

W.F.F.F. High Hell of Interesors

设度深圳精神相似证

D1 Tina Linux 存储

开发指南

A THE WALL OF THE PROPERTY OF

Light Hangogs

##Kithitakidos

发布日期: 2020.04.05

以保持挨批批



X/R/E/KHIKHAHAMOG88X

文档密级: 秘密

版本历史

(A	LLWIMER		a Mangly	a mango s	文档密级: 和
	FIN		版本历5	ŧ "	
A STATE OF THE STA	版本号	日期	制/修订人	内容描述	
A-SHITTER OF THE SHIP OF THE S	1.0	2020.04.05	AWA1046	初始版本	
\sim		~		~	γ

Wifted With the Constant of th XKREKHIKHA MINITURE NA FERTIFIE HALL BURG

White Hillippe





录 目

ALLWIMERS STATES	- TOTAL BOOK BENT		文档密约	及: 秘密
A PARTY OF THE PAR	W ^V ■	录	V	A A A A A A A A A A A A A A A A A A A
1 概述	a Kilifix	a fight.		
1.1 编写目的 1.2 适用范围				7. A.
1.2 适用范围 1.3 相关人员				1
				2
2 分区管理 2.1 分区配置文件	<u>-</u>			2 2
2.1 分区配置 X III				2
2.3 常见分区及其				3
2.4 分区大小与对				3
2.5 分区与文件系	· 统			4
2.6 分区资源镜像	R文件		· Walge	4
	squashfs 镜像	الي	7	5
2.6.2 创建、				6
2.6.3 创建 ((*/_\strain_1)			6
2.6.4 创建 1	10-			7 %)
◇ 2.7 总容量说明 .	%			~/
3 系统挂载		1 M		10
3.1 块设备节点 .				10
3.2 挂载点		N.		11
	圭载设备目录			11
	圭载点			
-	下的挂载			
3.3.1 fstab	编写格式			12
3.3.2 globa	al 尖型 config			12
N/4	nt 类型 config ,,	N/AL	*	WYL.
, X4	**	, X.4-		1.4 5
5.4.1 /工总=	事项	·····································		16
4 文件系统支持情况	K-19	XIR-Y		16
4.1 ext4 与日志				16
	的日志			
	大小与日志			
4.1.3 修复 (ext4			18
5 UBI (spi) Nano	d			19
=	Į			20
	odatevol			
11-	ock	1.		1
5.1.4 其他				21
	XXXXX	A THE STATE OF THE		A KANA
	#E+175C± @ T4 ½ A + 1714 107 /			
XXX-	版权所有 © 珠海全志科技股份	刀有收公可。沐笛之功仪利		XI.



6 rootfs data 及 UDISK **22** 22 22 7 关键数据保护 24 25 25 7.2.4 在 uboot 中读写 7.2.5 在用户空间读写

IN THE REAL PROPERTY.

7.3 secure storage 区域与 private 分区比较.

NA FEBRUARY

文档密级: 秘密

XAA Kilikili —

* Hilly To Ingroogset

LANGE TO THE PARTY OF THE PARTY

版权所有 © 珠海全志科技股份有限公司。保留一切权利

jii



表	格

	2-1 2-2	west Seat		2088H				COSSA		
	ALLA	WEARS		18C12				SU ₁₂	文档密级	3: 秘密
	THIS IV		THE IV	表	格		A THE LAND OF THE PARTY OF THE			
	2-1	分区属性 ※	JEXXIV.			i kiliki	<i>\$</i> 4.			2
1	2-2	常见分区和用途	,							2
X's	2-3	对齐规则				XII.				×13
	2-4	常见分区的文件系统 .								4
	2-5	创建系统镜像命令								5
	3-1	不同介质对应的节点 .								10
	3-2	设备名与分区关系								10
	3-3	默认挂载点								11
	3-4	global 配置项								13
	3-5	config 配置项								13
	3-6	挂载属性		. Cooper				1908		14
	4-1	文件系统支持情况	· · · · · _@//	· · · ·			· · · · · · · · · · · · · · · · · · ·			16
	4-2	ext 日志模式					JEN V			17
	5-1	UBI 子系统层次结构	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX			×	\$ · · · ·	(a)		19
	5-2	UBI 方案物理分区 . 💉	j			C. X. XII.				19
XX	'	XIR. M.				X.B.	Et			X/R



文档密级: 秘密

概述

1.1 编写目的

介绍 TinaLinux Flash,分区,文件系统等存储相关信息,指导方案的开发定制。

从是提供推推推開起調的吸吸

J 相关人员 适用于 TinaLinux 平台的客户及相关技术人员。



2

分区管理

2.1 分区配置文件

在全志平台中,通过 sys_partition.fex 文件配置分区。在 Tina 中,可以在 **lunch** 选择方案后,通过命令 **cconfigs** 快速跳转到分区配置目录,通常情况下,其路径如下。

tina/device/config/chips/<芯片编号>/configs/<方案名>/sys_partition.fex

2.2 分区配置格式

以 rootfs 分区为例:

[partition]
 name = rootfs
 size = 25200
 downloadfile = "rootfs.fex"
 user_type = 0x8000

每个分区以 [partition] 标识,分区属性及其意义如下表。

表 2-1: 分区属性

	D. V		Ø. /
属性	含义	必选	备注
name	分区名	Y	
size	分区大小	Y	单位: 扇区 (512B), 注 1
downloadfile	分区烧入的镜像文件	N	注 2
verify	量产后校验标识	N	(默认)1: 使能; 0: 禁用,注 3
user_type	分区属性	N	注 4
keydata	量产时是否擦除本分区	N	0x8000: 使能; 其他无效

📙 说明

- 1. 最后一个分区 (UDISK),不设置 size,表示分配所有剩余空间。
- 2. downloadfile 支持绝对路径和相对路径,相对于tina/out/<方案名>/image。
- 3. verify 决定是否校验 downloadfile 中指定的镜像,若为 ext4 稀疏镜像,务必禁用。
- 4. 历史遗留,目前只对 UBI 方案有效。bit0为 1 时,表示创建静态卷,反之为动态卷。

创建 downloadfile 的资源镜像包看章节分区资源镜像文件。



[partition] 标识用户空间的逻辑分区,在 UBI 方案中,表现为 UBI 卷。此外,在sys_partiton. fex中存在特殊的配置 **MBR**,用于配置 MBR 空间大小。例如:

[mbr] size = 252

MBR 分区以 Kbyte 为单位,其大小也必须满足对齐原则。



△ 警告

一般情况下,不建议用户修改 mbr 分区的大小。

2.3 常见分区及其用途

表 2-2: 常见分区和用途

	(V) Y	~/·3	
分区名	用途	大小	备注
boot	内核镜像分区	比实际镜像等大或稍大即可	
rootfs	根文件系统镜像	比实际镜像等大或稍大即可	
recovery	恢复系统镜像	根据需求配置	参考 OTA 文档
private	存储 SN、MAC 等数据	使用默认大小即可	量产时默认不丢失
env	存放 Uboot 使用的数据	使用默认大小即可	
rootfs_data	根目录覆盖分区	根据需求配置	注 1
UDISK	用户数据分区	不需要配置大小	注 2
- 0,5		0.5	0.7

□ 说明

- 1. rootfs data 分区通过 overlayfs 覆盖根文件系统,以支持 squashfs 根文件系统的可写,此时对根文件系统写入的数 据实际是保存到 rootfs_data 分区,因此 rootfs_data 分区的容量标识着根文件系统最大可写数据量。
- 2. UDISK 作为最后一个分区,不需要设置 size,表示分配剩余所有空间给 UDISK。

2.4 分区大小与对齐

分区大小的对齐要求与不同介质、不同存储方案相关。不按对齐要求配置,可能出现文件系统异 常,分区边界数据丢失等现象。对齐规则如下表。

表 2-3: 对齐规则

介质	对齐大小		备注	80884
(ubi) spinand	2 × 物理块 -	2 × 质	注 1	_B/Rall
sdcard	16M	KIN TO THE PARTY OF THE PARTY O	原则上只须扇区对齐,	实际应用常以 16M 对齐



☑ 说明

1. 在常见的 128M Spi Nand 中,为 252K 对齐。

如果分区使用 ubifs 文件系统、分区最小为 5M ,否则大概率提示空间不够。

分区大小、文件系统大小、文件大小更多内容,请参考总容量说明。

2.5 分区与文件系统

常见的分区与文件系统对应关系如下表

表 2-4: 常见分区的文件系统

200		~00	_		200
分区名	默认文件系统	文件系	系统特性	备注	W. Carlotte
rootfs	squashfs	压缩、	只读	为了安全,	根文件系统建议只读
rootfs_data	ext4/ubifs	可写		注 1	RXXY ®
UIDSK	ext4/ubifs	可写		注 1 6	40
boot	none			裸数据分区	
private	vfat			注 2	
misc	none			裸数据分区	<u> </u>
env	none		4	裸数据分区	₹
pstore	pstore	. 1	M	转存奔溃E	1志
		_			

🔰 说明

- 1. 可写的分区, UBI 方案为 ubifs; sdcard 为 ext4。
- 2. private 默认为裸数据,使用 dragonSN 工具烧录后会成为 vfat 文件系统。

只读文件系统推荐使用 sugashfs。

可写文件系统,UBI 方案推荐 ubifs, sdcard 推荐 ext4。

更多文件系统信息,请参考文件系统支持情况。

2.6 分区资源镜像文件

在sys partition.fex中通过 downloadfile 指定需要烧录到分区的资源镜像文件。

大多数情况下,资源镜像文件都构建在文件系统上,通过某些命令实现把系统需要的文件,例如 音频文件、视频文件等资源,打包成一个带文件系统的镜像包,并在烧录时把镜像包烧写入存储 介质。

创建不同文件系统镜像的命令不一样,常见有以下几种:

版权所有 © 珠海全志科技股份有限公司。保留一切权利





表 2-5: 创建系统镜像命令

χ,	文件系统	创建镜像命令
	vfat	mkfs.vfat
	ext4	make_ext4fs
	ubifs	mkfs.ubifs
S	quashfs	mksquashfs4

为了最大程度利用空间,一般会使文件系统等于物理分区大小,即创建文件系统时使用分区表划定的分区大小来创建。

如果不希望硬编码大小,则可在打包时从分区表获得大小,再传给文件系统创建工具,具体的实现可以参考 tina/scripts/pack_img.sh 中的 make_data_res() 和 make_user_res() 等函数。

2.6.1 创建 squashfs 镜像

生成 squashfs 的命令,可参考编译过程的 log 得到,或者在网上搜索 squashfs 生成方式。

例如在 scripts/pack img.sh 中定义一个函数

找个地方调用下即可。

这里不用传入分区表的原因是,制作 squashfs 不需要指定文件系统大小,只读的文件系统大小完全取决于文件内容。

文档密级: 秘密



2.6.2 创建 vfat 镜像

mkfs.vfat <输出镜像> -C <文件系统大小> mcopy -s -v -i <输出镜像> <资源文件所在文件夹>/* ::

可参考 pack img.sh(在其中搜索 mkfs.vfat 找到相关代码)。

2.6.3 创建 ext4 镜像

使用 tina/out/host/bin/make_ext4fs 创建 ext4 镜像,推荐的使用方法如下:

make_ext4fs -l <文件系统大小> -b <块大小> -m、0 -j <日志块个数> <输出的镜像保存路径> <资源文件所在文件 夹>

其中,

- -m 0: 表示不需要要为 root 保留空间。
- -j < 日志块个数 >: 日志总大小为 块大小*日志块个数。

例如:

make_ext4fs -l 20m -b 1024 -m 0 -j 1024 \${R00T_DIR}/img/data.fex \${FILE_PATH}

如果空间不够大,会显示类似如下的错误日志:

\$./bin/make_ext4fs -l 10m -b 1024 -m 0 -j 1024 data.fex ./bin

Creating filesystem with parameters:

Size: 10485760

Block size: 1024

Blocks per group: 8192

Inodes per group: 1280

Inode size: 256

Journal blocks: 1024

Label:

Blocks: 10240

Block groups: 2

Reserved blocks: 0

Reserved block group size: 63

error: ext4_allocate_best_fit_partial: failed to allocate 7483 blocks, out of space?

上述错误中,资源文件达到 100+M,但是创建的镜像 -**l** 指定的大小只有 10M,导致空间不够而报错。只需要扩大镜像大小即可。

如需使用分区大小作为文件系统大小,可参考 pack_img.sh(在其中搜索 make_ext4fs 找到相关代码)。



2.6.4 创建 ubifs 镜像

使用 tina/out/host/bin/make.ubifs 创建 ubifs 镜像,推荐的使用方法如下:

```
mkfs.ubifs -x <压缩方式> -b <超级页大小> -e <逻辑擦除块大小> -c <最大逻辑擦除块个数> -r <资源文件所在文件夹> -o <输出的镜像保存路径>

压缩方式可选 none lzo zlib, 压缩率zlib > lzo > none

对常见的 128MB spinand, 1 page = 2048 bytes, 1 block = 64 page, 则
超级页大小为 2048 * 2 = 4096
逻辑擦除块大小为 2048 * 2 * 64 = 262144

最大逻辑擦除块个数,可简单设置为一个较大的值,例如 128MB / ( 2048 bytes * 2 * 64) = 512

则最终的命令为:

mkfs.ubifs -x zlib -b 4096 -e 262144 c 512 -r ${FILE_PATH} -o ${R00T_01R}/img/data_ubifs.

fex
```

2.7 总容量说明

UBI 方案中,除了必要的 mtd 物理分区之外 (boot0/uboot/pstore 等),其余空间划分到一个 mtd 物理分区。在此 mtd 物理分区中根据 sys_partition.fex 的划分构建 ubi 卷。UBI 的机制,每个块都需要预留 1~2 个页作为 EC/VID 头。因此可用容量会小于 mtd 物理分区容量。

对于 ubi 方案,整个 nand 分为若干 mtd。可使用以下命令查看

```
# cat /proc/mtd
dev: size erasesize name
mtd0: 00100000 00040000 "boot0"
mtd1: 00300000 00040000 "uboot"
mtd2: 00100000 00040000 "secure_storage"
mtd3: 00080000 00040000 "pstore"
mtd4: 07a80000 00040000 "sys"
```

如例子中的结果,整个 nand 分为 5 个 mtd。

- mtd0 存放 boot0, size 1 MB
- mtd1 存放 uboot, size 3 MB
- mtd2 存放 secure_storage, size 1 MB
- mtd3 存放 pstore, size 512 KB
- mtd4 则会进一步分为多个 ubi 卷,占用剩余所有空间

以上所有 mtd 的 size 相加,应该恰好等于 flash 总 size。

分区表中定义的每个逻辑分区,会对应 mtd sys 上的 ubi 卷。可使用以下命令查看



ubinfo -a UBI version: Count of UBI devices: UBI control device major/minor: 10:51 Present UBI devices: ubi0 ubi0 Volumes count: 12 258048 bytes, 252.0 KiB Logical eraseblock size: Total amount of logical eraseblocks: 489 (126185472 bytes, 120.3 MiB) Amount of available logical eraseblocks: 0 (0 bytes) Maximum count of volumes Count of bad physical eraseblocks: 1 Count of reserved physical eraseblocks: 19 Current maximum erase counter value: 3 4096 bytes Minimum input/output unit size: Character device major/minor: 247:0 Present volumes: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, Volume ID: 0 (on ubi0) Type: static Alignment: 1 LEBs (258048 bytes, 252.0 KiB) Size: Data bytes: 258048 bytes (252.0 KiB) State: Name: mbr Character device major/minor: 247:1 Volume ID: 1 (on ubi0) Type: dynamic Alignment: 2 LEBs (516096 bytes, 504.0 KiB) Size: State: boot-resource Character device major/minor: 247:2 ... #此处省略若干卷 Volume ID: 11 (on ubi0) dynamic Type: Alignment: 1 242 LEBs (62447616 bytes, 59.6 MiB) Size: State: Name: UDISK Character device major/minor: 247:12

如例子中的结果

- volume 0 为 mbr, 占 1 LEBs(252 KB),对应分区表本身
- volume 1 为 boot-resource, 占 2 LEBs(504 KB),对应分区表中第一个分区
- ...
- volume 11 为 UDISK, 占 242 LEBs(59.8 MB),对应分区表中最后一个分区

常见的关于容量的疑惑与解答。

4. 问:df 查看 UDISK 分区大小,明明分区有 50M,怎么显示总大小只有 40+M?

版权所有 © 珠海全志科技股份有限公司。保留一切权利

18²



答。df 显示的是文件系统的大小,文件系统本身需要额外的空间存储元数据,导致实际可用空间 会比分区大小略少。

2. 问: df 查看 squashfs 使用率总是 100%?

答:squashfs 是只读压缩文件系统,文件系统大小取决于总文件大小,使用率总是 100%,跟分 区大小无关。

🗓 说明

1. 文件大小

我们常说的文件大小,指的是文件内容有多少字节。但在一个文件系统中,空间分配以块为单位,必然会造成内部碎片。假 设块为 4K,如果文件大小为 1K,文件系统依然为其分配 4K 的块,就会造成 3K 的空间浪费。

2. 文件系统大小

文件系统大小,指的是文件系统元数据中标识的可用大小。形象来说,是 df 命令或者 statfs() 函数反馈的大小。文件系 **统大小不一定等于分区大小**,既可大于分区大小,也可小于分区大小。

3. 分区大小

在划分分区时规定的大小,往往是 sys_partition.fex 中指定的大小。





3

系统挂载

此处的自动挂载指**开机冷挂载**以及**热插拔挂载**,其中**冷挂载**指启动时挂载,**热挂载**指 TF/U 盘等插拔设备时的挂载。

3.1 块设备节点

Tina 中设备节点都在 /dev 目录下,对于不同存储介质,生成的设备节点会不一样。

表 3-1: 不同介质对应的节点

		, 1/		
存储介质	设备节点	X13-34	备注	K
nand(ubi)	/dev/ubi0_{1,2}		非块设备	点 计
TF 🖶	$\dev/mmcblk{0,1}p{1,2}$	2}	注 1	
U 盘	/dev/sda{1,2}			
SATA 硬盘	/dev/sda{1,2}			

🗓 说明

1. 若使用 mmc 做内部存储介质,由于 mmc 占用了mmcblk0的设备名,此时 TF 卡的设备名序号递增为mmcblk1,否则生成mmcblk0的设备名。因此配置 fstab 时尤其注意 TF 设备名是否正确。

对 sys_partition.fex 中设置的内部存储介质的设备节点,会自动动态在 /dev/by-name 中创建软链接。

因此,在 fstab 也可以使用/dev/by-name/xxxx的形式匹配设备。

块设备如果有分区,会形成分区设备节点,以 mmc、U 盘为例介绍设备节点名与分区的关系:

表 3-2: 设备名与分区关系

设备节点名	含义
/dev/mmcblk0	表示整个 mmc 空间,包含所有分区
/dev/mmcblk0p1	表示 mmc 中的第 1 个分区
/dev/mmcblk0p2	表示 mmc 中的第 2 个分区
/dev/sda	表示整个 U 盘,包含所有分区
/dev/sda1	表示 U 盘内的第 1 个分区
/dev/sda2	表示 U 盘内的第2个分区

版权所有 © 珠海全志科技股份有限公司。保留一切权利



热插拔块设备分区有以下特殊情况。

1. 块设备没有分区

有一些特殊的 TF 卡/ 盘没有分区,而是直接使用整个存储,表现为只有 /dev/mmcblk1和 /dev/sda ,而没有分区节点 /dev/mmcblk1p1 和 /dev/sda1 。此时需要直接挂载整个存储设备,Tina 大部分方案都支持这种特殊情况。

2. 块设备有多个分区

有一些特殊的 TF 卡/U 盘被分为多个分区,表现为存在多个 /dev/mmcblk1p $\{1,2...\}$ 和 /dev/sda $\{1,2...\}$ 。默认情况下,Tina 的 fstab 配置为只支持挂载热插拔存储设备的第一个 分区到 /mnt/SDCARD 或者 /mnt/exUDISK。

3.2 挂载点

3.2.1 默认挂载设备目录

Tina 中对常见的分区和热插拔块设备,有默认的挂载点。

表 3-3: 默认挂载点

存储介质	挂载节点	设备节点	备注
nand	/mnt/UDISK	/dev/by-name/UDISK	注 1
TF 🖶		$\/\dev/mmcblk{0,1}p1$	注 2
U 盘	/mnt/exUDISK	/dev/sda1	注 3
SATA 磁盘	/mnt/exUDISK	/dev/sda1	達3

前 说明

- 1. /dev/by-name/UDISK 为 sys partition.fex 的 UDISK 分区的软连接。
- 2. 当无分区时,默认挂载整个 (TF 卡; 当有 1 个或多个分区时,只挂载第一分区。
- 3. 当无分区时,默认挂载整个设备 (/dev/sda),当有 1 个或多个分区时,只挂载第一个分区。

3.2.2 新建挂载点

挂载文件系统需要有挂载点。

如果挂载点所在目录可写,则在挂载之前先创建目录即可。

mkdir -p xxx

若挂载点所在目录为只读,则需要在制作文件系统时提前创建好。

文档密级: 秘密



如创建非空目录,则在对应方案的 base-files 目录创建。

```
procd-init: target/allwinner/方案/base-files
```

如创建空目录,由于 git 不管理空目录,因此需在 Makefile 中动态创建,可仿照现有 Makefile 中创建 UDISK 目录的写法。

```
procd-init: package/base-files/Makefile
```

3.3 procd 启动下的挂载

procd 启动时,自动挂载由 procd、fstools、fstab 配合完成。如果需要修改冷/热挂载规则,只 需要修改 fstab 配置文件即可。

SDK 中,配置文件位于:

```
tina/target/allwinner/<方案名>/base-files/etc
```

若只是调试或临时修改挂载规则,只需要修改小机端的配置文件

```
/etc/config/fstab
```

3.3.1 fstab 编写格式

fstab 由多个 config 组成,每个 config 的基本格式示例如下:

```
config
       'xxxx
config 有 3 种类型,分别是 mount|global|swap 。Tina SDK 中没使用 swap,在本文中不做介绍。
```

3.3.2 global 类型 config

global 类型的 config 是全局配置,示例如下。

```
config 'global'
   option anon_swap
    option anon_mount
    option auto_swap
    option auto_mount
    option delay_root
```



Noption	check fs	'1'

配置项的意义如下表:

表 3-4: global 配置项

配置名称	可选值	意义
anon_mount	0/1	注 1
anon_swap	0/1	swap 使用,此处省略
auto_mount	0/1	只适用于设置热插拔是否自动挂载块设备
auto_swap	0/1	swap 使用,此处忽略
check_fs	0/1	建议配置为 1,注 2
delay_root	1,2,3	注,388

说 说明

- 1. anon_mount: 当 fstab 中无匹配要挂载设备的 uuid/label/device 属性的配置节时,是否采用默认挂载为/mnt/"\$device-name"。
- 2. check_fs: 是否在挂载前用 /usr/sbin/e2fsck 检查文件系统一致性 (只适用于 ext 系统)。
- 3. delay_root: 对应 fstab 中 target 为"/" 或"/overlay" 的设备节点不存在时,最长等待 delay_root 秒。

3.3.3 mount 类型 config

mount 类型的 config 是具体的设备挂载配置,示例如下

config 'mount'		
option starget	'/mnt/UDISK'/	*
option device	'/dev/by-name/UDISK'	20S
option options	'rw,sync'	ESC.
option enabled	'1'	

配置项的意义如下表:

表 3-5: config 配置项

配置名称	意义	备注
target	挂载点	————————————————————— 必须是绝对路径,必须有效
device	设备名	通过设备名指定待挂载的设备,注 1
uuid	设备 UUID	通过 fs 的 UUID 指定待挂载的设备,注 1
label	设备 label	通过 fs 的 label 指定待挂载的设备,注 1
enabled	是否使能	该节点是否有效 (0/1)
options	挂载属性	例如只读挂载等,注 2



□ 说明

- 1. device/uuid/label 是匹配挂载的设备,三者中至少要有一个有效。
- 2. 默认挂载支持的属性如下表:

表 3-6: 挂载属性

配置名称	意义	缺省值
ro / rw	只读 / 可读写	rw
nosuid / suid	忽略 suid/sgid 的文件属性	suid
nodev / dev	不允许/允许访问设备文件	dev
noexec / exec	不允许/允许执行程序	exec
sync / async	同步/异步写入	async
mand / nomand	允许/不允许强制锁	nomand
dirsync	同步更新文件夹	无效
noatime / atime	不更新/更新访问时间 (atime)	atime
nodiratime / diratime	不更新/更新目录访问时间 (atime)	diratime
relatime / norelatime	允许/不允许根据 ctime/mtime 更新 actime	norelatime
strictatime	禁止根据内核行为来更新 atime, 但允许用户空间修改	无效

3.4 挂载文件系统

在分区表中增加的分区默认是空分区,如需挂载成文件系统使用,则首先需要在分区中写入一个 文件系统。

方式一,在 PC 端预先生成好一个文件系统,并在分区表中指定为 download_file,则启动后可直接挂载。

例如 rootfs 分区就是在 PC 端制作好文件系统,烧录时写入 rootfs 分区。

方式二,在小机端进行格式化。

例如 UDISK 分区就是在第一次启动时,由启动脚本进行格式化。

客户可自行在某一启动脚本或应用中调用格式化工具(mkfs.xxx)进行格式化。如需参考,可仿照 UDISK 分区的格式化:

procd-init: package/base-files/files/lib/preinit/79 format partition

3.4.1 注意事项

一些格式化工具并未默认选中,需要时请自行在 make menuconfig 界面配置。

部分文件系统对分区大小有最低要求,如 ext4、ubifs,如果在小机端调用格式化分区时报错,可根据报错信息提示增大分区。

对于 private 分区默认为空,使用 DragonSN 工具写号后,则为 vfat 格式的文件系统。



XAA Kithili Ki

文档密级: 秘密

对于 ubi 方案来说,如果需要使用基于块设备的文件系统,则需要在 ubi 之上模拟块设备。在用户空间可调用 ubiblock 工具完成,注意这样模拟出的块设备是只读的,如需可写建议直接使用 ubifs。详见后文ubi 方案特殊说明。

Xinter and All the state of the



4

文件系统支持情况

表 4-1: 文件系统支持情况

存储介质	jffs2	squashfs	ext4	vfat	ntfs	exfat	ubifs
(UBI) spinand	N	Y	Y(ro)	Y(ro)	Y(ro)	N	Y
TF +	N	N S	Y	Y	Y	N	ge ^{er} N
U 盘	N	Negres	Y	Y	Y	Walue	N

🛄 说明

- 1. (ro) 表示只能实现只读: ubi 卷可通过模拟块设备,实现块文件系统的读,但不支持写。
- 2. vfat(fat32)使用内核原生的支持,exfat 需要在 Linux-5.7 后社区才正式支持,因此此处标注为不支持。
- 3. ntfs 依赖于第三方工具 ntfs-3g。
- 4. TF 卡/U 盘等,建议使用 vfat 实现 Window/Linux/MacOS 的最大兼容
- 5. vfat/ntfs 等 Window 文件系统,不建议 用做嵌入式存储,除非您能保证其掉电安全和移植文件系统修复工具。

🔔 警告

关于文件系统的选择,有以下几点需要注意:

- 1. 为了保证掉电不变砖,根文件系统务必只读 (squashfs),或者挂载为 ro 模式。
- 2. ext4/ubifs 等文件系统分区大小必须足够大,以确保能正确创建日志块,否则有掉电变砖风险分区大小请参考章节分区大小与对齐。

4.1 ext4 与日志

4.1.1 ext4 的日志

与服务器等长期稳定供电的情况不同,嵌入式设备随时有掉电的可能。不管在任意时间掉电,文件系统都需要保持一致性,换句话说,保证文件不会因为掉电丢失。

如果文件系统只读,则不需要日志。日志只是确保写的安全。

🛄 说明

什么是文件系统的一致性?

文件系统元数据块记录了有什么文件,数据块则保存了实际的文件内容。一致性则表示,元数据块记录了存在某个文件,必定存在对应的数据块,换句话说,就是保证元数据和数据的一致。

如果出现,元数据记录文件 A 存在,但文件 A 的数据块是无效的,或者明明数据块是有效的,但元数据并没任何记录,导致系统并不知道文件存在,就出现了文件系统的不一致。





保证文件不丢失,只保证之前写入的文件数据正常,而非正在写,且因为掉电导致没写完整的文件。对大多数文 件系统而言,更多时候会直接丢弃这没写完整的文件以保证一致性。

ext4 通过日志的形式保证文件系统一致性。其支持 3 种日志模式:

表 4-2: ext 日志模式

日志模式	原理	特点
journal	元数据与数据都写入日志	最安全,但性能最慢
	只有元数据写入日志,但不保证数据先落盘	性能最快,但最不安全
ordered	只有元数据写入日志,且保证数据先落盘,元数据后落盘	折中,默认方案

考虑安全和性能的折中,建议使用 ordered 的日志模式。系统默认使用的就是 ordered 模式。

我们在 mount 命令中显示的挂载参数可显示使用的哪种日志。

\$mount
/dev/by-name/UDISK on /mnt/UDISK type ext4 (rw,...,data=ordered)

4.1.2 分区大小与日志

有时候分区太小,系统会默认把日志功能关闭。可以通过以下方法判断:

\$dumpe2fs \$分区 or 镜像文件>
...
Filesystem features: has_journal...

Journal backup: inode blocks
Journal features: (none)
日志大小: 1024k
Journal length: 1024
Journal sequence: 0x00000001
Journal start: 0
...

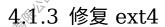
在 **Filesystem features** 中有 has_journal 的标志表示支持日志。在 **Jorunal** 片段中也详细描述了日志块的大小等信息。

如果创建的文件系统没有日志,对大多数用户而言,扩大分区大小是最简单的做法。专业的做法 可以通过缩小块大小,取消预留块等方式为日志腾挪出空间。

按以往经验,对小容量 (<100M) 的存储而言,在资源文件之外预留 3-5M 的空间用于文件系统的元数据即可。







ALLWIMERS

ext4 文件系统每次重启后,建议都进行一次修复,确保文件系统稳定。

修复可以参考以下命令:

e2fsck -y <分区>

XAA Kithili Ki

如章节global 类型 config中描述,如果使用 procd 引导启动,在 fstab 中使能check_fs,也可实现在挂载前自动修复。

*II_GILUANGOBS

版权所有 © 珠海全志科技股份有限公司。保留一切权利

18

XIFAFE KITIKH HALIZ ALIKE



5 UBI (spi) Nand

当前 UBI 方案仅适用于小容量 spinand。

UBI 方案是社区普遍使用的 Flash 存储方案,其构建在 mtd 设备之上,由 UBI 子系统屏蔽 Nand 特性,对接 UBIFS 文件系统。其层次结构由上往下大致如下:

表 5-1: UBI 子系统层次结构

	-02	200
层次	层级	功能
0 (最上层)	UBIFS	ubifs 文件系统
1	UBI 子系统	Nand 特性管理
2	MTD 子系统	封装 Flash,向上提供统一接口
3 (最下层)	Flash	具体的 Flash 驱动

我们 把 MTD 分区称为物理分区,把 UBI 卷 (分区) 成为逻辑分区,因为

- MTD 分区是按 Flash 的物理地址区间划分分区
- UBI 卷 (分区) 是动态映射的区间

全志的 UBI 方案中,创建了这些 MTD 物理分区:

表 5-2: UBI 方案物理分区

	37	, XT
分区名	大小	作用
boot0	4/8 个物理块	存放 boot0
uboot	4/20M	存放 uboot
secure storage	8 个物理块	存放关键数据
pstore	512K	奔溃日志转存
sys	剩余空间	提供给 ubi 子系统划分逻辑分区

驱动会在 sys 的 MTD 物理分区上根据sys_partition.fex构建 UBI 逻辑分区 (卷)。

UBI 设备向上呈现为字符设备,无法直接使用诸如 ext4 这样基于块设备的文件系统。但 UBI 子系统支持模拟只读的块设备,即把 UBI 逻辑卷模拟成只读的块设备。基于此,可以做到根文件系统依然使用 squashfs 这样的块文件系统。

版权所有 © 珠海全志科技股份有限公司。保留一切权利





社区为 UBI 设备专门设计了 ubifs 文件系统。经过验证,其配合 UBI 子系统可保证数据掉电安全以及提供通用文件系统所有功能,甚至还提供文件系统压缩功能。

除此之外,UBI 设备更新(OTA 更新)也不能直接裸写设备,需要通过 **ubiupdateval** 命令更新。

♡ 技巧

详细的 OTA 方法,请参见 OTA 相关文档。

5.1 ubi 相关工具

5.1.1 ubinfo

输出指定 ubi 设备信息。

例子:

ubinfo /dev/by-name/root(s #查看rootfs卷的信息 ubinfo -a #查看所有卷的信息

可参考总容量说明

5.1.2 ubiupdatevol

更新指定卷上的数据。

例子:

ubiupdatevol -t /dev/by-name/boot #

#清除boot卷的数据

ubiupdatevol /dev/by-name/boot /tmp/boot.img #将/tmp/boot.img写到boot卷,卷上原有数据会完全丢失

可参考模拟块设备

5.1.3 ubiblock

基于一个 ubi 卷,生成模拟的只读块设备

例子:

【 ubiblock -c /dev/by-name/test #将test卷生成一个块设备节点

可参考模拟块设备



5.1.4 其他

在 tina 方案上,烧录固件时已经完成 mtd 和 ubi 卷的创建,启动时自动 attach 并挂载对应的分区,无需再自行处理。因此以下命令一般不会用到。

ubiformat: 将裸 mtd 格式化成 ubi

ubiattach: 将 mtd 关联到 ubi

ubidetach: 将 ubi 与 mtd 解除关联

ubimkvol: 创建 ubi 卷

ubirmvol: 移除 ubi 卷

XIRAGE RETURN TO THE REPORT OF THE PARTY OF



6

rootfs data 及 UDISK

6.1 overlayfs 简介

Tina 默认根文件系统格式使用 squashfs 格式,这是一种只读压缩的文件系统。很多应用则需要文件系统可写,特别是/etc 等存放较多配置文件的目录,为了满足可写的需求,Tina 默认使用 overlayfs 技术。overlayfs 是一种堆叠文件系统,可以将底层文件系统和顶层文件系统的目录进行合并呈现。

6.2 使用 rootfs data 作为 overlayfs

Tina 常用的方式是专门划分一个 rootfs_data 分区,先格式化成可写的文件系统(如 ext4/ubifs),再进一步挂载为 overyfs,成为新的根,让上层应用认为 rootfs 是可写的。

rootfs_data 分区的大小就决定了应用能修改多少文件。具体原理和细节请参考网上公开资料,此处仅举简单例子辅助理解。

- 1. 底层(即 rootfs 分区的文件系统)不存在文件 A,应用创建 A,则 A 只存在于上层(即 rootfs data 分区)。
- 2. 底层存在文件 B,应用删除 B,则 B 仍然存在于底层,但上层会创建一个特殊文件屏蔽掉,导致对应用来说 B 就看不到了,起到删除的效果。
- 3. 底层存在文件 C,上层修改 C,则 C 会先被整个拷贝到上层,C 本身多大就需占用多大的上层空间,在这个基础上应用对上层的 C 进行修改。

基于以上理解,可按需配置 rootfs_data 的大小。一般开发期间会配置得较大,量产时可减小(考虑实际只会修改少量配置文件)甚至去除(需要确认所有应用均不依赖 rootfs 可写)。

6.3 使用 UDISK 作为 overlayfs

如果希望 overlayfs 的空间尽可能较大,也可考虑直接使用 UDISK 分区作为上层文件系统空间。

可将_target/allwinner/xxx/base-files/etc/config/fstab 中的 rootfs_data 分区及 UDISK 分区的挂载配置 disable 掉。



```
config 'mount'
                         '/overlay®
    option
            target
                         '/dev/by-name/rootfs_data'
    option device
    option options
                         'rw, sync, noatime'
    option
            enabled
                              #此行改成0
config 'mount'
                         '/mnt/UDISK'
    option target
                         '/dev/by-name/UDISK'
    option
            device
    option
            options
                         'rw,async,noatime'
    option
            enabled
                         '0'
```

再新增一个配置,将 UDISK 直接挂载到/overlay 目录。

```
config 'mount'
    option target
                        '/overlay'
                        '/dev/by-name/UDISK
   option device
                        'rw,sync,noatime'
    option options
   option
           enabled
```

6.4 如何清空 rootfs_data 和 UDISK

出于恢复出厂设置或其他需要,有时需要清空 rootfs data 和 UDISK。

一般不建议在文件系统仍处于挂载状态时直接操作对应的底层分区,因此建议需要清空时,不要 直接操作对应块设备,而是先设置标志并重启,再在挂载对应分区前的启动脚本中检测到对应标 志后,对分区进行重新格式化。

当前 $79_{\text{format_partition}}$ 中实现了 $+ \uparrow_{\text{clean_parts}}$ 功能,会检测 env 分区中的 parts_clean 变量并清空对应分区的头部。清空后,rootfs_data 和 UDISK 分区会自动重 新触发格式化。

```
# to clean rootfs_data and UDISK, please run
fw_setenv parts_clean rootfs_data:UDISK
reboot
```

具体实现请查看

```
package/base-files/files/lib/preinit/79_format_partition
```



7 关键数据保护

设备上保存的一些关键的数据,例如 mac,SN 号等,一般要求在重新刷机时不丢失。以下介绍 刷机数据不丢失的解决方案。

7.1 逻辑分区保护方案

7.1.1 分区设置

此处的逻辑分区,是指在分区表 (sys_partition.fex/sys_partition_mor.fex) 中定义的分区。

名字为 private 的分区会特殊处理,默认刷机数据不丢失。

其他名字的分区,如果指定 keydata=0x8000 属性,则刷机数据不丢失。

对于 private 分区或设置了 keydata=0x8000 属性的分区,请勿设置 downloadfile。

7.1.2 实现原理

对 private 分区 (配置了 keydata=0x8000 属性同理) 保护的方式是,擦除之前先申请一片内存,然后根据 flash 中的旧分区表,读出 private 分区内容。

接着进行擦除,然后再按照新的分区表,将 private 分区内容写回 flash 上新分区所在位置。

7.1.3 常见用法

- 1. 使用全志的 DragonSN 工具,选择私有 key 模式,将 key 写入 private 分区。写入后 private 分区默认会是一个 vfat 文件系统,启动后挂载/dev/by-name/private,即可读出 key。 DragonSN 的具体用法请参考工具自带文档。
- 2. 不使用 DragonSN 工具,由应用自行负责写入数据和读出数据,直接在用户空间操作/dev/by-name/private 节点即可。量产时可自行开发 PC 端工具,通过 adb 命令来完成 key 的写入。

成权所有 © 珠海全志科技股份有限公司。保留一切权利

文档密级: 秘密



7.1.4 ubi 方案特殊说明

7.1.4.1 模拟块设备

对于 nand ubi 方案,逻辑分区是对应到 ubi 卷,由于 ubi 的特性,无法直接写数据到/dev/by-name/private 节点,需要通过 ubiupdatevol 工具来更新卷,或者自行在应用中按照 ubi 卷更新步骤操作。

当基于 ubi 卷构建 vfat 文件系统时,需要先基于 ubi 卷模拟块设备,且挂载上的 vfat 文件系统是只读的。操作示例如下。

```
#查看private分区对应的ubi节点
root@Tinatinux:/# ll /dev/by-name/private
lrwxrwxrwx
             1 root
                        root
                                       11 Jan 1 00:00 /dev/by-name/private -> /dev/
  wbi0_4
#创建模拟的块设备
root@TinaLinux:/# ubiblock -c /dev/ubi0_4
block ubiblock0_4: created from ubi0:4(private)
root@TinaLinux:/# mkdir /tmp/private
root@TinaLinux:/# mount -t vfat /dev/ubiblock0_4 /tmp/private/
#可以读取内容
root@TinaLinux:/# ls /tmp/private/
III T
          magic.bin
#查看挂载情况,为ro
root@TinaLinux:/# mount | grep private
/dev/ubiblock0_4 on /tmp/private type vfat (ro,relatime,fmask=0022,dmask=0022,codepage=437,
    iocharset=iso8859-1,shortname=mixed,errors=remount-ro)
```

7.1.4.2 在设备端制作 vfat 镜像

由于 ubifs 需占用 17 个 LEB,比较占空间,对于只在工厂——次性写入信息,后续只读的场景,一种可考虑的方案是使用 vfat 文件系统。

首先选上 kernel 的 loopback 支持:

```
make kernel_menuconfig 选上 Device Drivers --> [*] Block Devices --> <*>Loopback device support
```

分区表中建立分区,假设为 test 分区

随后可以在小机端准备一个 vfat 镜像:

```
dd if=/dev/zero of=/mnt/UDISK/test.img bs=1M count=1
mkfs.vfat /mnt/UDISK/test.img
mkdir -p /tmp/test
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利



mount /mnt/UDISK/test.img /tmp/test 此时可向 /tmp/test 写入文件 umount /tmp/test

将镜像写入卷中:

ubiupdatevol /dev/by-name/test /mnt/UDISK/test

后续按上文介绍的方法,使用模拟块设备挂载,注意使用模拟块设备挂载后就是只读的了。

7.1.5 可能造成数据丢失的情况

出现以下情况,会导致 private 分区数据丢失。

- 1. 配置了强制擦除,例如 sys config.fex 中配置了 eraseflag = 0x11。
- 2. 无法读取 flash 上的分区表或 private 分区。这个可能的原因包括 flash 卫的数据被破坏了等。
- 3. 新的分区表不包含 private 分区。
- 在烧录过程中掉电。如上所述,烧录时是读出->擦除->写回,在擦除之后,写回之前掉电,则数据丢失。

检测到 private 分区,开始执行保护 private 分区的代码,但执行过程中出错,如 malloc 失败, private 无法读取等,则会导致烧录失败。这个时候,简单的解决方式是,打包一份不保护 private 分区的 tina 固件先进行一次烧录,即可解决问题。具体的:将 sys_config.fex 的 **erase-flag 改为 0x11**,强制擦除。或者临时移除 sys_partition.fex 中的 private 分区。

7.2 物理区域保护方案

另一种保护数据不丢失的思路是,在 flash 上划定一块物理区域,烧录时默认不擦除。

在 Tina 上实现的 secure storage 区域即具有这种特性。secure storage 区域用于保存 key,可代替 private 分区,理论上更为安全(被烧录时误擦除和被别的应用误写的可能性较低)。

7.2.1 nand ubi 方案实现

nand ubi 方案中,预留了一块物理区域,用于 secure storage,对上表现为一个 mtd 分区。用户空间可见。烧录时不会擦除这块区域。在用户空间读写 secure storage 需要使用 ioctl,由内核来协助完成。理论上也可以直接读写 mtd 设备节点,但不推荐,使用这种方式应用需要自行处理坏块等问题。



7.2.2 常见用法

- 1. 使用全志的 DragonSN 工具,选择安全 key 模式,将 key 写入 secure storage 区域。启动后在用户空间调用库读出 key。DragonSN 的具体用法请参考工具自带文档。
- 2. 不使用 DragonSN 工具,由应用自行在用户空间调用库写入数据和读出数据。量产时可自行 开发 PC 端工具,通过 adb 命令来完成 key 的写入。

7.2.3 secure storage 格式

secure storage 有预设的格式,简单总结如下。

- secure storage 总大小为 256KB,因为有备份,所以实际能用 128KB。
- 128KB 分为 32 个 item, 即每个 item 是 4KB。
- item0 用作 secure sotrage map,所以用户能用的实际为 31 个 item。
- 每个 item 内部为键值对的格式,包含 CRC 校验,其中用户可存储的 key 长度最多为 3KB。
- secure_storage_map 中是以 "name:length" 的格式保存所有 item 的信息,所以所有 item 的 "name:length" 字符串长度总和不能超过 secure_storage_map 中 data 的长度。

一般而言,31 个 3KB 的 key 可以满足需求,如果无法满足,例如需要更多数量的 key,则一种解决方式是上层应用自行将多个 key 拼接起来,只要总大小不超过 3KB 即可当成一个 key 写入 secure storage。读出时应用自行反向解析出目标 key 即可。

7.2.4 在 uboot 中读写

基于以上介绍的格式,uboot 中封装了 pst 命令。

```
Usage:

pst pst read|erase [name]
pst read, then dump all data
pst read name, then dump the dedicate data
pst write name, string, write the name = string
pst erase, then erase all secure storage data
pst erase key_burned_flag, then erase the dedicate data
```

pst - read data from secure storageerase flag in secure storage

7.2.5 在用户空间读写

Tina 提供了读写库。

版权所有 © 珠海全志科技股份有限公司。保留一切权利



make menuconfig 选中 Allwinner --> 🕪 libsec key --> [*] Enable secure storage key support

如果选上 demo,则会编译出 demo 程序 sec key test。

root@TinaLinux:/# sec_key_test -h

- -w <name> <data>: write key to sst and private
- -r <name> <data>: read key from sst or private
- -t 0: write some key to secure storage
- -t 1: repetitve read + verify some key from secure storage
- -t 2: repetitve write + verify some key from secure storage
- -t 3: verify some key from secure storage
- -t 4: write some key to private
- -t 5: repetitve read + verify some key from private
- -t 6: repetitve write + verify some key from private
- -t 7: verify some key from private
- -l : list@
- -d : printf hex
- -h : help

更详细的使用方式请参考:

tina/package/allwinner/libkey/readme.txt

7.3 secure storage 区域与 private 分区比较

刷机不丢失的特性:

- private 分区每次刷机,是先读出到 dram,擦除 flash,再写入 flash。刷机中途掉电可能丢失
- secure storage 则是刷机过程完全不会擦除。理论上 secure storage 的数据更安全。

用户空间误操作的可能

- private 分区可以用常规分区更新命令清除掉。
- secure storage 需要通过 ioctl 专用接口访问。

数据格式:

- private 分区可以裸分区读写,也可以格式化成文件系统。
- secure storage 已限制为键值对,key 的长度也有限制。

存放位置:

private 分区大小由分区表配置、是一个可见的普通分区。

版权所有 © 珠海全志科技股份有限公司。保留一切权利



文档密级: 秘密

secure storage 是 flash 上的保留区域,大小固定,分区表中不可见。

备份:

- private 分区没有备份,请避免写入时掉电。或者自行在 private 分区中构建备份。
- secure storage 默认是双备份。

在 uboot 访问:

- private 分区, uboot 通过通用的读写 flash 接口或文件系统接口访问,取决于数据格式。
- sectre storage 区域,有固定格式,通过 uboot 提供的 secure storage 专用接口访问。

- private 分区如果是裸数据,是否校验由应用自行处理。如果是文件系统,则由文件系统特性
- secure storage 格式中带了 crc 校验。





著作权声明

版权所有 © 2021 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



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。

版权所有 © 珠海全志科技股份有限公司。保留一切权利