and decompression speeds: Gzip, LZMA, and LZO.

- Gzip: The default and classical compression method used by the Linux kernel. It balances the compression rate and decompression speed.
- LZMA: A new compression method of the Linux kernel. Compared with the other two compression methods, LZMA features higher compression rate but lower compression and decompression speed. (The size of a file compressed using LZMA is only 70% of that compressed using Gzip.) This method is suitable when the size of SPI flash is small.
- LZO: A method featuring the lowest compression rate and highest compression and decompression speed. Note that this method is not complete in 3.0 kernel and insecure. Therefore, the method is not used currently.

The selection method is as follows (LZMA is used as an example):

```
General setup  --->

Kernel compression mode (LZMA)  --->

( ) Gzip
```

```
(X) LZMA

( ) LZO
```

📖 **NOTE**

The server used to compile kernel images must support the LZMA compression algorithm. If the server does not support LZMA, copy the following source code package of the LZMA compression algorithm to the server, and decompress, compile, and install the package as user **root**. The commands are as follows:

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

lzma-4.32.7.tgz

2.2.7 Printing and Debugging Information

The Linux kernel contains a large amount of information about debugging. These contents are important in system debugging and analysis, and occupy certain space. When storage resources are extremely insufficient, the debugging function can be disabled. The running of the system is not affected when the debugging function is disabled.

2.2.7.1 Debugging File System of the Linux Kernel

This virtual file system is used to store debugging files for kernel developers.

```
Kernel hacking  --->

[ ] Debug Filesystem
```

2.2.7.2 Kernel Tracer

```
Kernel hacking  --->

[ ] Tracers  --->
```

2.2.7.3 Information About User Mode Fault

When a user mode program fails and crashes, the kernel reports a short message, indicating causes of the fault. The message is helpful during application debugging. Fault messages will not be reported when the option is deselected.

```
Kernel hacking  --->

[ ] Verbose user fault messages
```

2.3 Tailoring the File System

File systems can be tailored in the following methods:

- Configuring BusyBox to disable functions and commands that are not required.

- Removing useless debugging information and symbol information from executable files and databases of the file system to reduce the file system capacity.

- Using a file system with higher compression rate.

The second method is easy and effective.

2.3.1 Configuring BusyBox Options

Open the configuration option menu of BusyBox.

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

```
$ make menuconfig
```

where **busy_cfg_file** specifies the default configuration file of the BusyBox for the specific product.

The BusyBox configuration menu is as follows:

```
    Busybox Settings  --->                   /* Basic BusyBox
configurations to be described in the following parts */

--- Applets

    Archival Utilities  --->                 /* Functions related
to compressing and decompressing files to be selected as required
*/

    Coreutils  --->                          /* Core BusyBox
command sets to be selected as required */

    Console Utilities  --->                  /* Commands related to
the console */

    Debian Utilities  --->                   /* Functions related
to the Debian system (basically not selected) */

    Editors  --->                            /* Functions related to
the editor to be selected as required */

    Finding Utilities  --->                  /* Functions related
to searching to be selected as required */

    Init Utilities  --->                     /* Configuration
related to the system start (must be reserved) */

    Login/Password Management Utilities  --->  /* Management of
login and user password */

    Linux Ext2 FS Progs  --->                /* Commands related to
the Ext2 file system */

    Linux Module Utilities  --->             /* Commands used to
load and unload modules */

    Linux System Utilities  --->             /* Support of various
modules in the Linux */

    Miscellaneous Utilities  --->            /* Unclassified
function */
```

```
     Networking Utilities  --->                /* Support related to
the network to be selected as required */

     Print Utilities  --->                     /* Support related to
the printer that can be disabled without special requirements */

     Mail Utilities  --->                      /* Support related to
email that can be disabled without special requirements */

     Process Utilities  --->                   /* Functions related
to progresses to be configured carefully as required */

     Runit Utilities  --->                     /* Support related to
system services to be configured as required */

     Shells  --->                              /* Configurations for
various shell interpreters */

     System Logging Utilities  --->            /* Support related to
various records and logs */

---

     Load an Alternate Configuration File

     Save Configuration to an Alternate File
```

Most options must be filtered based on actual service requirements. This section describes only contents that may be tailored in basic BusyBox configurations.

```
Busybox Settings  --->

General Configuration  --->
```

[ ] Show verbose applet usage messages

The command can be entered with an end of "--help" so that more detailed descriptions of the command are displayed. If the option is deselected, 13 KB space is reduced.

[ ] Store applet usage messages in compressed form

The command description is compressed and stored.

This option can be deselected.

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

The command can be generated by creating symbols when BusyBox is executed.

This option can be deselected.


```
Build Options  --->

Debugging Options  --->

Installation Options  --->

Busybox Library Tuning  --->
```

[ ] Support for /etc/networks

Network names can be used in route commands in addition to IP addresses.

The option is seldom used and can be deselected.

[ ]   vi-style line editing commands

The commands set vi modes. The option can be disabled.

[ ]   Fancy shell prompts

The preceding options determine the start of the shell command line prompt. If the options are selected, the start of the command line prompt is:

john@bvt-bsp:~/workspace$

If the options are deselected, the start of the command line prompt is:

$

This option can be deselected.

[ ]   Query cursor position from terminal

The option is used to query cursor position from the terminal, and can be deselected.

[ ] Use clock_gettime(CLOCK_MONOTONIC) syscall

If the option is selected, the time, ping, and traceroute commands invoke the value of the clock_gettime function.

If the option is deselected, the value of the gettimeofday function is invoked, and the time may be inaccurate.

This option can be deselected.

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

If the option is selected, fault information of the ioctl function reports the command name. If the option is deselected, a binary value is reported. The product version may not be reported. Then 1 KB space is reduced.

Except necessary options such as Init Utilities, other options are selected as required and are not described in the section.

2.3.2 Deleting Useless Debugging Information and Symbol Information from Executable Files and Databases of the File System

📖 NOTE

Do not strip .ko files in the file system. Otherwise, the files cannot be used.

Run the following command to strip all executable files and library files in the root directory:

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

2.3.3 Using Squashfs as the Root File System

To create a small embedded Linux system, compress every file that can be compressed, because every byte is important on storage devices such as floppy disks and flash memories. Squashfs implements the achievements. The following are squashfs features

- Squashfs compresses data, inodes, and directories.

- Squashfs stores all 32 bits of UID and GIDS, and file creation time. (Cramfs stores 8 bits of UID and GIDS, and does not store creation time.)

- Squashfs supports a maximum of 4 GB file systems. (Cramfs supports 16 MB file systems.)

- Squashfs detects and deletes repeated files.

- Squashfs supports big endian and little endian architectures at the same time and supports mounting file systems to computers with different byte-order.

The following describes how to create a squashfs file system.

The compression rate of the squashfs file system is higher than that of the jffs2 file system. The linux-3.0.y kernel supports the squashfs file system. Because the squashfs options are deselected by default, you need to select these options as follows:

```
File systems  --->

[*] Miscellaneous filesystems  --->

    <*>   SquashFS 4.0 - Squashed file system support

    [*]    Include support for XZ compressed file systems
```

The file system images created by using mksquashfs are compressed in compliance with the Gzip algorithm by default. Mksquashfs can also support the LZMA and XZ compression algorithms. For the same file system image, the size of the image compressed by using the XZ algorithm is 4/5 of the size of the image compressed by using the Gzip algorithm. To enable mksquashfs to support the LZMA and XZ compression algorithms, select the XZ compression algorithm option when compiling mksquashfs. As most servers do not support the LZMA library, add a zlib library to the source code package of mksquashfs attached.

squashfs4.2.+xz
    .tgz

Run the following commands to add the zlib library:

```
tar –xvf squashfs4.2.+xz.tgz

cd squashfs4.2/

make
```

The mksquashfs generated in the current directory is the file system creation tool that supports the XZ compression algorithm. To use mksquashfs, run the following command:

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

Note:

- **rootfs** is the created root file system.

- **rootfs.squashfs.img** is the image of the generated squashfs file system.

- **-b 256K** indicates that the specified block size of the squashfs file system is 256 KB.

- **–comp xz** indicates that the algorithm for compressing the file system is XZ.

2.4 Deleting Unused Library Files

After all files are tailored and development is complete, you can search for and delete unused library files.

To search for unused library files, run the following scripts:

```
$vi libsave.sh
```

```
#!/bin/bash
```

```
find rootfs/ -perm +700 -exec arm-hisiv100nptl-linux-readelf -d {} \;>log1.txt
```

```
grep ".so" log1.txt | sort | uniq >log2.txt
```

After the preceding scripts are executed, the **./log2.txt** file is generated in the current directory. The library files listed in the **./log2.txt** file can be deleted because they are not used by any program.