



Tina Linux 系统裁剪 开发指南

版本号: 1.4
发布日期: 2022.02.25

版本历史

版本号	日期	制/修订人	内容描述
1.0	2019.02.14	AWA0916	first version
1.1	2020.07.22	AWA0916	更换格式
1.2	2021.04.20	AWA0916	删除不确定的内核选项说明
1.3	2022.02.17	AWA0985	增加 C 库切换说明
1.4	2022.02.25	AWA1742	增加内核压缩选项说明



目 录

1 概述	1
1.1 编写目的	1
1.2 适用范围	1
1.3 相关人员	1
2 Tina 系统裁剪简介	2
2.1 boot0 裁剪	2
2.2 uboot 裁剪	2
2.3 内核裁剪	3
2.3.1 删除不使用的功能	3
2.3.2 删除不使用的驱动	4
2.3.3 修改内核源代码	5
2.3.3.1 size 工具	5
2.3.3.2 ksize.py 脚本	5
2.3.3.3 nm 命令	8
2.3.3.4 kernel 压缩方式	9
2.4 文件系统裁剪	9
2.4.1 应用程序及冗余文件裁剪	9
2.4.2 库的裁剪	10
2.4.2.1 C 库的选择	10
2.4.2.2 删除没用到的库	10
2.4.3 应用程序与库 strip	11
2.4.4 文件系统压缩	11
3 参考资料	13

1 概述

1.1 编写目的

嵌入式产品往往为了压缩成本而使用较小的 flash 存储器，因此可能需要对系统进行裁剪来减少对 flash 的占用。系统经过裁剪过后，通常也会提升启动速度以及减少内存占用。

本文介绍 TinaLinux 中系统裁剪的方法，为有裁剪需求的使用者提供参考。

1.2 适用范围

适用于基于硬件平台：全志 R/MR/V/H 系列芯片。

软件平台：Tina V3.5 及其后续版本。

1.3 相关人员

适用于 TinaLinux 平台的客户及相关技术人员。

2 Tina 系统裁剪简介

Tina 固件中通常包含 boot0、uboot、kernel、rootfs 等镜像。基于经验，各个镜像尺寸的量级如下表所示：

表 2-1: 各镜像尺寸的量级

镜像	大小
boot0	< 100K
uboot	< 1M
kernel	$\geq 3\text{M}$, < 15M
rootfs	$\geq 4\text{M}$

可以看到 boot0、uboot、kernel、rootfs 的尺寸是依次增大的。对于大尺寸的裁剪效果往往比小尺寸的裁剪效果明显，比如 rootfs 裁剪 1M 可能很容易，对于 uboot 来说，则非常困难。

因此，后续主要介绍 kernel 以及 rootfs 的裁剪。

2.1 boot0 裁剪

由于 boot0 很小，通常来说 boot0 代码也不开源，因此略过。

2.2 uboot 裁剪

目前 tina 环境中有不同版本的 uboot，分别存在两个不同的文件路径中，以实际 sdk 的目录为准，cboot 可以进入到相应的 uoot 目录，这两个路径分别为lichee/brandy/u-boot*或者lichee/brandy-2.0/u-boot*

主要有下面两种裁剪思路：

- 修改 uboot 配置文件，删减不需要的配置。uboot 配置文件通常位于源码下include/configs/\${CHIP}.h或者configs/\${CHIP}*_defconfig。
- 删除不需要的 uboot 命令。

2.3 内核裁剪

通常关于 Linux 内核裁剪主要有如下方法：

- 删除不使用的功能。如符号表、打印、调试等功能。
- 删除不使用的驱动。
- 修改内核源代码。
- 内核压缩。

2.3.1 删除不使用的功能

下表中列出了一些内核选项，包含选项的描述，默认值以及推荐值（减小内核镜像尺寸）。

表 2-2: 内核选项及描述

CONFIG option	Description	Def	Small
CORE_SMALL	tune some kernel data sizes	N	Y
NET_SMALL	tune some net-related data sizes	N	Y
KMALLOC_ACCOUNTING	turn on kmalloc accounting	N	Y *
AUDIT_BOOTMEM	print out all bootmem allocations	N	Y *
DEPRECATE_INLINES	cause compiler to emit info about inlines	N	Y *
PRINTK	printk code and message data	Y	N
BUG	allow elimination of BUG code	Y	N
ELF_CORE	allow disabling of ELF core dumps	Y	N
PROC_KCORE	allow disabling of /proc/kcore	Y	N
AIO	allow disabling of async IO syscalls	Y	N
XATTR	allow disabling of xattr syscalls	Y	N
FILE_LOCKING	allow disabling of file locking syscalls	Y	N
DIRECTIO	allow disabling of direct IO support	Y	N
MAX_SWAPFILES_SHIFT	number of swapfiles	5	0
NR_LDISCS	number of tty line disciplines	16	2
MAX_USER_RT_PRIO	number of RT priority levels	100	5
KALLSYMS	load all symbols for debugging/kksymoops	Y	N
SHMEM	allow disabling of shmem filesystem	Y	N +
SWAP	support for a swap segment	Y	N
SYSV_IPC	support for System V IPC	Y	N +
POSIX_MQUEUE	POSIX message queue support	Y	N +
SYSCTL	allow disabling of sysctl support	Y	N +
LOG_BUF_SHIFT	control size of kernel printk buffer	14	11
CC_OPTIMIZE_FOR_SIZE	Use gcc -os to optimize for size	Y	Y
MODULES	allow support for kernel loadable modules	Y	N +

CONFIG option	Description	Def	Small
KMOD	automatic kernel module loading	Y	N
PCI	allow support for PCI bus and devices	Y	Y -
XIP_KERNEL	allow support for kernel Execute-in-Place	N	N
BLK_DEV_LOOP	support for loopback block device	Y	Y -
BLK_DEV_RAM	block devices for RAM filesystems	Y	Y -
IOSCHED_AS	Include Anticipatory IO scheduler	Y	Y
IOSCHED_DEADLINE	Include Deadline IO scheduler	Y	N +
IOSCHED_CFQ	Include CFQ IO scheduler	Y	N +
IP_PNP	support for IP autoconfiguration	Y	N +
IP_PNP_DHCP	support for IP autoconfiguration via DHCP	Y	N +
IDE	support for IDE devices	Y	N +
SCSI	support for SCSI devices	Y	N +

其中：

- "Y *" - 表示开发的时候设置成 Y，发布的时候可以设置成 N。
- "N +" - 表示基于应用需要来判断是否设置成 N。
- "Y -" - 表示可能需要，可以设置 N 尝试一下。

在 Tina 中，集成了 CONFIG_REDUCE_KERNEL_SIZE 宏。一旦使能该宏后，将会采用部分上面的裁剪措施来减小 kernel 镜像尺寸，主要思路是关闭与 log/debug 等相关的配置，然后对 kernel 进行 xz 压缩，可参考 tina/scripts/reduce-kernel-size.sh。

执行 make menuconfig，开启如下选项：

```
Tina Configuration
Target Images --->
[*] downsize the kernel size (EXPERIMENTAL)
```

说明

此功能当前是 **EXPERIMENTAL** 的。建议直接执行 **make kernel_menuconfig**，然后按照上述表格来配置。

2.3.2 删除不使用的驱动

方案明确之后，所需的内核驱动也明确了。可以执行 make kernel_menuconfig，将没有用到的驱动关闭。

2.3.3 修改内核源代码

内核源码庞大，直接修改往往难度很大，可借助相关工具来评估模块以及符号的大小，然后进行针对性的裁剪。

2.3.3.1 size 工具

size 命令可查看内核镜像的 text、data、bss 等段的大小。如执行"size vmlinux"，将会得到：

text	data	bss	dec	hex	filename
5818117	1378944	168972	7366033	706591	vmlinux

2.3.3.2 ksize.py 脚本

在 tina/lichee/linux-4.9/scripts 目录下有一个 ksize 脚本，可以对内核目录下的 built-in.o 进行解析，并将解析的内容按照尺寸进行排序，显示出来。执行结果如下所示：

Linux Kernel	total	text	data	bss
vmlinux	7366033	5818117	1378944	168972
drivers/built-in.o	2244823	2080782	123885	40156
net/built-in.o	1682005	1630911	32590	18504
fs/built-in.o	975830	950780	5442	19608
kernel/built-in.o	678363	593347	41064	43952
mm/built-in.o	302442	272965	7309	22168
sound/built-in.o	237890	227338	6836	3716
security/built-in.o	170272	145055	13989	11228
block/built-in.o	149110	145408	2458	1244
crypto/built-in.o	145972	131610	7258	7104
lib/built-in.o	141721	141093	559	69
init/built-in.o	33551	18558	14909	84
ipc/built-in.o	29998	29218	772	8
usr/built-in.o	138	138	0	0
sum	6792115	6367203	257071	167841
delta	573918	-549086	1121873	1131
drivers	total	text	data	bss
drivers/built-in.o	2244823	2080782	123885	40156
drivers/usb/built-in.o	448756	409279	27813	11664
drivers/block/built-in.o	357202	324752	21582	10868
drivers/tty/built-in.o	174213	155371	13938	4904
drivers/base/built-in.o	157961	153861	3460	640
drivers/mmc/built-in.o	133678	131782	1756	140
drivers/scsi/built-in.o	105021	95105	9348	568
drivers/md/built-in.o	100909	98382	1291	1236
drivers/mtd/built-in.o	96023	92244	1467	2312
drivers/hid/built-in.o	86072	81552	4160	360

drivers/clk/built-in.o	69737	58289	10856	592
drivers/cpufreq/built-in.o	51525	44400	1793	5332
drivers/pinctrl/built-in.o	50463	46458	3921	84
drivers/input/built-in.o	45250	44046	1156	48
drivers/i2c/built-in.o	43511	42791	656	64
drivers/spi/built-in.o	39888	38323	1557	8
drivers/thermal/built-in.o	38654	36673	1893	88
drivers/regulator/built-in.o	36217	35257	820	140
drivers/of/built-in.o	35994	35095	407	492
drivers/gpio/built-in.o	29432	29167	224	41
drivers/leds/built-in.o	25548	25076	464	8
drivers/rtc/built-in.o	25072	24428	460	184
drivers/tee/built-in.o	24823	24662	113	48
drivers/char/built-in.o	23718	21642	1224	852
drivers/soc/built-in.o	20916	13980	6804	132
drivers/bluetooth/built-in.o	20223	19319	100	804
drivers/dma/built-in.o	17892	17458	334	100
drivers/irqchip/built-in.o	14767	12371	2308	88
drivers/pwm/built-in.o	14636	14036	472	128
drivers/dma-buf/built-in.o	13975	13904	23	48
drivers/cpuidle/built-in.o	12613	10848	1749	16
drivers/watchdog/built-in.o	9986	9660	281	45
drivers/power/built-in.o	9836	8296	1224	316
drivers/clocksource/built-in.o	9608	8708	796	104
drivers/misc/built-in.o	8471	8105	340	26
drivers/bus/built-in.o	6357	5691	618	48
drivers/hwmon/built-in.o	5230	5054	144	32
drivers/hwspinlock/built-in.o	4792	4664	128	0
drivers/firmware/built-in.o	4453	4384	9	60
drivers/reset/built-in.o	3818	3686	132	0
drivers/net/built-in.o	1803	1755	48	0
drivers/mfd/built-in.o	1623	1511	108	4
drivers/video/built-in.o	379	379	0	0
<hr/>				
sum	2381045	2212444	125977	42624
delta	-136222	-131662	-2092	-2468
<hr/>				
net	total	text	data	bss
<hr/>				
net/built-in.o	1682005	1630911	32590	18504
<hr/>				
net/ipv4/built-in.o	428233	401161	14719	12353
net/mac80211/built-in.o	302085	301822	259	4
net/core/built-in.o	267334	256693	8573	2068
net/bluetooth/built-in.o	227913	226708	1033	172
net/wireless/built-in.o	160236	158651	557	1028
net/xfrm/built-in.o	74537	72737	1384	416
net/bridge/built-in.o	59936	58812	1112	12
net/sched/built-in.o	29706	28344	1346	16
net/packet/built-in.o	26453	26172	281	0
net/netlink/built-in.o	26105	25498	455	152
net/unix/built-in.o	24671	22266	340	2065
net/key/built-in.o	21095	20783	308	4
net/*.o	16279	15811	412	56
net/8021q/built-in.o	15245	14981	264	0
net/ipv6/built-in.o	9445	8295	1136	14
net/rfkill/built-in.o	7142	6710	408	24
net/ethernet/built-in.o	2431	2391	40	0
net/llc/built-in.o	2068	1980	72	16

net/802/built-in.o	1944	1792	140	12
sum	1702858	1651607	32839	18412
delta	-20853	-20696	-249	92
fs	total	text	data	bss
fs/built-in.o	975830	950780	5442	19608
fs/*.o	351665	339511	1774	10380
fs/ext4/built-in.o	295807	294110	1125	572
fs/jffs2/built-in.o	99642	99446	124	72
fs/proc/built-in.o	77696	73111	377	4208
fs/fat/built-in.o	49264	49088	144	32
fs/jbd2/built-in.o	47379	47254	65	60
fs/overlayfs/built-in.o	24939	24842	93	4
fs/squashfs/built-in.o	23156	23092	60	4
fs/kernfs/built-in.o	21086	16863	111	4112
fs/configfs/built-in.o	18193	17916	261	16
fs/debugfs/built-in.o	16120	16056	52	12
fs/pstore/built-in.o	13904	13531	325	48
fs/crypto/built-in.o	13083	12799	264	20
fs/notify/built-in.o	12525	12186	227	112
fs/nls/built-in.o	11024	10904	116	4
fs/sysfs/built-in.o	7041	6990	39	12
fs/devpts/built-in.o	3359	2986	365	8
fs/ramfs/built-in.o	1820	1776	40	4
sum	1087703	1062461	5562	19680
delta	-111873	-111681	-120	-72
kernel	total	text	data	bss
kernel/built-in.o	678363	593347	41064	43952
kernel/*.o	376931	346029	17531	13371
kernel/sched/built-in.o	127386	119841	6265	1280
kernel/time/built-in.o	101386	89633	7465	4288
kernel/printk/built-in.o	55033	18477	8444	28112
kernel/irq/built-in.o	47323	44325	906	2092
kernel/rcu/built-in.o	29815	27655	2131	29
kernel/locking/built-in.o	25617	25592	21	4
kernel/power/built-in.o	16652	15256	848	548
kernel/bpf/built-in.o	8348	7988	68	292
sum	788491	694796	43679	50016
delta	-110128	-101449	-2615	-6064
sound	total	text	data	bss
sound/built-in.o	237890	227338	6836	3716
sound/soc/built-in.o	125556	119220	5416	920
sound/core/built-in.o	108923	104799	1400	2724
sound/*.o	6612	6440	28	144
sum	241091	230459	6844	3788

delta	-3201		-3121	-8	-72
security	total		text	data	bss
security/built-in.o	170272		145055	13989	11228
security/selinux/built-in.o	142606		119156	12250	11200
security/keys/built-in.o	31690		30618	788	284
security/*.o	25826		24091	1719	16
security/integrity/built-in.o	1838		1806	20	12
sum	201960		175671	14777	11512
delta	-31688		-30616	-788	-284
block	total		text	data	bss
block/built-in.o	149110		145408	2458	1244
block/*.o	145968		142021	2699	1248
block/partitions/built-in.o	7563		7543	16	4
sum	153531		149564	2715	1252
delta	-4421		-4156	-257	-8
lib	total		text	data	bss
lib/built-in.o	141721		141093	559	69
lib/*.o	216133		214935	1092	106
lib/zlib_inflate/built-in.o	11187		11187	0	0
lib/xz/built-in.o	8215		8179	36	0
lib/lzo/built-in.o	2551		2551	0	0
lib/lz4/built-in.o	1188		1188	0	0
sum	239274		238040	1128	106
delta	-97553		-96947	-569	-37

可以对各个模块的代码段数据段的统计信息进行确认，对占用空间大的进行针对性优化。

2.3.3.3 nm 命令

nm 命令可查看内核模块中各个符号的尺寸。如执行"nm --size -r vmlinux | head -10"，可得到：

```
00004000 b __log_buf
00003e58 D nand_tbl
00003b14 T __blockdev_direct_IO
0000398c T hidinput_connect
00002f6c t ext4_fill_super
000027fc T hci_event_packet
0000245c t l2cap_recv_frame
000023d4 T dev_ethtool
00002274 t test_atomics
000020e4 t nl80211_send_wiphy
```

说明，一共有三列数据，分别表示大小、符号类型、符号名。其中符号类型：

- b/B - 符号位于 bss 段。
- t/T - 符号位于 text 段。
- d/D - 符号位于 data 段。

如果某些函数或者全局变量占用较大，可以进行针对性的优化。

2.3.3.4 kernel 压缩方式

tina 环境提供了几种压缩格式方式，选择方式为 make kernel_menuconfig:

```
General setup --->
  Kernel compression mode (Gzip) --->
    (X) Gzip
    ( ) LZMA
    ( ) XZ
    ( ) LZ0
    ( ) LZ4
```

同一个 kernel 镜像使用不同的压缩方式，镜像大小如下表所示：

压缩方式	镜像大小	加载内核时间（从 falsh 加载到 dram 的时间）	解压时间	总时间
GZIP	2.31M	124ms	89ms	213ms
LZO	2.53M	136ms	23ms	159ms
LZ4	2.68M	143ms	27ms	170ms
XZ	1.95M	104ms	667ms	771ms

2.4 文件系统裁剪

对于文件系统裁剪来说，主要思路是删、换、压。

- 删。删除不需要的内容。如帮助文档、没用到的库、调试程序等。
- 换。使用小尺寸的实现替换大尺寸的实现。如使用 musl libc 库替换 glibc 库等。
- 压。使用合适的压缩算法。

2.4.1 应用程序及冗余文件裁剪

在不影响整体功能的情况下，一些应用程序或冗余文件往往可以删除：

- 调试工具。比如 tcpdump、mpstat、strace 等等。
- 性能测试工具。比如 lmbench、sysstat、tiobench 等等。
- 冗余文件。帮助文档、辅助程序、配置文件和数据模块等，又比如很多应用有相同的共能，只留其一。
- 采用具有通用功能的替代软件包。Linux 上有许多具有相似功能的软件包，可以选择其中占存储空间较小的软件包并移植到嵌入式设备上。
- 资源文件。一些音视频以及 UI 资源往往占用很大空间，如果没有用到，也需要删除。

2.4.2 库的裁剪

关于库的裁剪主要有两个思路：

- 使用较小的 C 库，如 musl libc，uclibc 等来替换 glibc。
- 删除没有用到的库。

2.4.2.1 C 库的选择

下表列出了当前一些通用的 C 库及其特征。

表 2-4: 常用 C 库及其特征

C 库	环境	大小	优点	缺点
glibc	Distribution	大	强大稳定，支持最多的 cpu 架构	占用空间大
uclibc	Embedded	小	为嵌入式设计，可配置性好	不支持 libdb 与 libnss
bionic	Android	小	提供了 Android 特性的函数	不提供 libthread_db/libm
musl	Embedded	小	更小，高效静态链接，稳定	支持较少的 cpu arch

当前 Tina 环境下可支持 glibc 与 musl libc 两种 C 库。具体可通过 menuconfig 的方式来配置使用哪一套。

```
[*] Advanced configuration options (for developers) --->
    Select external toolchain C library (Use glibc) --->
```

注意，更换 C 库后，需要清除方案编译产物（可执行 make distclean），重新编译。

2.4.2.2 删除没用到的库

嵌入式产品通常应用程序有限，因此可能存在很多库不会被用到，可以进行删除。

当前 Tina 环境提供了一种删除方法，执行 make menuconfig，打开如下选项：

```
Tina Configuration
Target Images --->
[*] downsize the root filesystem or initramfs
```

打开之后，在生成 rootfs/initramfs 之前会对其中没有用到的库进行删除。

具体可参考 scripts/reduce-rootfs-size.sh 文件，其主要思路是：

- 分析 rootfs 下的应用程序所依赖的库。
- 分析“应用程序依赖库”所依赖的库，一直递归下去，直到完全找出所有依赖的库。
- 根据上述查找结果，删除没有被依赖的库。

说明

此方法有一定的限制：

- 当前只分析 `/lib`，`/usr/lib` 下的库，其他目录不会处理。
- 对于部分使用 `dlopen` 的应用程序，解析库可能会出现问题。

2.4.3 应用程序与库 strip

strip 会去掉应用程序与库的符号信息和调试信息，大大减少空间占用。

当前 Tina 环境下默认开启了 strip 功能，如果没开启，请确保开启以减少空间占用。

```
Tina Configuration
Global build settings --->
Binary stripping method (strip) --->
```

说明

注：Tina 上还支持 **sstrip**，即 **super strip**，相对于 **strip** 来说，更能减少应用与库的大小。

2.4.4 文件系统压缩

有些文件系统支持压缩，有些不支持。下表列出了常见的文件系统类型：

表 2-5: 常用文件系统类型

FS	使用	压缩	读写	备注
ext2	block device	无	RW	突然断电或当机时可能导致数据丢失
ext3	block device	无	RW	向前兼容 ext3，日志式文件系统，非常成熟稳定
ext4	block device	无	RW	向前兼容 ext2 和 ext3，扩展存储限制，提升性能
btrfs	block device	有	RW	着重于容错、修复及易管理
FAT	block device	无	RW	Windows，长期使用速度变慢，不支持 >4G 文件
NTFS	block device	有	RW	Windows，基于 FAT 做若干改进，日志文件系统
Cramfs	NAND Flash	无	RO	2013 停用，使用 Squashfs

FS	使用	压缩	读写	备注
Squashfs	Raw Flash	有	RO	压缩度更高，没有大小限制
UBIFS	Raw Flash	有	RW	基于 JFFS2，Linux3.7 之后
JFFS2	Raw Flash	有	RW	mount 时间很慢，读写性能不好
YAFFS2	NAND Flash	无	RW	没有透明压缩，不在 Linux 主线

当前 Tina 环境下比较常用的是 squashfs、ext4、jffs2 三种文件系统。具体可执行 make menuconfig 进行选择：

```
Tina Configuration
Target Images --->
  *** Root filesystem images ***
  [ ] ext4  ----
  [ ] jffs2
  [*] squashfs --->
```

常见的压缩有 lzop, gzip, xz 等，压缩率最高的是 xz。但是 xz 压缩解压最慢，非常影响启动速度。实际在选择压缩方式时应综合考虑。

3 参考资料

[1] https://elinux.org/Kernel_Size_Tuning_Guide

[2] Karim Yaghmour. Building Embedded Linux Systems [M]

[3] Michael Opdenacker. Embedded Linux size reduction techniques

[4] <https://tiny.wiki.kernel.org/>



著作权声明

版权所有 © 2022 珠海全志科技股份有限公司。保留一切权利。

本文档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本文档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

商标声明

、 全志科技 （不完全列举）均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标，产品名称，和服务名称，均由其各自所有人拥有。

免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的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所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。