

UBI 文件系统使用指南

文档版本 V1.0

发布日期 2024-12-21

前言

概述

本文档主要介绍 UBI 文件系统的使用方法，包括：内核配置文件的修改、UBI 根文件系统的配置、烧录脚本和启动参数的修改等。



说明

未经特殊说明，以 XM7606V13 指代 XM76 系列所有芯片。

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
- 软件开发工程师

修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

修订日期	版本	修订说明
2024-12-21	V1.0	DI 版本发布。

目 录

1 内核使能 UBI 配置.....	1
1.1 内核驱动配置 UBI	1
1.2 内核使能 UBI 文件系统.....	1
2 Ubi 根文件系统制作	2
2.1 Sdk 使能 ubi 根文件系统编译	2
2.2 Bootargs 分区修改.....	2
2.3 烧写文件修改	2
3 Ubi 文件系统挂载.....	4
3.1 Mount 一个空 UBIFS 文件系统.....	4
3.1.1 格式化 ubi 分区	4
3.1.2 绑定 UBI 到 MTD 分区	4
3.1.3 创建 UBI 卷	5
3.1.4 挂载空 UBIFS 文件系统	6
3.2 Mount 一个非空 UBIFS 文件系统	7
3.2.1 绑定 UBI 到 MTD 分区	7
3.2.2 挂载 UBIFS 文件系统.....	8

1 内核使能 UBI 配置

1.1 内核驱动配置 UBI

```
Device Drivers --->
<*> Memory Technology Device (MTD) support --->
<*> Enable UBI - Unsorted block images --->

--- Enable UBI - Unsorted block images
(4096) UBI wear-leveling threshold (NEW)
(20) Maximum expected bad eraseblock count per 1024 eraseblocks
(NEW)
[ ] UBI Fastmap (Experimental feature) (NEW)
< > MTD devices emulation driver (gluebi) (NEW)
[ ] Read-only block devices on top of UBI volumes (NEW)
```



说明

必须先使能 UBI 设备驱动，才能找到 UBIFS 文件系统选项。

1.2 内核使能 UBI 文件系统

```
File systems --->
-* Miscellaneous filesystems --->
<*> UBIFS file system support
```



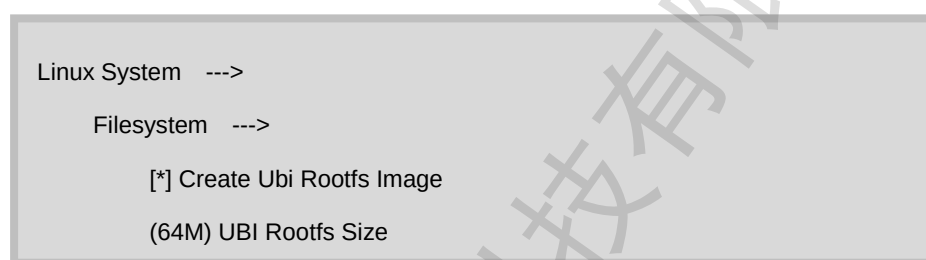
说明

所有配置按以上图中所示，其它 UBI/UBIFS 配置选项使用系统默认值，不要随意选择配置，如果选择不慎，UBI 文件系统可能无法正常工作。

2 Ubi 根文件系统制作

2.1 Sdk 使能 ubi 根文件系统编译

在 SDK 中开启如下相关选项



 说明

必须先使能 UBI 设备驱动，才能找到 UBIFS 文件系统选项。

2.2 Bootargs 分区修改

rootfs 大小根据上面配置的 PARTSIZE 配置

```
bootargs=mem=32M console=ttyAMA0,115200 ubi.mtd=3 root=ubi0:ubifs
rootfstype=ubifs rw rootwait
mtdparts=nand:512K(boot),512K(bootargs),4M(kernel),100M(rootfs),-(others)
```

 说明

Ubi 文件系统分区大小需要大于上面配置的 menuconfig UBI Rootfs Size

2.3 烧写文件修改

到对应产品中 sdk 配置目录下的 prebuilts 目录，nand_partitions.xml 或者 nand_partitions_buildin.xml 修改如下参数

将根文件系统中修改：FileSystem= “raw”，界面

- Length 参数根据 MTD 分区配置
- SelectFile= "rootfs_2k_128k.ubiimg", 这里需要根据我们 nand 的实际情况来选择, 如果是 4K 256K 的 nand 则需要 4K 256K 的镜像。

3

Ubi 文件系统挂载

3.1 Mount 一个空 UBIFS 文件系统

单板当前有 5 个分区，分区的情况如下图。

```
~ # cat /proc/mtd
dev:   size   erasesize  name
mtd0: 00080000 00020000 "boot"
mtd1: 00080000 00020000 "bootargs"
mtd2: 00400000 00020000 "kernel"
mtd3: 02000000 00020000 "rootfs"
mtd4: 06400000 00020000 "ubifs"
mtd5: 07700000 00020000 "others"
```

通过以下，就可以把 mtd4 分区映射成 ubi 卷，做为 ubi 分区使用。

3.1.1 格式化 ubi 分区

使用以下命令格式化 ubi 分区。

```
~ # ubiformat /dev/mtd4
```

 说明

- (1) cfg.mak 中修改 CONFIG_XMEDIA_MTDUTILS_SUPPORT=y
- (2) make rootfs 编译完成后，生成的 UBI 工具放在 out/chip_name/rootfs_build_dir/mtd-utils-x.x.x\install\sbin 目录下， chip_name 路径的命名与编译时选择产品相关。
- (3) 需将 UBI 工具下载到单板，通过命令“chmod +x ‘ubi 工具’”加可执行权限。
- (4) 不推荐擦除(如:用命令 flash_eraseall)分区，擦除分区后，可以正常 mount 到 ubifs。但是擦除分区操作，将使 UBI 系统丢失记录的每个擦除块的擦除次数。

3.1.2 绑定 UBI 到 MTD 分区

绑定 UBI 到 mtd4 分区，使用以下命令。

```
~ # ubiattach /dev/ubi_ctrl -m 4 -b n
```

参数“-m 4”表示使用 mtd4 分区,“-b n”表示保留 n 个块用于坏块处理。只有绑定了 ubi 到 mtd 分区以后,才能在 /dev/ 下找到 ubi 设备“ubi0”,如果曾经创建过 ubi 卷,那么绑定以后才能在 /dev/ 下找到并且访问 ubi 卷“ubi0_0”。命令执行成功,显示信息如下图。

```
~ # ubiattach /dev/ubi_ctrl -m 4
ubi0: attaching mtd4
ubi0: scanning is finished
ubi0: attached mtd4 (name "ubifs", size 100 MiB)
ubi0: PEB size: 131072 bytes (128 KiB), LEB size: 126976 bytes
ubi0: min./max. I/O unit sizes: 2048/2048, sub-page size 2048
ubi0: VID header offset: 2048 (aligned 2048), data offset: 4096
ubi0: good PEBs: 800, bad PEBs: 0, corrupted PEBs: 0
ubi0: user volume: 0, internal volumes: 1, max. volumes count: 128
ubi0: max/mean erase counter: 3/1, WL threshold: 4096, image sequence number:
    1473557509
ubi0: available PEBs: 756, total reserved PEBs: 44, PEBs reserved for bad PEB handling:
    40
ubi0: background thread "ubi_bgt0d" started, PID 91
```

最后一行打印“ubi_bgt0d”表示成功创建设备 ubi0, 查看所有设备“ls /dev/ubi*”, 将发现多一个设备“/dev/ubi0”。

3.1.3 创建 UBI 卷

UBI 卷可以理解为 UBI 设备的分区。创建 ubi 卷命令如下:

```
~ # ubimkvol /dev/ubi0 -N ubifs -s SIZE
```

参数“/dev/ubi0”是上一步骤创建的 ubi 设备。

参数“-N ubifs”表示创建的卷名为“ubifs”。

参数“-s SIZE”表示创建的分区大小。

 说明

- (1) SIZE 值应小于“/dev/ubi0”设备能提供的空间大小。
- (2) 可以使用命令“ubinfs”查看当前可使用的 LEBs 大小。如下图红色标记行所示,当 UBI 设备提供的空间为 100 MiB 时,可使用的空间大小为 91.5MiB。所以,应该保证所创建的卷的 SIZE 值小于可使用的 LEBs 空间大小。


```

~ # ubinfo /dev/ubi0
ubi0
Volumes count:                      0
Logical eraseblock size:             126976 bytes, 124.0 KiB
Total amount of logical eraseblocks: 800 (101580800 bytes, 96.8 MiB)
Amount of available logical eraseblocks: 756 (95993856 bytes, 91.5 MiB)
Maximum count of volumes             128
Count of bad physical eraseblocks:    0
Count of reserved physical eraseblocks: 40
Current maximum erase counter value:  3
Minimum input/output unit size:       2048 bytes
Character device major/minor:          250:0

```

创建 UBI 卷命令如下：

```
~ # ubimkvol /dev/ubi0 -N ubifs -s 80MiB
```

查看所有设备“ls /dev/ubi*”，将发现多一个设备“/dev/ubi0_0”。

 说明

卷只用创建一次，创建后，卷信息将被记录在 UBI 设备上，下一次启动，不用再次创建卷。删

除卷，用命令“ubirmvol”。如果使用此命令删除卷，卷上所有数据，也将被删除。

eg: ubirmvol /dev/ubi0 -n 1 - remove UBI volume 1 from UBI device corresponding to /dev/ubi0

3.1.4 挂载空 UBIFS 文件系统

此时就可以将创建的卷挂载到指定的目录上去了，命令如下：

```
~ # mount -t ubifs /dev/ubi0_0 /mnt/
```

或者

```
~ # mount -t ubifs ubi0:ubifs /mnt/
```

参数“/dev/ubi0_0”表示 mount 到卷“ubi0_0”，也可以使用参数“ubi0:ubifs”。某些版本的内核，不支持“/dev/ubi0_0”形式的参数，只能使用“ubi0:ubifs”形式的参数。

“ubi0:ubifs”中的“ubifs”表示卷的名称，在创建 ubi 卷时设置。

Mount 成功，将显示如下信息：

```

~ # mount -t ubifs /dev/ubi0_0 /mnt/
UBIFS (ubi0:0): default file-system created
UBIFS (ubi0:0): background thread "ubifs_bgt0_0" started, PID 97
UBIFS (ubi0:0): UBIFS: mounted UBI device 0, volume 0, name "ubifs"
UBIFS (ubi0:0): LEB size: 126976 bytes (124 KiB), min./max. I/O unit sizes: 2048
bytes/2048 bytes
UBIFS (ubi0:0): FS size: 82534400 bytes (78 MiB, 650 LEBs), journal size 4190208 bytes
(3 MiB, 33 LEBs)
UBIFS (ubi0:0): reserved for root: 3898303 bytes (3806 KiB)
UBIFS (ubi0:0): media format: w4/r0 (latest is w4/r0), UUID 64B9DBEB-970B-44DE-
826D-BDC0FFC6B7E0, small LPT model

```

查看分区信息，将显示如下内容：

```

~ # df -h

```

Filesystem	Size	Used	Available	Use%	Mounted on
/dev/root	32.0M	13.6M	18.4M	42%	/
/dev/ubi0_0	72.1M	24.0K	68.4M	0%	/mnt

UBIFS 文件系统显示的分区大小、剩余空间并不准确。因为 UBIFS 文件保存的是文件压缩后的内容，压缩比率与文件内容相关。可能剩余空间显示只有 2M，但是可以将一个 4M 的文件完整保存。

3.2 Mount 一个非空 UBIFS 文件系统

单板当前有 6 个分区，分区的情况如下图。

```

~ # cat /proc/mtd
dev:   size  erasesize  name
mtd0: 00080000 00020000 "boot"
mtd1: 00080000 00020000 "bootargs"
mtd2: 00400000 00020000 "kernel"
mtd3: 02000000 00020000 "rootfs"
mtd4: 06400000 00020000 "ubifs"
mtd5: 07700000 00020000 "others"

```



说明

mtd4 分区预置了 UBIFS 文件系统。

3.2.1 绑定 UBI 到 MTD 分区

```

~ # ubiattach /dev/ubi_ctrl -m 4

```

命令执行成功，显示信息如下图。

```
~ # ubiattach /dev/ubi_ctrl -m 4
ubi0: attaching mtd4
ubi0: scanning is finished
ubi0: volume 0 ("ubifs") re-sized from 529 to 756 LEBs
ubi0: attached mtd4 (name "ubifs", size 100 MiB)
ubi0: PEB size: 131072 bytes (128 KiB), LEB size: 126976 bytes
ubi0: min./max. I/O unit sizes: 2048/2048, sub-page size 2048
ubi0: VID header offset: 2048 (aligned 2048), data offset: 4096
ubi0: good PEBs: 800, bad PEBs: 0, corrupted PEBs: 0
ubi0: user volume: 1, internal volumes: 1, max. volumes count: 128
ubi0: max/mean erase counter: 1/0, WL threshold: 4096, image sequence number:
    979536375
ubi0: available PEBs: 0, total reserved PEBs: 800, PEBs reserved for bad PEB
    handling: 40
ubi0: background thread "ubi_bgt0d" started, PID 87
```

3.2.2 挂载 UBIFS 文件系统

```
~ # mount -t ubifs /dev/ubi0_0 /mnt/
```

命令执行成功，显示信息如下图。

```
~ # mount -t ubifs /dev/ubi0_0 /mnt/
UBIFS (ubi0:0): background thread "ubifs_bgt0_0" started, PID 93
UBIFS (ubi0:0): start fixing up free space
UBIFS (ubi0:0): free space fixup complete
UBIFS (ubi0:0): UBIFS: mounted UBI device 0, volume 0, name "ubifs"
UBIFS (ubi0:0): LEB size: 126976 bytes (124 KiB), min./max. I/O unit sizes: 2048
    bytes/2048 bytes
UBIFS (ubi0:0): FS size: 63614976 bytes (60 MiB, 501 LEBs), journal size 8634368 bytes
    (8 MiB, 68 LEBs)
UBIFS (ubi0:0): reserved for root: 0 bytes (0 KiB)
UBIFS (ubi0:0): media format: w4/r0 (latest is w4/r0), UUID 53B36CD4-7276-431D-949F-
    82CA8AA0FE1E, small LPT model
```

查看分区信息以及其中内容，将显示如下内容：

```
~ # df -h
Filesystem      Size      Used Available Use% Mounted on
/dev/root        32.0M     13.6M     18.4M   42% /
/dev/ubi0_0      55.4M      6.3M     49.1M   11% /mnt
~ # cd /mnt/
/mnt # ls
bin      home      linuxrc   opt       sbin      usr
boot     init      lost+found  proc      share     var
dev      komod     mnt       root      sys
etc      lib       nfsroot   run       tmp
```