UBIFS Support

Introduction

UBIFS is next generation Flash based file-system. Its bit different from other file-systems like JFFS2, YFFS2 because, JFFS2 file system works on top of MTD devices, but UBIFS works on top of UBI volumes and cannot operate directly on top of MTD devices. In other words, there are 3 subsystems involved:

- **MTD subsystem**, which provides uniform interface to access flash chips. MTD provides an notion of MTD devices (e.g., /dev/mtd0) which basically represents raw flash;

    http://www.linux-mtd.infradead.org/doc/general.html [1]

- **UBI subsystem**, which is a wear-leveling and volume management system for flash devices; UBI works on top of MTD devices and provides a notion of UBI volumes; UBI volumes are higher level entities than MTD devices and they are devoid of many unpleasant issues MTD devices have (e.g., wearing and bad blocks); see here for more information;

    http://www.linux-mtd.infradead.org/doc/ubi.html [2]

    http://www.linux-mtd.infradead.org/faq/ubi.html [3]

- **UBIFS file system**, which works on top of UBI volumes.

    http://www.linux-mtd.infradead.org/doc/ubifs.html [4]

    http://www.linux-mtd.infradead.org/faq/ubifs.html [5]

    And some in-depth background about UBIFS architecture is as presentation from UBI/UBIFS Authors @ http://www.linux-mtd.infradead.org/doc/ubifs.html#L_documentation [6]

Configuration

How to enable UBI/UBIFS support in Linux Kernel ?

To enable UBIFS support, start the *Linux Kernel Configuration* tool:

- Enabling UBI support on MTD devices.

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

- Enabling UBIFS file-system support.

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

Note: Above configurations are enabled default in omap2plus_defconfig

How to enable UBI/UBIFS support in u-boot ?

The following macros need to be defined in the board config for enabling UBI/UBIFS in uboot

```
#define CONFIG_CMD_NAND
#define CONFIG_CMD_UBI
#define CONFIG_CMD_UBIFS
#define CONFIG_RBTREE
#define CONFIG_MTD_DEVICE
#define CONFIG_MTD_PARTITIONS
#define CONFIG_CMD_MTDPARTS
#define CONFIG_LZO
```

Note: Details of UBI command available in u-boot are given in $U-BOOT/doc/README.ubi

UBIFS User-space tools

UBI user-space tools, as well as other MTD user-space tools, are available as separate package called **mtd-utils**

> http://www.linux-mtd.infradead.org/doc/ubi.html#L_usptools [7]

- **Compiling UBIFS Tools**

For instructions on compiling MTD-utils, refer

> MTD_Utilities
>
> MTD User-space utilities for UBIFS [7]

- **How to creating UBIFS image ?**'

> creating-flashing-ubi-ubifs-images [8]
>
> make ubifs [9]

- **Booting with UBIFS as root file-system**'

> **u-boot#> setenv bootargs 'console=ttyO0,115200n8 root=ubi0:rootfs rw ubi.mtd=<partion_id|or name>,2048 noinitrd rootfstype=ubifs mem=256M rootwait=1**'

> The value of PARTITION_ID depends on MTD device which holds the root filesystem. The below example assumes UBIFS file system is flashed on MTD partition 7.

> One may instead pass in the name of the partition as set in the partition table for the device.

>> u-boot> setenv bootargs 'console=ttyO0,115200n8 root=ubi0:rootfs rw ubi.mtd=7,2048 noinitrd rootfstype=ubifs mem=256M rootwait=1'

- Remotely updating UBI Volume [10]

Creating UBIFS file system

> http://www.linux-mtd.infradead.org/faq/ubifs.html#L_mkfubifs [9]
>
> http://www.linux-mtd.infradead.org/faq/ubi.html#L_ubi_mkimg [11]
>
> creating-flashing-ubi-ubifs-images [8]

white-space-fixup

If the UBIFS image is flashed from u-boot using **nand write** command, then UBIFS image should be built with "white-space-fixup" feature enabled (mkfs.ubifs -F). For details refer

- UBIFS white-space-fixup [12]
- Why is white-space-fixup required ? [13]

subpage support

Currently, OMAP NAND drivers do not support subpage, therefore while building and mounting UBI/UBIFS images

- *subpage-size* should set same as *page-size*
- *offset-of-vid-header* should also be set same as *page-size*

LEB Size Calculations

UBIFS adds two headers (erase-header and volume-id-header) at the start of each NAND block for block identification purposes.

- When sub-page feature is not supported then

    erase-header is written to '1st page' and

    volume-id-header is flashed in '2nd page' of every block,

    so remaining 'n-2' pages are available for user-data, Hence LEB_SIZE = *$BLOCK_SIZE - (2 x $PAGE_SIZE)*

- When sub-page feature is supported then

    both these headers are packed into '1st page' of the block and

    remaining 'n-1' pages are available to store user-data. Hence *LEB_SIZE = $BLOCK_SIZE - (1 x $PAGE_SIZE)*

Flash space overhead [14]

Using UBIFS file system

Preparing NAND partition

Kindly erase the NAND partition before using it for UBI file system. The partition can be erased from either u-boot or from Linux.

Follow below steps to erase.

- From U-boot. Assuming NAND partition to be erased starts from "0x780000" and is of size "0xF880000".

```
u-boot# nand erase 0x00780000 0xF880000
```

- From Linux. Assuming MTD partition 7 needs to be erased and used for UBI file system.

```
root@arago-armv7:~# flash_eraseall /dev/mtd7
```

Flashing UBIFS image to a NAND partition

We can Flash UBIFS image from either Linux Kernel or U-Boot.

Follow steps mentioned here to create an UBIFS image.

From U-Boot,

Get the UBIFS image to U-Boot from tftp or MMC/SD or UART. Lets consider an example of MMC card.

Since we copy the data to NAND, Empty/Erase the required RAM. Then, get the UBIFS image to U-Boot

```
u-boot# mw.b 0x82000000 0xFF <filesystem_image_size> <=== filesystem image size is upward aligned to NAND block size,
                                                         This is required to get rid of "Empty Flash" JFFS2 during kernel boot.
u-boot# mmc rescan

u-boot# fatload mmc 0 0x82000000 ubi.img
```

Next, erase the and flash the UBIFS image to correct NAND partition.

NOTE

On flashing UBIFS image from U-Boot, make sure that ECC selected is in sync with Linux

Example Assuming

1. NAND partition to be erased starts from "0x780000",
2. NAND partition of size "0xF880000" and
3. File system image size to be flashed is 0xFC0000 which is upward aligned to NAND block size

```
u-boot# nand erase 0x780000 0xF880000
u-boot# nand write 0x82000000 0x780000 0xFC0000
```

From Linux,

- Flash the UBI file system image (ubi.img) to MTD partition "X"

```
ubiformat /dev/mtd<X> -f ubi.img -s <subpagesize> -O 2048
```

Here subpage size depends MTD driver. Find subpage size of MTD partition using

```
mtdinfo /dev/mtd<X>
```

Assuming **7**th mtd partition with 2048 byte subpage size, we can use the following command to flash the ubifs image to partition 7.

```
#ubiformat /dev/mtd7 -f ubi.img -s 2048 -O 2048
```

Using UBIFS image as root file system

- Set up the bootargs environment variable as below to use the UBIFS file system image present in a MTD partition:

```
setenv bootargs 'console=ttyO0,115200n8 noinitrd ip=off mem=256M rootwait=1 rw ubi.mtd=X,YYYY rootfstype=ubifs root=ubi0:rootfs init=/init'
```

Where X is the MTD partition number being used for file system and YYYY is the NAND page size. make sure that an UBI file system is flashed into this partition before passing it as a boot partition for Linux.

Assuming **7**th mtd partition,

```
#setenv bootargs 'console=ttyO0,115200n8 noinitrd ip=off mem=256M rootwait=1 rw ubi.mtd=7,2048 rootfstype=ubifs root=ubi0:rootfs init=/init'
```

NOTE

On booting with UBIFS as rootfs, the first boot happens successfully. Before subsequent boot-ups, it is recommended to do a manual "sync" from the console. This allows UBIFS meta data properly updated on the partition. This initial sync will help later recovery.

Mounting UBIFS image as a regular NAND partition

Assuming UBIFS image is already flashed to a NAND Partition, follow below steps to mount the same.

• Attach MTD device to UBI

```
ubiattach /dev/ubi_ctrl -m <X> -O 2048
```

Where "X" is the MTD partition number

Mtd device number **7** can be attached to ubi using

```
#ubiattach /dev/ubi_ctrl -m 7 -O 2048
```

• Mount the UBIFS image

```
mount -t ubifs ubiX:NAME /mount/point
```

Where "X" - UBI device number and "NAME" - UBI volume name from ubinize.cfg file.

Assuming ubi device **0** and **rootfs** is the volume name given in **ubinize.cfg**, ubifs image can be mounted to **/media/card** using

```
#mount -t ubifs ubi0:rootfs /media/card
```

NOTE
On mounting UBIFS as regular partition, it is recommended to do a manual "sync" from the console after mounting. This allows UBIFS meta data properly updated on the partition. This initial sync will help later recovery.

Mounting a NAND partition using UBIFS

We can mount a particular NAND partition with UBIFS file system in the following way

• Format and attach the MTD partition

```
ubiformat /dev/mtd<X> -s <subpagesize> -O 2048
ubiattach /dev/ubi_ctrl -m <X> -O 2048
```

Where "X" is the MTD partition number and "subpagesize" determined using mtdinfo command

MTD partition number 7 with 2048 subpage size can be formatted and attched using

```
ubiformat /dev/mtd7 -s 2048 -O 2048
#ubiattach /dev/ubi_ctrl -m 7 -O 2048
```

• Create an UBI volume - the created volume will be empty

```
ubimkvol /dev/ubi0 -N <label> -s XXMiB
```

Where "XX" is the size of the partition to be mounted and "label" is the name for the volume.

10 MB ubi volume can be created with label **ubifs_volume** as

```
ubimkvol /dev/ubi0 -N ubifs_volume -s 10MiB
```

Also, user can create a UBI volume with volume size set to maximum available size

```
ubimkvol /dev/ubi0 -N <label> -m
```

where "label" is the name for the volume.

```
ubimkvol /dev/ubi0 -N ubifs_volume -m
```

- Mount the MTD partition

```
mount -t ubifs ubi0:<label> /mount/point
```

Make sure that the "label" used during ubimkvol is passed as an argument here.

Mounting of ubi volume can be achieved using

```
#mount -t ubifs ubi0:ubifs_volume /media/card
```

NOTE

On mounting NAND partition as UBIFS, it is recommended to do a manual "sync" from the console after mounting NAND partition. This allows UBIFS meta data properly updated on the partition. This initial sync will help later recovery.

References

[1]   http://www.linux-mtd.infradead.org/doc/general.html
[2]   http://www.linux-mtd.infradead.org/doc/ubi.html
[3]   http://www.linux-mtd.infradead.org/faq/ubi.html
[4]   http://www.linux-mtd.infradead.org/doc/ubifs.html
[5]   http://www.linux-mtd.infradead.org/faq/ubifs.html
[6]   http://www.linux-mtd.infradead.org/doc/ubifs.html#L_documentation
[7]   http://www.linux-mtd.infradead.org/doc/ubi.html#L_usptools
[8]   http://free-electrons.com/blog/creating-flashing-ubi-ubifs-images
[9]   http://www.linux-mtd.infradead.org/faq/ubifs.html#L_mkfubifs
[10] http://www.linux-mtd.infradead.org/doc/ubi.html#L_volupdate
[11] http://www.linux-mtd.infradead.org/faq/ubi.html#L_ubi_mkimg
[12] http://www.linux-mtd.infradead.org/faq/ubifs.html#L_free_space_fixup
[13] http://lists.infradead.org/pipermail/linux-mtd/2014-January/051342.html
[14] http://www.linux-mtd.infradead.org/doc/ubi.html#L_overhead

Article Sources and Contributors

UBIFS Support *Source*: http://processors.wiki.ti.com/index.php?oldid=168589 *Contributors*: Gururaja, Kevinsc, Parth.saxena, Pekon, Trini, X0155329

License

matter take effect in the relevant jurisdiction in which the License terms are sought to be enforced according to the corresponding provisions of the implementation of those treaty provisions in the applicable national law. If the standard suite of rights granted under applicable copyright law includes additional rights not granted under this License, such additional rights are deemed to be included in the License; this License is not intended to restrict the license of any rights under applicable law.

Warning: is_executable() [function.is-executable]: open_basedir restriction in effect. File(/usr/bin/php) is not within the allowed path(s): (/var/www/vhosts/mytiwiki.com/subdomains/wiki/httpdocs/:/tmp/) in **/var/www/vhosts/mytiwiki.com/subdomains/wiki/httpdocs/includes/Wiki.php** on line **655**