Deployment

U-Boot booting options:

- **Booting kernel with network via TFTP:**

Setup host system:

- Install tftp server

veda@linux # **sudo apt-get install tftpd xinetd tftp**

- Configure tftp server

- Create tftp file in /etc/xinetd.d/ directory

veda@linux # **sudo vim /etc/xinetd.d/tftp**

- Copy following into that file and save it.

```
service tftp
{
   protocol        = udp
   port            = 69
   socket_type     = dgram
   wait            = yes
   user            = nobody
   server          = /usr/sbin/in.tftpd
   server_args     = /tftpboot
   disable         = no
}
```

```
root@linux:~/elinux/workspace# vi /etc/xinetd.d/tftp
```

```
service tftp
        {
                protocol        = udp
                port            = 69
                socket_type     = dgram
                wait            = yes
                user            = nobody
                server          = /usr/sbin/in.tftpd
                server_args     = /tftpboot
                disable         = no
        }
```

- **/tftpboot** is the server searching path to serve files to clients.

- Start tftp server

veda@linux # **sudo service xinetd stop**

veda@linux # **sudo service xinetd start**

```
root@linux:~/elinux/workspace# service xinetd stop
xinetd stop/waiting
root@linux:~/elinux/workspace# service xinetd start
xinetd start/running, process 14192
```

- Now host system in ready to transfer images using tftp

- Copy **uImage** into **/tftpboot** (server directory).
- Copy **rootfs.img** into **/tftpboot** .

veda@linux # **cp $(/path/to/linux-src)/arch/arm/boot/uImage /tftpboot**
veda@linux # **cp $(path/to/rootfs.img)/rootfs.img /tftpboot**

```
root@linux:~/elinux/workspace# cp linux-3.9/arch/arm/boot/uImage /tftpboot/
root@linux:~/elinux/workspace# ls /tftpboot
uImage
root@linux:~/elinux/workspace# cp rootfs.img /tftpboot/
root@linux:~/elinux/workspace# ls /tftpboot/
rootfs.img  uImage
```

Setup target system:

- Set server ip address on target
MINI2440 # **setenv serverip 10.0.0.4**

- Set target ip address
MINI2440 # **setenv ipaddr 10.0.0.111**

```
' - try 'help'
MINI2440 # setenv serverip 10.0.0.4
MINI2440 # printenv serverip
serverip=10.0.0.4
MINI2440 # setenv ipaddr 10.0.0.111
MINI2440 # printenv ipaddr
ipaddr=10.0.0.111
```

Porting using initrd:(Filesystem size should be <=16M)

- Transfer kernel image to target
tftpboot command in U-Boot is used to transfer images from host to target

MINI2440 # **tftpboot $<RAM_Addr> $<File_name>**

Example :
on mini2440
- Copying kernel to ram_addr 0x31000000
MINI2440 # **tftpboot 0x31000000 uImage**

- Copying rootfs.img to ram_addr 0x32000000
MINI2440 # tftpboot 0x31000000 uImage

- Set bootargs environment variable to transfer boot arguments to the kernel.
MINI2440 # **setenv bootags console=<serial device>,<baud_rate> root=<rootfs_device>**
initrd=<ram_addr_rootfs.img>,<size_of_rootfs.img>

Example:
on mini2440

MINI2440 # **setenv bootags console=ttySAC0,115200 root=/dev/ram0 initrd=0x32000000,4M**

- Booting kernel image

MINI2440 # **bootm $<RAM_Addr_kernel>**

Example :

on mini2440

MINI2440 # **bootm 0x31000000**

```
MINI2440 # tftpboot 0x31000000 uImage
dm9000 i/o: 0x20000300, id: 0x90000a46
DM9000: running in 16 bit mode
MAC: 08:08:11:18:12:27
TFTP from server 10.0.0.4; our IP address is 10.0.0.111
Filename 'uImage'.
Load address: 0x31000000
Loading: checksum bad
checksum bad
T #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         ########################################
done
Bytes transferred = 2542144 (26ca40 hex)
```

```
MINI2440 # tftpboot 0x32000000 rootfs.img
dm9000 i/o: 0x20000300, id: 0x90000a46
DM9000: running in 16 bit mode
MAC: 08:08:11:18:12:27
TFTP from server 10.0.0.4; our IP address is 10.0.0.111
Filename 'rootfs.img'.
Load address: 0x32000000
Loading: checksum bad
T #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         ##############checksum bad
##################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         ########################################
done
Bytes transferred = 4194304 (400000 hex)
```

```
MINI2440 # setenv bootargs console=ttySAC0,115200 root=/dev/ram0 initrd=0x32000000,4M
MINI2440 # printenv bootargs
bootargs=console=ttySAC0,115200 root=/dev/ram0 initrd=0x32000000,4M
MINI2440 # bootm 0x31000000
## Booting kernel from Legacy Image at 31000000 ...
   Image Name:   Linux-3.9.0
   Created:      2014-02-05  13:10:20 UTC
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    2542080 Bytes =  2.4 MB
   Load Address: 30008000
   Entry Point:  30008000
   Verifying Checksum ... OK
   Loading Kernel Image ... OK
OK

Starting kernel ...

Uncompressing Linux... done, booting the kernel.
Booting Linux on physical CPU 0x0
Linux version 3.9.0 (root@linux) (gcc version 4.7.3 (Buildroot 2013.08.1) ) #2 Wed Feb 5 18:40:13 IST
CPU: ARM920T [41129200] revision 0 (ARMv4T), cr=c0007177
CPU: VIVT data cache, VIVT instruction cache
Machine: MINI2440
Memory policy: ECC disabled, Data cache writeback
CPU S3C2440A (id 0x32440001)
S3C24XX Clocks, Copyright 2004 Simtec Electronics
S3C244X: core 405.000 MHz, memory 101.250 MHz, peripheral 50.625 MHz
CLOCK: Slow mode (1.500 MHz), fast, MPLL on, UPLL on
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 16256
Kernel command line: console=ttySAC0,115200 root=/dev/ram0 initrd=0x32000000,4M
PID hash table entries: 256 (order: -2, 1024 bytes)
```

```
EXT3-fs (ram0): using internal journal
EXT3-fs (ram0): mounted filesystem with ordered data mode
VFS: Mounted root (ext3 filesystem) on device 1:0.
Freeing init memory: 132K
dm9000 dm9000 eth0: link down
dm9000 dm9000 eth0: link up, 100Mbps, full-duplex, lpa 0xCDE1

mini2440 login: veda
login: can't change directory to '/root'
Jan  1 11:31:05 login[779]: root login on 'ttySAC0'
[veda@mini2440:/]# ls
bin         etc         linuxrc     mnt         sbin        usr
dev         lib         lost+found  proc        sys
[veda@mini2440:/]#
```

portinting using Initramfs:(Filesystem size should be <=16M)

- mount rootfs.img to **/mnt**

    veda@linux # **mount /path/to/rootfs.img /mnt -o loop**

- In initramfs kernel by default locate for **init** file under taget **root** (/).
- Create init file in the undeer target **root** (/) .

    veda@linux # **vim /mnt/init**

- Copy the following into **init** file and save it, to start **/sbin/init**

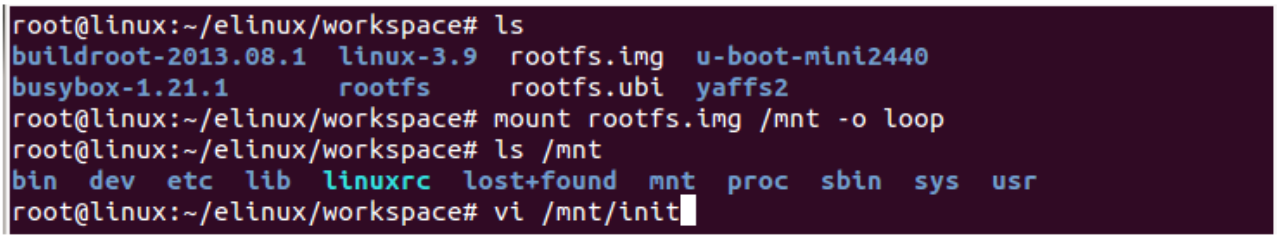
```
#!/bin/sh
#/sbin/init does not get automounted for initramfs
/bin/echo "/sbin/init start now."
exec /sbin/init
```

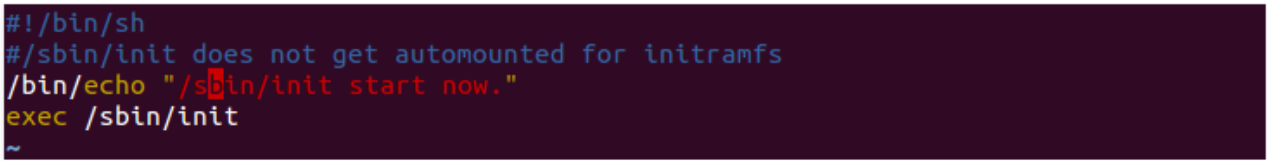
- Give executable permissions to init file

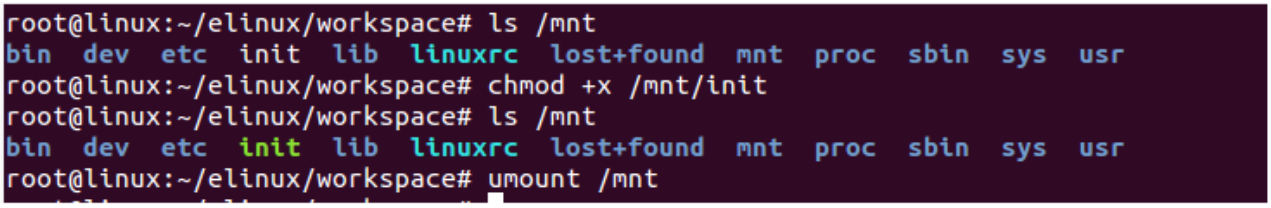
    veda@linux # **chmod +x /mnt/init**

- Unmount virtual block device

    veda@linux # **umount /mnt**







- For initramfs kernel cpio image.
- Create cpio image for the target rootfs.
- Mount rootfs.img to /mnt

    veda@linux # **mount /path/to/rootfs.img /mnt -o loop**

- Create rootfs.cpio

    veda@linux # **cd /mnt**

    veda@linux # **find . | cpio -ovH newc > /output/path/to/rootfs.cpio**

        Note: For more info about cpio give command '**info cpio**'

    veda@linux # **cd -**

- unmount virtual block device.

veda@linux # **umount /mnt**

```
root@linux:~/elin  root@linux: /embedded/arm9/toolchain/buildroot-2013.08.1
root@linux:~/elinux/workspace# cd /mnt
root@linux:/mnt# find . | cpio -oH newc > /root/elinux/workspace/rootfs.cpio
3462 blocks
root@linux:/mnt# cd -
/root/elinux/workspace
root@linux:~/elinux/workspace# ls
buildroot-2013.08.1  linux-3.9  rootfs.cpio  rootfs.ubi       yaffs2
busybox-1.21.1       rootfs     rootfs.img   u-boot-mini2440
root@linux:~/elinux/workspace# umount /mnt
```

- linux source code changes:

veda@linux # **cd linux-3.9/**

veda@linux # **make ARCH=arm menuconfig**

General setup --->

[*] Initial RAM filesystem and RAM disk (initramfs/initrd) support

(**/root/elinux/workspace/rootfs.cpio**) Initramfs source file(s)

- Save it

- Create uImage:

veda@linux # make ARCH=arm CROSS_COMPILE=arm-linux- uImage

```
  OBJCOPY arch/arm/boot/zImage
  Kernel: arch/arm/boot/zImage is ready
  UIMAGE  arch/arm/boot/uImage
Image Name:   Linux-3.9.0
Created:      Sat Feb  8 13:42:35 2014
Image Type:   ARM Linux Kernel Image (uncompressed)
Data Size:    3811232 Bytes = 3721.91 kB = 3.63 MB
Load Address: 30008000
Entry Point:  30008000
  Image arch/arm/boot/uImage is ready
```

- Copy uImage into **/tftpboot** as **uImage-initramfs**

```
root@linux:~/elinux/workspace/linux-3.9# cp arch/arm/boot/uImage /tftpboot/uImage-initramfs
root@linux:~/elinux/workspace/linux-3.9# ls /tftpboot/
rootfs.img  uImage  uImage-initramfs
```

Setting target system:

- Load uImage-initramfs into target ram location 0x31000000

MINI2440 # tftpboot uImage-initramfs 0x31000000

- Setting bootarg environment variable

MINI2440 # setenv bootags console=ttySAC0,115200

- Boot linux kernel from memory 0x31000000

MINI2440 # bootm 0x31000000

```
MINI2440 # tftpboot 0x31000000 uImage-initramfs
dm9000 i/o: 0x20000300, id: 0x90000a46
DM9000: running in 16 bit mode
MAC: 08:08:11:18:12:27
TFTP from server 10.0.0.4; our IP address is 10.0.0.111
Filename 'uImage-initramfs'.
Load address: 0x31000000
Loading: checksum bad
T #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         ############################checksum bad
####################################
         ##############################
done
Bytes transferred = 3811296 (3a27e0 hex)
MINI2440 # setenv bootargs console=ttySAC0,115200
MINI2440 # bootm 0x31000000
## Booting kernel from Legacy Image at 31000000 ...
   Image Name:   Linux-3.9.0
   Created:      2014-02-08   8:12:35 UTC
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    3811232 Bytes =  3.6 MB
   Load Address: 30008000
   Entry Point:  30008000
   Verifying Checksum ...
```

```
Freeing init memory: 1880K
/sbin/init start now.
dm9000 dm9000 eth0: link down
dm9000 dm9000 eth0: link up, 100Mbps, full-duplex, lpa 0xCDE1

mini2440 login: veda
login: can't change directory to '/root'
Jan  1 18:00:19 login[800]: root login on 'ttySAC0'
[veda@mini2440:/]# ls
bin         etc         lib         lost+found  proc        sys
dev         init        linuxrc     mnt         sbin        usr
[veda@mini2440:/]#
```

Mounting nfs root filesystem:

- **Setup host system :**
  - Install NFS server
    # **sudo apt-get install nfs-kernel-serve**
  - Add the following line in **/etc/exports**

    veda@linux # **vim /etc/exports**

    /mnt 10.0.0.111(rw,sync,no_root_squash,no_all_squash,no_subtree_check)

  - save it

    /mnt - Folder to mount as root filesystem on targer
    10.0.0.111 - Target ip address
    Options - (rw,sync, no_root_squash, no_all_squash, no_subtree_check)

  - Start NFS Server

    veda@linux # **/etc/init.d/nfs-kernel-server restart**

```
root@linux:~/elinux/workspace# ls
buildroot-2013.08.1  linux-3.9  rootfs.cpio  rootfs.ubi       yaffs2
busybox-1.21.1       rootfs     rootfs.img   u-boot-mini2440
root@linux:~/elinux/workspace# cp -Rfp rootfs /mnt
root@linux:~/elinux/workspace# ls /mnt
rootfs
root@linux:~/elinux/workspace# ls /mnt/rootfs/
bin  dev  etc  lib  linuxrc  mnt  proc  sbin  sys  usr
root@linux:~/elinux/workspace#
```

Setup target system :(Filesystem size should be <=16M)

- Set bootargs env variable with nfs

MINI2440 # setenv bootargs console=ttySAC0,115200 ip=10.0.0.111:10.0.0.4::255.255.255.0 root=/dev/nfs nfsroot=10.0.0.4:$(mount/path/in/host)

MINI2440 # tftpboot 0x31000000 uImage

MINI2440 # bootm 0x31000000

```
<INTERRUPT>
MINI2440 # setenv bootargs console=ttySAC0,115200 ip=10.0.0.111:10.0.0.4::255.255.255.0 root=/dev/nfs r
MINI2440 # tftpboot 0x31000000 uImage
dm9000 i/o: 0x20000300, id: 0x90000a46
DM9000: running in 16 bit mode
MAC: 08:08:11:18:12:27
TFTP from server 10.0.0.4; our IP address is 10.0.0.111
Filename 'uImage'.
Load address: 0x31000000
Loading: checksum bad
T #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         ########################################
done
Bytes transferred = 2542144 (26ca40 hex)
MINI2440 # bootm 0x31000000
## Booting kernel from Legacy Image at 31000000 ...
   Image Name:   Linux-3.9.0
   Created:      2014-02-05  13:10:20 UTC
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    2542080 Bytes =  2.4 MB
   Load Address: 30008000
   Entry Point:  30008000
   Verifying Checksum ...
```

```
IP-Config: Complete:
     device=eth0, hwaddr=08:08:11:18:12:27, ipaddr=10.0.0.111, mask=255.255.255.0, gw=255.255.255.255
     host=10.0.0.111, domain=, nis-domain=(none)
     bootserver=10.0.0.4, rootserver=10.0.0.4, rootpath=
ALSA device list:
  No soundcards found.
dm9000 dm9000 eth0: link up, 100Mbps, full-duplex, lpa 0xCDE1
VFS: Mounted root (nfs filesystem) on device 0:11.
Freeing init memory: 132K

mini2440 login: veda
login: can't change directory to '/root'
Jan  1 18:44:41 login[778]: root login on 'ttySAC0'
[veda@mini2440:/]# ls
bin      etc      linuxrc  proc     sys
dev      lib      mnt      sbin     usr
[veda@mini2440:/]#
```

Flashing Filesystem and Kernel into Mtd-partitions:(nand)

- For flashing file-system images into Mtdparts we need flash tools for target.
- For that Crosscompile Busybox with flash-tools (i.e flasherase_all ,ubi-tools etc).
- Copy rootfs directory tree into rootfs directory as rfs

    veda@linux # **cp -Rfp rootfs rfs**
    veda@linux # **mv rfs rootfs**

- Copy rootfs directory tree into **/mnt** directory for nfs mount

    veda@linux # cp -Rfp rootfs /mnt

- **Flashing kernel into mtdpartition:**
- Copy uImage into ram

    MINI2440 # **tftpboot 0x31000000 uImage**

- Check the mtd partions info using following command.

    MINI2440 # **mtdparts**

- Erase kernel partition using following command (Becareful while using this cmd, because if you are not giving partition name it erases all flash memory partitions).

    MINI2440 # **nand erase <name_of_partition>**
    name_of_partition – OFFSET of nand
    Example:
    MINI2440 # **nand erase kernel**

- Write uImage into Nand kernel partition
    MINI2440 # **nand write <ram_addr_kernel> <name_of_partition> <size_of_image>**

    Example:
    MiNI2440 # **nand write 0x31000000 kernel 2933888**

- Whenever we need kernel image we just read from nand kernel partition instead of downloading from host. For that following command is used.

    MINI2440 # **nand read <ram_addr_kernel> <name_of_partition> <size_of_image>**
    MINI2440 # **nand read 0x31000000 kernel 2542144**

```
MINI2440 # tftp 0x31000000 uImage-3.9
dm9000 i/o: 0x20000300, id: 0x90000a46
DM9000: running in 16 bit mode
MAC: 08:08:11:18:12:27
TFTP from server 10.0.0.4; our IP address is 10.0.0.111
Filename 'uImage-3.9'.
Load address: 0x31000000
Loading: checksum bad
T #################################################################
         #################################################################
         #################################################################
         ############################checksum bad
#####################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         ############################################################
done
Bytes transferred = 2933888 (2cc480 hex)
MINI2440 # nand erase kernel

NAND erase: device 0 offset 0x60000, size 0x500000
Erasing at 0x540000 -- 100% complete.
OK
MINI2440 # nand write 0x31000000 kernel 2933888

NAND write: device 0 offset 0x60000, size 0x500000
 5242880 bytes written: OK
MINI2440 #
```

- **Boot linux to the target with nfs support as previous demonstration**

```
[veda@mini2440:/]# ls
bin      etc      linuxrc  mnt      rfs      sbin     usr
dev      lib      media    proc     root     sys
[veda@mini2440:/]#
```

- **After booting:**

- We get rfs directory in the target root tree.

- Nand flash device supports few file-systems like UBIFS, Jffs2, Yaffs2 etc.

- To see how many nand partitios supported by linux give the following command:

[veda@mini240:/]# **cat /proc/mtd**

```
[veda@mini2440:/]# cat /proc/mtd
dev:    size   erasesize  name
mtd0: 00040000 00020000 "u-boot"
mtd1: 00020000 00020000 "u-boot-env"
mtd2: 00500000 00020000 "kernel"
mtd3: 3faa0000 00020000 "root"
[veda@mini2440:/]#
```

- These mtdpartitions suppoerts both character drivers and block drivers

[veda@mini240:/]# **ls -l /dev/mtd***

```
[veda@mini2440:/]#  ls -l /dev/mtd*
crw-rw----    1 veda     0          90,   0 Mar 11 10:50 /dev/mtd0
crw-rw----    1 veda     0          90,   1 Mar 11 10:50 /dev/mtd0ro
crw-rw----    1 veda     0          90,   2 Mar 11 10:50 /dev/mtd1
crw-rw----    1 veda     0          90,   3 Mar 11 10:50 /dev/mtd1ro
crw-rw----    1 veda     0          90,   4 Mar 11 10:50 /dev/mtd2
crw-rw----    1 veda     0          90,   5 Mar 11 10:50 /dev/mtd2ro
crw-rw----    1 veda     0          90,   6 Mar 11 10:50 /dev/mtd3
crw-rw----    1 veda     0          90,   7 Mar 11 10:50 /dev/mtd3ro
brw-rw----    1 veda     0          31,   0 Mar 11 10:50 /dev/mtdblock0
brw-rw----    1 veda     0          31,   1 Mar 11 10:50 /dev/mtdblock1
brw-rw----    1 veda     0          31,   2 Mar 11 10:50 /dev/mtdblock2
brw-rw----    1 veda     0          31,   3 Mar 11 10:50 /dev/mtdblock3
[veda@mini2440:/]#
```

character driver - for erasing purpose
block driver - for mounting and data manipulations.

- We copy our root filesystem (rfs directory tree) into root partition by using corresponding block-device in the target.

Flashing rootfs hierarchy as JFFS2 filesystem:

- For this linux kernel uImage must support this flash filesystem.
- See the kernel documemtation to select Jffs2 filesystem
- Boot linux to the target with nfs support as previous.
- Giving following command to see the filesystem suppot in target.

[veda@mini240:/]# **cat /proc/filesystems**

```
        msdos
nodev   nfs
nodev   nfs4
nodev   jffs2
        romfs
nodev   autofs
        yaffs
        yaffs2
nodev   mqueue
nodev   mtd_inodefs
nodev   ubifs
[veda@mini2440:/]#
```

- Erase the root parttion(**mtd3**) of nand device

[veda@mini240:/]# **flash_eraseall /dev/mtd3**
/dev/mtd3 - character device for root partition.

- Mount the root partition on /mnt of type jffs2 filesystem as block device
[veda@mini240:/]# **mount -t jffs2 /dev/mtdblock3 /mnt**
/dev/mtdblock3 - block device for root partition.
- Copy rfs directory content into /mnt
[veda@mini240:/]# **cp -Rfp /rfs/* /mnt**
- Unmount root partition

[veda@mini240:/]# **umount /mnt**

- reboot the target system

[veda@mini240:/]# **reboot**

```
[veda@mini2440:/]#  flash_eraseall  /dev/mtd3
Erasing 128 Kibyte @ b9c0000 - 18% complete.
Skipping bad block at 0x0b9e0000
Erasing 128 Kibyte @ 2a220000 - 66% complete.
Skipping bad block at 0x2a240000
Erasing 128 Kibyte @ 3faa0000 - 100% complete.
[veda@mini2440:/]# mount -t jffs2 /dev/mtdblock3 /mnt
[veda@mini2440:/]# cp -Rfp /rfs/* /mnt
[veda@mini2440:/]# umount /mnt
[veda@mini2440:/]# reboot
```

- Set the u-boot environment variable for mounting root filesystem as root

MINI2440 # **setenv bootargs console=ttySAC0,115200 root=/dev/mtdblock3 rootfstype=jffs2**

- Load the uImage from nand to RAM and boot from memory.

MINI2440 # **nand read 0x31000000 kernel 2542144**
MINI2440 # **bootm 0x31000000**

```
MINI2440 #  setenv bootargs console=ttySAC0,115200 root=/dev/mtdblock3 rootfstype=jffs2
MINI2440 # nand read 0x31000000 kernel 2542144

NAND read: device 0 offset 0x60000, size 0x500000
 5242880 bytes read: OK
MINI2440 # bootm 0x31000000
## Booting kernel from Legacy Image at 31000000 ...
   Image Name:   Linux-3.9.0
   Created:      2014-03-11   4:28:06 UTC
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    2933824 Bytes =  2.8 MB
   Load Address: 30008000
   Entry Point:  30008000
   Verifying Checksum ...
```

```
VFS: Mounted root (jffs2 filesystem) on device 31:3.
Freeing init memory: 148K
dm9000 dm9000 eth0: link down
dm9000 dm9000 eth0: link up, 100Mbps, full-duplex, lpa 0xCDE1

##########################################
###                                    ###
###     Welcome to Veda SOlutions      ###
###                                    ###
###       login name : veda            ###
###       password   : veda            ###
###                                    ###
##########################################
mini2440 login: veda
Mar 11 11:36:46 login[799]: root login on 'ttySAC0'
[veda@mini2440:/root]# ls /
bin      etc      linuxrc  mnt      root     sys
dev      lib      media    proc     sbin     usr
[veda@mini2440:/root]#
```

<u>Auto Booting Process For Jffs2 From U-BOOT :</u>

MINI2440 # **setenv bootcmd nand read 0x31000000 kernel 2542144 \; bootm 0x31000000**
MINI2440 # **setenv bootargs console=ttySAC0,115200 root=/dev/mtdblock3 rootfstype=jffs2**
MINI2440 # **saveenv**
MINI2440 # **reset**

Flashing rootfs hierarchy as YAFFS2 filesystem:

- For this linux kernel uImage must support this flash filesystem.
- See the kernel documemtation to select Yaffs2 filesystem
- Boot linux to the target with nfs support as previous.
- Giving following command to see the filesystem suppot in target.

[veda@mini240:/]# **cat /proc/filesystems**

```
nodev   nfs
nodev   nfs4
nodev   jffs2
        romfs
nodev   autofs
        yaffs
        yaffs2
nodev   mqueue
nodev   mtd_inodefs
nodev   ubifs
[veda@mini2440:/]#
```

- Erase the root parttion(**mtd3**) of nand device

[veda@mini240:/]# **flash_eraseall /dev/mtd3**

/dev/mtd3 - character device for root partition.

- Mount the root partition on /mnt of type Yaffs2 filesystem as block device

[veda@mini240:/]# **mount -t yaffs2 /dev/mtdblock3 /mnt**

/dev/mtdblock3 - block device for root partition.

- Copy rfs directory content into /mnt

[veda@mini240:/]# **cp -Rfp /rfs/* /mnt**

- Unmount root partition

[veda@mini240:/]# **umount /mnt**

- reboot the target system

[veda@mini240:/]# **reboot**

```
[veda@mini2440:/root]# flash_eraseall  /dev/mtd3
Erasing 128 Kibyte @ b9c0000 - 18% complete.
Skipping bad block at 0x0b9e0000
Erasing 128 Kibyte @ 2a220000 - 66% complete.
Skipping bad block at 0x2a240000
Erasing 128 Kibyte @ 3faa0000 - 100% complete.
[veda@mini2440:/root]# mount -t yaffs2 /dev/mtdblock3 /mnt
yaffs: dev is 32505859 name is "mtdblock3" rw
yaffs: passed flags ""
[veda@mini2440:/root]# cp -Rfp /rfs/* /mnt
[veda@mini2440:/root]# umount /mnt
[veda@mini2440:/root]# reboot
```

- Set the u-boot environment variable for mounting root filesystem as root
MINI2440 # **setenv bootargs console=ttySAC0,115200 root=/dev/mtdblock3 rootfstype=yaffs2**

- Load the uImage from nand to RAM and boot from memory.
MINI2440 # **nand read 0x31000000 kernel 2542144**
MINI2440 # **bootm 0x31000000**

```
MINI2440 # nand read 0x31000000 kernel 2542144

NAND read: device 0 offset 0x60000, size 0x500000
 5242880 bytes read: OK
MINI2440 # setenv bootargs console=ttySAC0,115200 root=/dev/mtdblock3 rootfstype=yaffs2
MINI2440 #  bootm 0x31000000
## Booting kernel from Legacy Image at 31000000 ...
   Image Name:   Linux-3.9.0
   Created:      2014-03-11   4:28:06 UTC
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    2933824 Bytes =  2.8 MB
   Load Address: 30008000
   Entry Point:  30008000
   Verifying Checksum ... OK
   Loading Kernel Image ...
```

```
yaffs: dev is 32505859 name is "mtdblock3" rw
yaffs: passed flags ""
VFS: Mounted root (yaffs2 filesystem) on device 31:3.
Freeing init memory: 148K
dm9000 dm9000 eth0: link down
dm9000 dm9000 eth0: link up, 100Mbps, full-duplex, lpa 0xCDE1

##########################################
###                                    ###
###     Welcome to Veda SOlutions      ###
###                                    ###
###       login name : veda            ###
###       password   : veda            ###
###                                    ###
##########################################
mini2440 login: veda
Mar 11 13:12:28 login[799]: root login on 'ttySAC0'
[veda@mini2440:/root]# ls /
bin         etc         linuxrc     media       proc        sbin        usr
dev         lib         lost+found  mnt         root        sys
[veda@mini2440:/root]#
```

<u>Auto Booting Process For Yaffs2 From U-BOOT :</u>
MINI2440 # **setenv bootcmd nand read 0x31000000 kernel 2542144 \; bootm 0x31000000**
MINI2440 # **setenv bootargs console=ttySAC0,115200 root=/dev/mtdblock3 rootfstype=yaffs2**
MINI2440 # **saveenv**
MINI2440 # **reset**

Flashing rootfs hierarchy as UBIFS filesystem:

- For this linux kernel uImage must support this flash filesystem.
- See the kernel documemtation to select UBIFS filesystem
- Boot linux to the target with nfs support as previous.
- Giving following command to see the filesystem suppot in target.

[veda@mini240:/]# **cat /proc/filesystems**

```
nodev   nfs
nodev   nfs4
nodev   jffs2
        romfs
nodev   autofs
        yaffs
        yaffs2
nodev   mqueue
nodev   mtd_inodefs
nodev   ubifs
```

- Erase the root parttion(**mtd3**) of nand device

[veda@mini240:/]# **flash_eraseall /dev/mtd3**

/dev/mtd3 - character device for root partition.

```
[veda@mini2440:/root]# flash_eraseall  /dev/mtd3
Erasing 128 Kibyte @ b9c0000 - 18% complete.
Skipping bad block at 0x0b9e0000
Erasing 128 Kibyte @ 2a220000 - 66% complete.
Skipping bad block at 0x2a240000
Erasing 128 Kibyte @ 3faa0000 - 100% complete.
[veda@mini2440:/root]#
```

- For UBIFS we need ubifs tools

```
[veda@mini2440:/root]# ubi
ubiattach     ubimkvol      ubirsvol
ubidetach     ubirmvol      ubiupdatevol
```

- Initially we don't have UBI device files and no kernel data for ubi device in **/sys** directory.

```
[veda@mini2440:/root]# ls /dev/u*
/dev/ubi_ctrl  /dev/urandom
```

- ubiattach command create a device file for given mtdpartition and maintain kernel data for tht mtdpartition.

[veda@mini240:/] # **ubiattach -m 3 /dev/ubi_ctrl**

```
[veda@mini2440:/root]#  ubiattach -m 3 /dev/ubi_ctrl
UBI: attaching mtd3 to ubi0
UBI: scanning is finished
UBI: empty MTD device detected
UBI: attached mtd3 (name "root", size 1018 MiB) to ubi0
UBI: PEB size: 131072 bytes (128 KiB), LEB size: 129024 bytes
UBI: min./max. I/O unit sizes: 2048/2048, sub-page size 512
UBI: VID header offset: 512 (aligned 512), data offset: 2048
UBI: good PEBs: 8147, bad PEBs: 2, corrupted PEBs: 0
UBI: user volume: 0, internal volumes: 1, max. volumes count: 128
UBI: max/mean erase counter: 0/0, WL threshold: 4096, image sequence number: 878249021
UBI: available PEBs: 7985, total reserved PEBs: 162, PEBs reserved for bad PEB handling: 8
UBI: background thread "ubi_bgt0d" started, PID 810
[veda@mini2440:/root]#
```

- Create a volume of size **64M** for the device (**ubi0**) as **rootfs1** for root filesystem

Usage: ubimkvol UBI_DEVICE -N NAME -s SIZE (size should be in bytes)

[veda@mini240:/]# **ubimkvol /dev/ubi0 -N rootfs1 -s 67108864**

- Recheck system ubi info in **/dev/** , **/sys/class/ubi**

```
[veda@mini2440:/root]# ls /dev/u*
/dev/ubi0       /dev/ubi0_0     /dev/ubi_ctrl   /dev/urandom
```

```
[veda@mini2440:/root]# ls /sys/class/ubi/
ubi0      ubi0_0    version
```

- Mount the ubi0 device volume1 on /mnt of type UBIFS filesystem

[veda@mini240:/]# **mount -t ubifs ubi0:rootfs1 /mnt**

/dev/mtdblock3 - block device for root partition.

- Copy rfs directory content into /mnt

[veda@mini240:/]# **cp -Rfp /rfs/* /mnt**

- Unmount root partition

[veda@mini240:/]# **umount /mnt**

- reboot the target system

[veda@mini240:/]# **reboot**

```
[veda@mini2440:/root]#  mount -t ubifs ubi0:rootfs1 /mnt
UBIFS: default file-system created
UBIFS: background thread "ubifs_bgt0_0" started, PID 819
UBIFS: mounted UBI device 0, volume 0, name "rootfs1"(null)
UBIFS: LEB size: 129024 bytes (126 KiB), min./max. I/O unit sizes: 2048 bytes/2048 bytes
UBIFS: FS size: 65931264 bytes (62 MiB, 511 LEBs), journal size 3354624 bytes (3 MiB, 26 )
UBIFS: reserved for root: 3114096 bytes (3041 KiB)
UBIFS: media format: w4/r0 (latest is w4/r0), UUID A24B518C-A98E-4326-A09F-D3717EDFACFC, l
[veda@mini2440:/root]# cp -Rfp /rfs/* /mnt
[veda@mini2440:/root]# umount /mnt
UBIFS: un-mount UBI device 0, volume 0
UBIFS: background thread "ubifs_bgt0_0" stops
[veda@mini2440:/root]# reboot
```

- Set the u-boot environment variable for mounting ubi device (ubi0) volume rootfs1 as root

MINI2440 **# setenv bootargs console=ttySAC0,115200 root=ubi0:rootfs1 ubi.mtd=3**

rootfstype=ubifs

root	-	ubi_device:volume_name
ubi.mtd	-	mtd_partition_number
rootfstype	-	ubifs

- Load the uImage from nand to RAM and boot from memory.

MINI2440 # **nand read 0x31000000 kernel 2542144**

MINI2440 # **bootm 0x31000000**

```
MINI2440 # setenv bootargs console=ttySAC0,115200 root=ubi0:rootfs1 ubi.mtd=3 rootfstype=ubifs
MINI2440 # nand read 0x31000000 kernel 2542144

NAND read: device 0 offset 0x60000, size 0x500000
 5242880 bytes read: OK
MINI2440 # bootm 0x31000000
## Booting kernel from Legacy Image at 31000000 ...
   Image Name:   Linux-3.9.0
   Created:      2014-03-11   4:28:06 UTC
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    2933824 Bytes =  2.8 MB
   Load Address: 30008000
   Entry Point:  30008000
   Verifying Checksum ... OK
   Loading Kernel Image ... OK
OK

Starting kernel ...

Uncompressing Linux... done, booting the kernel.
```

```
UBI: attaching mtd3 to ubi0
UBI: scanning is finished
UBI: attached mtd3 (name "root", size 1018 MiB) to ubi0
UBI: PEB size: 131072 bytes (128 KiB), LEB size: 129024 bytes
UBI: min./max. I/O unit sizes: 2048/2048, sub-page size 512
UBI: VID header offset: 512 (aligned 512), data offset: 2048
UBI: good PEBs: 8147, bad PEBs: 2, corrupted PEBs: 0
UBI: user volume: 1, internal volumes: 1, max. volumes count: 128
UBI: max/mean erase counter: 2/1, WL threshold: 4096, image sequence number: 878249021
UBI: available PEBs: 7464, total reserved PEBs: 683, PEBs reserved for bad PEB handling: 158
UBI: background thread "ubi_bgt0d" started, PID 787
input: gpio-keys as /devices/platform/gpio-keys/input/input0
s3c-rtc s3c2410-rtc: setting system clock to 2014-03-11 13:49:32 UTC (1394545772)
ALSA device list:
  No soundcards found.
UBIFS: background thread "ubifs_bgt0_0" started, PID 792
UBIFS: mounted UBI device 0, volume 0, name "rootfs1"(null)
UBIFS: LEB size: 129024 bytes (126 KiB), min./max. I/O unit sizes: 2048 bytes/2048 bytes
UBIFS: FS size: 65931264 bytes (62 MiB, 511 LEBs), journal size 3354624 bytes (3 MiB, 26 LEBs)
UBIFS: reserved for root: 3114096 bytes (3041 KiB)
UBIFS: media format: w4/r0 (latest is w4/r0), UUID A24B518C-A98E-4326-A09F-D3717EDFACFC, small LPT model
VFS: Mounted root (ubifs filesystem) on device 0:11.
Freeing init memory: 148K
dm9000 dm9000 eth0: link down
dm9000 dm9000 eth0: link up, 100Mbps, full-duplex, lpa 0xCDE1

########################################
###                                  ###
###     Welcome to Veda SOlutions    ###
###                                  ###
###       login name : veda          ###
###       password   : veda          ###
###                                  ###
########################################
mini2440 login: veda
Mar 11 13:49:42 login[805]: root login on 'ttySAC0'
[veda@mini2440:/root]# ls /
bin      etc      linuxrc  mnt      root     sys
dev      lib      media    proc     sbin     usr
```

Auto Booting Process For UBIFS From U-BOOT :

MINI2440 # **setenv bootcmd nand read 0x31000000 kernel 2542144 \; bootm 0x31000000**
MINI2440 # **setenv bootargs console=ttySAC0,115200 root=ubi0:rootfs1 ubi.mtd=3 rootfstype=ubifs**
MINI2440 # **saveenv**
MINI2440 # **reset**

MMC Booting:

Setup For Host System:

- Inset MMC card into system slot

- Unmount all prevoius partions

- Create required partitions of required size using "fdisk" command or "gparted" apt-package, both are dangerous becarefull!

    veda@linux # **fdisk /dev/mmcblk0** (Here i am using fdisk)

    -create partitions as below:
        64M ---> for kernel
        1G ---> for root-fiesystem

- After creating partitions check at **/dev** directoty then we find two device files
    /dev/mmcblk0p1 (used for 64MB kernel partition)
    /dev/mmcblk0p2 (used for 1GB rootfs partition)

- By using following command to create **ext2/fat** filesystem for 1st partition (ext2/fat because uboot support these two filesystems)
    veda@linux # **mkfs.ext2 /dev/mmcblk0p1** (for ext2 fs)

        OR

    ved@linux # **mkfs.vfat -F32 /dev/mmcblk0p1** (for fat32 fs)

- Creating filesystem for 2nd partition (i.e for root filesystem) is our wish (i.e ext2/ext3/ext4/fat). Here we are using ext3 for this, by using following command

    veda@linux # **mkfs.ext3 /dev/mmcblk0p2**

- Mount those partitions into your desired mount points by using "**mount**" command.

- Copy the **kernel uImage** into **1st** parttion and copy **root file-system** into **2nd** partition.

- Safely unmount those partitions using "**umount**" command.

Setup For Target System:

- Insert mmc card into target board mmc slot

    MINI2440 # **mmcinit** (This command initialize the mmc)

- Set bootargs as following, Here **rootfilesysyem type ext3** because as per the created filesystem type for 2nd partition in mmc

    MINI2440 # **setenv bootargs console=ttySAC0,115200 rootfstype=ext3**
        root=/dev/mmcblk0p2

- Copy the uImage from mmc kernel (1st) partition to RAM, by using following command
  Here '**mmc**' is device name, **0:1**--->means Device 0 and Partition 1

MINI2440 # **ext2load mmc 0:1 0x31000000 uImage** (if we are using ext2 for 1st partition)

OR

MINI2440 # **fatload mmc 0:1 0x31000000 uImage** (if we are using fat for ist partition)

- Start the booting using following command

  MINI2440 # **bootm 0x31000000**

Auto Booting Process For MMC From U-BOOT :

MINI2440 # **setenv bootcmd mmcinit \; ext2load mmc 0:1 0x31000000 uImage \; bootm 0x31000000**
MINI2440 # **setenv bootargs console=ttySAC0,115200 rootfstype=ext3 root=/dev/mmcblk0p2**
MINI2440 # **saveenv**
MINI2440 # **reset**

----------< **THE-END** >----------