Adding RAID & LVM support to Linux BSP for I.MX processors

Recently, some customers are using i.MX processor, they want to add raid & LVM function support to the kernel, but they have encountered the problem that the compilation cannot pass.

Tested it in L4.14.98, L4.19.35 & L5.4.x, Only L4.14.98 bsp exists the problem.

Here are the experimental steps I have done, including the same problems I encountered with the customer, and how to modify the kernel to ensure that the compilation passes.

1. Exporting cross compilation tool chain from yocto BSP

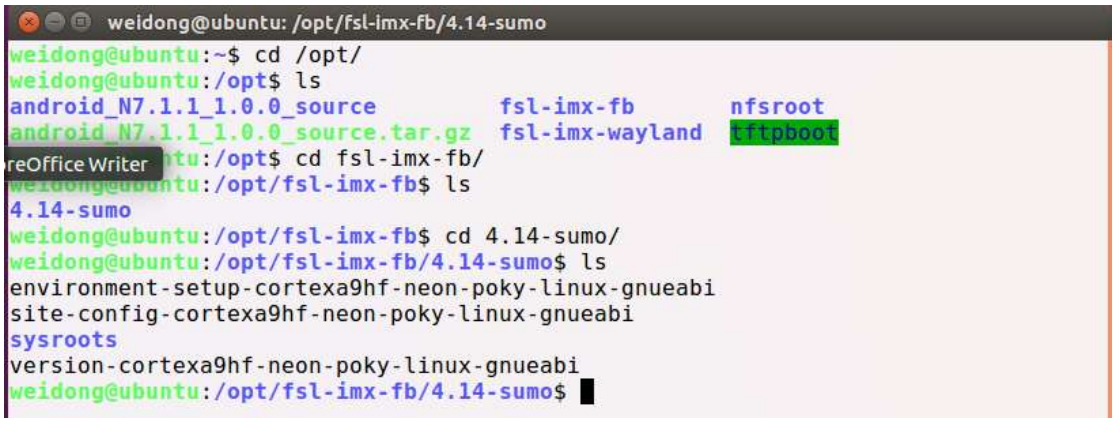
(1) Downloading Yocto BSP and compiling it.

Following steps in i.MX_Yocto_Project_User's_Guide.pdf, download Yocto BSP and compile it successfully.

(2) Exporting cross compilation tool chain

Following methods described in i.MX_Linux_User's_Guide.pdf, export cross compilation tool chain from yocto BSP. See Chapter 4.5.12 of the document, please!

Then cross compilation tool chain will be like below:



(3) Copying linux BSP source code to a new directory

cd ~

mkdir L4.14.98-2.0.0

cd L4.14.98-2.0.0

cp -r ~/imx-yocto-bsp/build-fb/tmp/work/imx6qsabresd-poky-linux-gnueabi/linux-imx/4.14.98-r0/git ./

Then all linux source code has been copied to L4.14.98-2.0.0, which is the top directory of linux kernel source code, I will compile kernel image here.

2. Compiling linux kernel

cd ~/L4.14.98-2.0.0

source /opt/fsl-imx-fb/4.14-sumo/environment-setup-cortexa9hf-neon-poky-linux-gnueabi

export ARCH=arm

make imx_v7_defconfig

make menuconfig

Then we will add RAID and LVM modules to linux kernel.

In order to reproduce errors, I added all related modules to kernel.

See below, please!

Device drivers---->Multiple devices driver support (RAID and LVM)

```
-*- Multiple devices driver support (RAID and LVM)
{M}   RAID support
<M>     Linear (append) mode
-M-     RAID-0 (striping) mode
-M-     RAID-1 (mirroring) mode
-M-     RAID-10 (mirrored striping) mode
-M-     RAID-4/RAID-5/RAID-6 mode
<M>     Multipath I/O support
<M>     Faulty test module for MD
<M>   Block device as cache
[*]     Bcache debugging
[*]     Debug closures
<M>   Device mapper support
[*]     request-based DM: use blk-mq I/O path by default
[*]     Device mapper debugging support
[*]     Block manager locking
[*]       Keep stack trace of persistent data block lock holders
<M>     Crypt target support
<M>     Snapshot target
<M>     Thin provisioning target
<M>     Cache target (EXPERIMENTAL)
<M>       Stochastic MQ Cache Policy (EXPERIMENTAL)
<M>     Era target (EXPERIMENTAL)
<M>     Mirror target
<M>       Mirror userspace logging
<M>     RAID 1/4/5/6/10 target
<M>     Zero target
<M>     Multipath target
<M>       I/O Path Selector based on the number of in-flight I/Os
<M>       I/O Path Selector based on the service time
<M>     I/O delaying target
[*]     DM uevents
<M>     Flakey target
<M>     Verity target support
[*]       Verity forward error correction support
<M>     Switch target support (EXPERIMENTAL)
<M>     Log writes target support
<M>     Integrity target support
```

After save and exit, began to compile kernel.

make (make –j4)

The following errors will occur:

---

drivers/md/dm-rq.c: In function ‘dm_old_init_request_queue’:

drivers/md/dm-rq.c:716:2: error: implicit declaration of function ‘elv_register_queue’; did you mean ‘blk_register_queue’? [-Werror=implicit-function-declaration]

```
  elv_register_queue(md->queue);
  ^~~~~~~~~~~~~~~~~~
  blk_register_queue
```

cc1: some warnings being treated as errors

scripts/Makefile.build:326: recipe for target 'drivers/md/dm-rq.o' failed

make[2]: *** [drivers/md/dm-rq.o] Error 1

scripts/Makefile.build:585: recipe for target 'drivers/md' failed

make[1]: *** [drivers/md] Error 2

Makefile:1039: recipe for target 'drivers' failed

make: *** [drivers] Error 2

---

3. Finding out root cause and solving it

(1) elv_register_queue( ) function

The function is loaded in dm-rq.c :

```
int dm_old_init_request_queue(struct mapped_device *md, struct dm_table *t)
{
… …
    elv_register_queue(md->queue);
… …
}
```

BUT compiler didn't find it's declaration and entity.

Searching source code, and found it **declared** in linux_top/block/blk.h:

```
… …
int elv_register_queue(struct request_queue *q);
… …
```

It's **entity** is in linux_top/block/elevator.c:

```
int elv_register_queue(struct request_queue *q)
{
… …
}
```

(2) Adding declaration and exporting the function

--- Declaration

Add the line below to dm-rq.c:

```
… …
extern int elv_register_queue(struct request_queue *q);
… …
```

--- Exporting the function(elevator.c)

Add EXPORT_SYMBOL(elv_register_queue); to the end of function, see below.

```
int elv_register_queue(struct request_queue *q)
{
… …
}
EXPORT_SYMBOL(elv_register_queue);
```

4. Re-compiling Linux Kernel

The above error will not occur and the compilation will complete successfully.