多卷UBI文件系统

Multiple Volumes on UBI Device

江天文

2024/06/12

目的

- 在串口NAND(W25N01GV 1G-bit)上划分4个MTD分区，在环境文件bootenv中定义MTD分区如下：

  mtdparts=nand0:0x200000@0x0(uboot),0x400000@0x200000(kernel),0x100000@0x600000(devicetree),0x7900000@0x700000(rootfs)

- 分区mtd3 ($ cat /proc/mtd)采用多卷UBI文件系统，卷0(ubi0:rootfs)为只读的rootfs UBIFS文件系统，卷1(ubi0:user)为可读可写的user UBIFS文件系统。在环境文件bootenv中指定root为只读为：

  ubi.mtd=3 root=ubi0:rootfs ro rootfstype=ubifs

配置UBIFS文件系统

- W25N01GV的规格如下

  Pagesize

  2048

  Physical Erase Block (PEB)

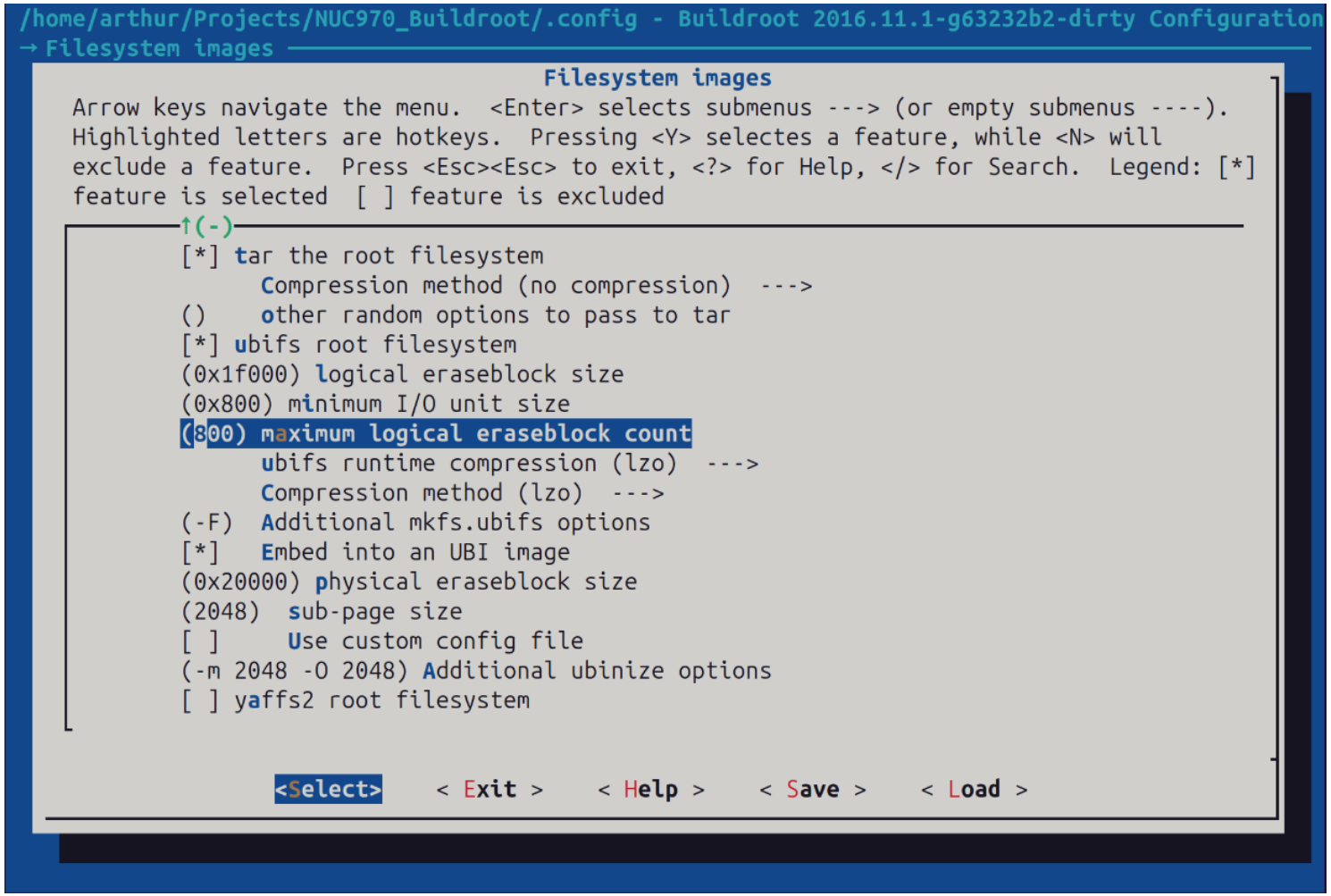
  128KiB

  Logical Erase Block (LEB)

  126976

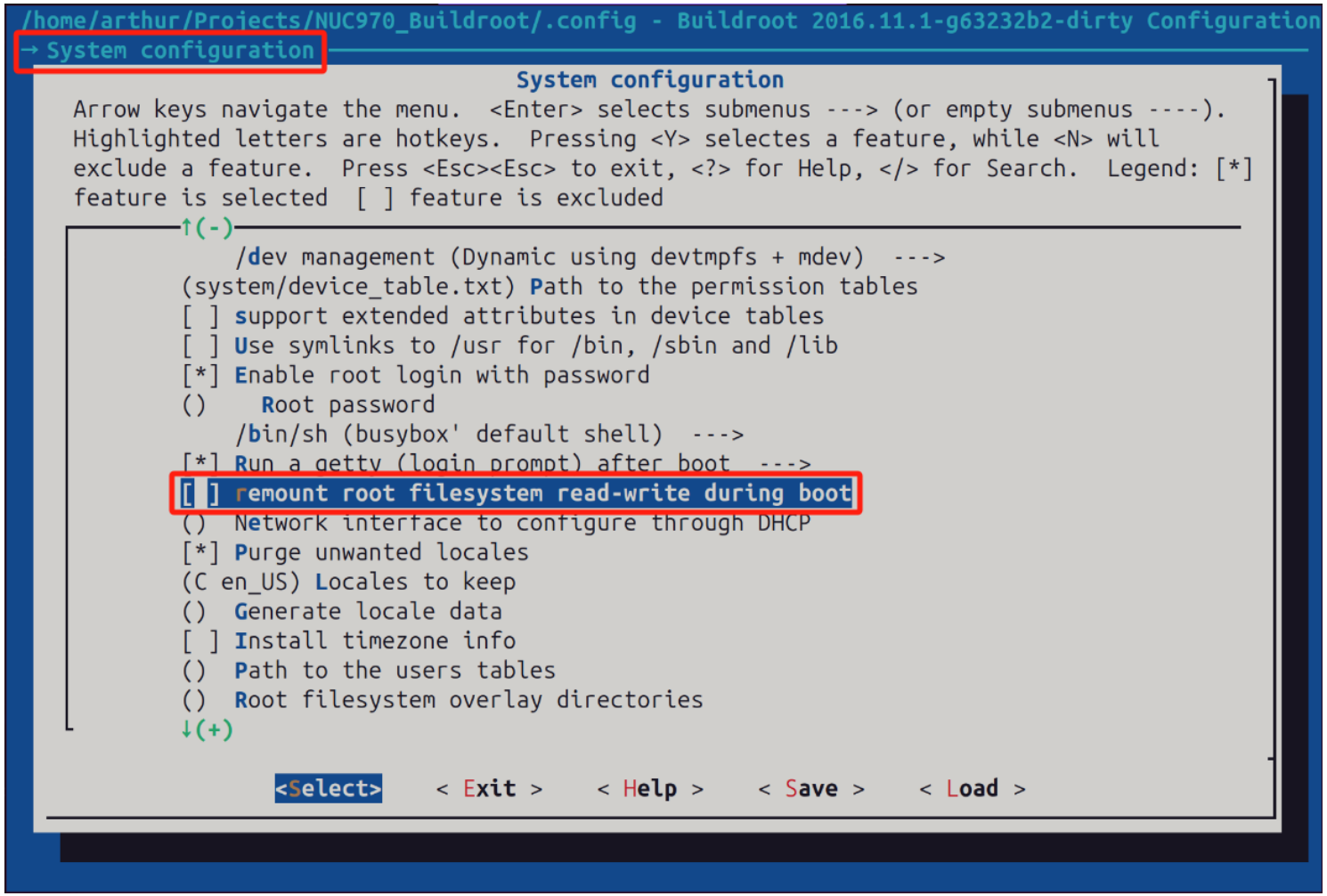
- 生成rootfs.ubifs

  $ make rootfs-ubifs



取消可读可写的方式重新挂载root文件系统

- 配置Buildroot，取消可读可写的方式重新挂载root文件系统



生成rootfs.ubifs和user.ubifs

- 在前面页面中配置好UBIFS文件系统参数后，执行命令$make rootfs-ubifs即可生成output/images/rootfs.ubifs.
  在执行make rootfs-ubifs前，先修改output/target/etc/fstab增加行：

```
ubi0:user   /mnt   ubifs   defaults   0   0
```

- 假设从用户目录workspace/ubifs_user创建user.ubifs
- 在Buildroot的当前目录下创建目录workspace/ubifs_user
  $ mkdir -p workspace/ubifs_user
- 准备一些文件 workspace/ubifs_user/file.txt
  $ echo “multiple volumes on UBI device” > workspace/ubifs_user/file.txt
- 创建user.ubifs
  $ output/host/usr/sbin/mkfs.ubifs -d workspace/ubifs_user -e 0x1f000 -c 32 -m 0x800 -x lzo -F -o output/images/user.ubifs
  -c 32代表4 MiB，8个块占用1 MiB，请参照NAND规格调整。

将rootfs.ubifs和user.ubifs打包成rootfs.ubi

- 准备文件workspace/ubinize.cfg

```
[rootfs]
mode=ubi
vol_id=0
vol_type=dynamic
vol_name=rootfs
vol_alignment=1
vol_size=100MiB
image=output/images/rootfs.ubifs

[user]
mode=ubi
vol_id=1
vol_type=dynamic
vol_name=user
vol_alignment=1
vol_size=4MiB
image=out/images/user.ubifs
```

- 打包rootfs.ubi

$ output/host/usr/sbin/ubinize -o output/images/rootfs.ubi -m 0x800 -p 0x20000 -s 2048 -m 2048 -O 2048 workspace/ubinize.cfg

环境文件

bootenv

```
baudrate=115200
bootdelay=3
#ethact=emac
#ethaddr=00:00:00:11:66:88
stderr=serial
stdin=serial
stdout=serial
#serverip=10.130.11.5
#ipaddr=10.130.11.120
# for SPI NAND
setupspi=sf probe 0 50000000
loadkernel=sf read 0x7fc0 0x200000 0x400000;sf read 0x1400000 0x600000 0x100000
bootcmd=run setupspi;run loadkernel;bootm 0x7fc0 - 0x1400000
# for parallel NAND
#loadkernel=nand read 0x7fc0 0x200000 0x400000
#bootcmd=nand read 0x7fc0 0x200000 0x400000;nand read 0x1400000 0x600000 0x100000;bootm 0x7fc0 - 0x1400000
bootargs=noinitrd ubi.mtd=3 root=ubi0:rootfs ro rootfstype=ubifs mtdparts=nand0:0x200000@0x0(u-boot),0x400000@0x200000(kernel),0x100000@0x600000(device-tree),0x7900000@0x700000(rootfs) console=ttyS0,115200n8 rdinit=/sbin/init mem=64M ignore_loglevel
```

NuWriter烧写地址

文件	类型	地址
u-boot-spl.bin	Loader	0x200
u-boot.bin	Data	0x100000
uImage	Data	0x200000
nuc980-iot-v1.0.dtb	Data	0x600000
rootfs.ubi	Data	0x700000
bootenv	Environment	0x80000

mkfs.ubifs用法

$ output/host/usr/sbin/mkfs.ubifs -d /path/to/dir -m 2048 -e 0x1f000 -c 32 -x lzo -F -o output/images/user.ubifs	
-r, -d, --root=DIR	从目录DIR开始创建文件系统
-m, --min-io-size=SIZE	最小I/O单位大小
-e, --leb-size=SIZE	逻辑擦除（LEB）块大小
-c, --max-leb-cnt=COUNT	最大逻辑擦除块数
-x, --compr=TYPE	压缩格式lzo
-F, --space-fixup	首次挂载时，固定文件系统的自由空间
-o	输出文件

ubinize用法

$ output/host/usr/sbin/ubinize -o output/images/rootfs.ubi -p 0x20000 -m 2048 -s 2048 -O 2048 ubinize.cfg	
-o, --output=<file name>	输出文件
-p, --peb-size=<bytes>	物理擦除块大小
-m, --min-io-size=<bytes>	最小I/O单位大小
-s, --sub-page-size=<bytes>	子页大小
-O, --vid-hdr-offset=<num>	VID头偏移地址
Ubinize.cfg	配置文件

测试只读root文件系统和可读可写的user文件系统

- 在 / 下执行$ touch hello会显示“只读文件系统”提示

```
# touch hello
touch: hello: Read-only file system
```

- 在 /mnt 下执行 $ touch test.txt 会显示成功 (user.ubifs或ubi0:user挂载在/mnt下)

```
# cd /mnt
# ls
file1  file2  jj
# touch test.txt
# ls
file1     file2     jj        test.txt
#
```

Joy of innovation

NUVOTON

谢谢

謝謝

Děkuji

Bedankt

Thank you

Kiitos

Merci

Danke

Grazie

ありがとう

감사합니다

Dziękujemy

Obrigado

Спасибо

Gracias

Teşekkür ederim

Cảm ơn