Gömülü Linux Sistemleri

Login'e Kadar Linux

Beaglebone İle Çalışmak, 1. Bölüm

Giriş

Bu yazı dizimizde, şu anda çok yaygın olan Beaglebone cihazı üzerinde çeşitli gömülü linux çalışmaları yapılacaktır. Yazı dizisi 4 bölüm şeklinde yayınlanacaktır. Her bölümde cihaz üzerine farklı tekniklerle gömülü sistemler kurulacaktır.

Burada amacımız, Angstrom veya Ubuntu gibi dört başı mağrur, havada takla atabilen bir sistem kurmak değildir. Yapılan işlerin FARKINDA olarak yapılmasını sağlamak gibi çok basit bir amacımız vardır.

Bir önceki yazımızda gömülü sistemlerle ilgili pek çok kavrama değindiğimiz için bu yazıda aynı kavramlar tekrar edilmeyecektir. Fakat tekrar edilmesinde yarar gördüğümüz bir iki cümle vardır…

Bu yazı dizisinde geçen hiç bir konu Linux dağıtımına veya paket yöneticisine bağlı değildir. Burada bahsi geçen konular bilgisayar başında denenirken automounter özelliğinin kapatılması ve Linux'un güzel yüzü metin tabanlı ekrandan çalışılması tavsiye edilir. Ayrıca her ne kadar Beaglebone cihazı üzerinde çalışsak da, bahsi geçen bütün konular diğer bordlara da çok az değişiklikle uygulanabilir.

Bir önceki yazımız Beagleboard üzerine idi. Bu cihazın Begalebone'a göre en büyük özelliği içinde flash diskinin bulunmasıdır. Bu sayede bir önceki yazıda flash disk ile ilgili pek çok kavrama değinildi ve uygulama yapıldı. Beaglebone üzerinde flash disk yoktur. Bundan dolayı bu yazı dizisinde flash disk ile ilgili hiç bir uygulama yapılmayacaktır.

Üzerinde flash disk olmamasına rağmen Beaglebone cihazının çok güzel bir özelliği mevcuttur. Üzerinde eternet kartı mevcuttur. Böylece daha önceki yazıda hiç bahsedilmeyen ağ destekli çalışmalardan ve ideal test sisteminin kurulmasından bu yazı dizisinde bahsedilecektir.

1. Cihazın Satın Alınması

Elimizde Beaglebone cihazı mevcut değil. Satın almamız gerekti. Bu cihazı hayrına satan Çizgi Tagem'de de hiç kalmamış. Türkiye'de satan başka bir yer de bulamadık. Sonunda tr.farnell.com adresinden, kredi kartı ile bir adetlik sipariş girdik. Siparişimizin alındığına dair e-posta geldi. Pek mutlu olduk. Cihazın gelmesini beklemeye başaldık. Ertesi gün, kırık bir Tükçe ile İngiltere'den bir hanımefendi aradı. Aramızda aynen aşağıdaki gibi görüşme geçti.

- (Telefon çalar) Efendim Nazım.

- Ben İngiltere'den arıyorum. Siz bir bord siparişi vermişsiniz.

- Doğrudur, buyurun.

- Bu bord stratejik bir cihazdır. Bu bordu satın almak istiyorsanız size bir kaç sayfalık belge göndereceğiz. Bu belgeye, bordu hangi amaçla kullanacağınızı açıkça yazacaksınız. Sonra biz bu belgeleri inceleyeceğiz ve bordun gönderilmesinin uygun olup olmadığına karar vereceğiz. Eğer bordu gönderecek olursak, belgeleri de yanına ekleyeceğiz. Sizler, şahsen gümrükten gidip bordu almanız gerekiyor. Kabul ediyor musunuz?

- Hayır etmiyorum, siparişi iptal edin lütfen.

- İptal edilmiştir (telefon kapandı).

Bu arada Çizgi Tagem tekrar bord satmaya başladı. Bizler de bu çok değerli stratejik malzemeyi, yol parası dahil 104 liradan satın aldık. Verdikleri bu hizmetten dolayı kendilerine çok teşekkür ederiz.

Avrupa kaynaklı bazı web sitelerinden de ilgili cihaz rahatlıkla ve hiç bir kısıt olmadan sipariş edilebilmektedir. Bu kadar rahat bulunan bir cihaz için İngilizlerin niye bu kadar salakça bir politika uyguladıklarını anlamış değilim. Şimdi sadede gelelim.

2. İlk İnceleme

Elimizde bulunan cihaz BegaleBone Rev A6 sürümüdür. Bu yazı yazılırken Rev6A sürümü çıkmıştır. Sürümler arasındaki farklar donanım ile ilgilidir. Bizler cihazın donanım tarafı ile ilgili değiliz. Sürümler arasındaki farklar bu adresten incelenebilir.

Bu yazı dizisinde bahsi geçen bütün konular diğer Beaglebone sürümlerine de uygulanabilir.

Cihaz satın alındığında bir veya iki adet MMC kart ile gelmektedir. Bu kartlardan birinde çalışmaya hazır Angstrom Linux sistemi bulunmaktadır. Öncelikle böyle bir sistem hemen çalıştırılmalı ve Linux tarafı için gerekli olan çok önemli bazı parametreler bir köşeye not edilmelidir ki kendi Linux sistemimizi kurarken saçımızı başımızı yolmayalım.

Öncelikle masaüstü veya host sisteminde herhangi bir Linux dağıtımının kurulu olduğu kabul edilmektedir. Beaglebone cihazından bazen cihaz bazen de bord olarak bahsedilecektir. Standard bir Linux dağıtımının kurulu olduğu makine ise bazen host bazen masaüstü olarak adlandırılacaktır.

Beaglebone cihazı, Beagleboard'dan farklı olarak doğrudan USB üzerinden beslenmektedir. Ayrıca bu USB ile de dışarıyla bağlantı kurmaktadır. Cihaz, USB ile host'a bağlandıktan sonra aşağıdaki gibi lsusb ile ilk inceleme yapılabilir.

```
$ lsusb

Bus 002 Device 032: ID 0403:6010 Future Technology Devices Int
Bus 002 Device 034: ID 0525:a4a5 Netchip Technology, Inc. Linu
```

Netchip satırı biraz geç gelir.

minicom ile aşağıdaki gibi terminal açılır ve /dev/ttyUSB2, 115200, 8n1 tanımları minicom'a girilir.

```
$ minicom -s
```

Benim notebook'a pek çok USB cihazı bağlıdır. Bundan dolayı cihaz ismi olarak /dev/ttyUSB2 kullanılmıştır. Bu cihazlar çıkarıldığında bu isim /dev/ttyUSB1 veya /dev/ttyUSB0 olabilir. Okuyucu deneme ile bulabilir.

Ya da cihaz ismini kesin olarak bulabilmek için, cihazı takmadan önce aşağıdaki komut girilir.

```
$ ls -l /dev/ttyUSB*
```

Cihaz, yani Beaglebone takıldıktan sonra aynı komut tekrar girilir. Beaglebone cihazının adı yeni bir USB numarası ile belirecektir.

minicom ile terminal açıldıktan sonra lsof komutu (list open files) /dev/ttyUSB2'nin durumu aşağıdaki gibi incelenebilir.

```
# lsof /dev/ttyUSB2
 COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF  NODE NAME
 minicom 4017 root    3u   CHR  188,2      0t0 79271 /dev/ttyU
```

/dev/ttyUSB2'nin minicom tarafından kullanıldığı açıkça bellidir. 188,2 sayıları ise /dev/ttyUSB2'nin majör ve minör numaralarıdır. Dikste bulunan her dosya için bir inode tahsis edilir. 79271 sayısı, /dev/ttyUSB2 için ayrılan inode yapısının indeks değeridir. Herhangi bir dosyanın inode değeri aşağıdaki gibi ls komutu ile bulunabilir.

```
# ls -li /dev/ttyUSB2
79271 crw-rw---- 1 root dialout 188, 2 Nov  8 14:08 /dev/ttyUS
```

df komutu ile bakıldığında, en sağ kolonda, “Mounted on” yazan kolonda, pek çok dizin adı bulunur. Eğer “$ ls -lid dizin_adı” komutu ile incelenirse, bütün bağlantı dizinlerinin inode değerinin 2 olduğu görülebilir. Aslında bütün bunların konu ile pek bir ilgisi yok, yine konudan saptık.

minicom ile cihaza bağlandığımızı ve hazır Linux yüklü MMC ile cihazın açıldığını kabul edelim. Hemen root ile giriş yapılır. Aşağıdaki gibi bazı incelemeler yapılabilir.

```
root@beaglebone:~# free
        total       used       free     shared    buffers
Mem:    254080      66532     187548          0       8432
```

free komutu ile mevcut sistemin ne kadar RAM bellek kullandığı tespit edilebilir. swap olmadığı için burada ram ve sanal bellek miktarı birbirine denk olacaktır. Çıkıştan da görüleceği gibi 256MB’lık belleğin yaklaşık 65MB’lık bölümü kullanılmaktadır. Gerçek bir gömülü sistemde, yani belirli bir işi yapmak üzere tasarlanmış gömülü sistemde bu kadar büyük bellek kullanımı gereksizdir. Angstrom sistemi çok genel olduğu için aynı anda pek çok programı çalıştırmaktadır. Eğer Angstrom sistemi bir projede kullanılacaksa, mutlaka açılış betikleri incelenmeli ve amaca uygun programlar seçilmelidir. Yoksa sistem şu anki hali bir tam bir masaüstü gibidir. Ayrıca kök dosya sistemini read/write bağlamak gibi çok büyük bir sakıncaya sahiptir.

Sistem üzerindeki MMC kartı aşağıdaki gibi, fdisk komutu ile incelenebilir.

```
root@beaglebone:/proc# fdisk -l
```

```
Disk /dev/mmcblk0: 3904 MB, 3904897024 bytes
 255 heads, 63 sectors/track, 474 cylinders, total 7626752 sec
 Units = sectors of 1 * 512 = 512 bytes
 Sector size (logical/physical): 512 bytes / 512 bytes
 I/O size (minimum/optimal): 512 bytes / 512 bytes
 Disk identifier: 0x00000000

Device Boot          Start      End          Blocks   Id  Syste
 /dev/mmcblk0p1  *      63      144584       72261    c  W95 F
 /dev/mmcblk0p2     144585     7132859     3494137+  83  Linux
```

Kendi MMC katımızı hazırlayacağımız için, buradan hemen birkaç kopya çekebiliriz. Cihaz her zaman 1. bölümden açılır. 1. bölüm, c tipine sahiptir. FAT32 ile formatlanmıştır. Aynı zamanda * ile işaretlenmiştir. Yani bootable bir bölümdür. Bu bilgiler hayati öneme sahiptir.

Birinci bölümün içinde ne vardır? Birinci bölüm aşağıdaki komutlar ile mount edilerek incelenebilir.

```
root@beaglebone:/proc# mkdir /mnt/vfat

root@beaglebone:/proc# mount /dev/mmcblk0p1 /mnt/vfat

root@beaglebone:/proc# ls -l /mnt/vfat
 total 3922
 drwxr-xr-x 4 root root    2048 May 16 15:29 Docs
 drwxr-xr-x 5 root root    2048 May 16 15:29 Drivers
 -rwxr-xr-x 1 root root    5829 May 16 06:48 LICENSE.txt
 -rwxr-xr-x 1 root root   37151 May  8 08:16 MLO
 -rwxr-xr-x 1 root root   13901 May 16 06:48 README.htm
 -rwxr-xr-x 1 root root     178 May 16 06:48 autorun.inf
 -rwxr-xr-x 1 root root     171 May 16 06:48 info.txt
 -rwxr-xr-x 1 root root  524288 Jun 18 11:04 test-file
 -rwxr-xr-x 1 root root  239744 May  8 08:16 u-boot.img
 -rwxr-xr-x 1 root root      33 May  8 08:16 uEnv.txt
 -rwxr-xr-x 1 root root 3179944 May  8 08:16 uImage
```

Burada bizim için hayati öneme sahip 4 adet dosya bulunur. Bu dosyalar ilerleyen bölümlerde tek tek, kaynak kodlarından derlenerek elde edilecektir. Bu dosyaları kısaca inceleyecek olursak…

- **MLO** Birinci boot yükleyicisidir. Beaglebone açılır açılmaz, MMC kartın birinci bölümünde bulunan MLO dosyasını yükler ve çalıştırır. MLO pek yetenekli değildir. Çekirdek veya dosya sistemi yükleyemez. Bu iş için u-boot gibi daha yenetekli bir yükleyiciye ihtiyaç duyar.

- **u-boot.img** Çok yetenekli ve ARM makineler için neredeyse standard hale gelmiş, ikinci seviye boot yükleyicisidir. MLO tarafından belleğe yüklenir.
- **uEnv.txt** U-Boot sisteminin çevre değişkenlerini saklar.
- **uImage** U-Boot imajı haline getirilmiş Linux çekirdeğinin imajıdır. Basitçe 64 baytlık başlık bilgisi ve zImage dosyasından oluşur.

MMC kartı VFAT olarak formatlayıp bu 4 dosyayı karta kopyalarsak, Linux sistemi kernel seviyesine kadar açılacaktır.

Şimdi reboot diyerek cihazı tekrar başlatalım. Ya da USB'yi çekip tekrar takalım ya da cihaz üzerindeki reset düğmesine basalım. Her durumda cihaz baştan açılacaktır. Açılış, bir tuşa basılarak U-Boot seviyesinde durdurulur. Bu durumda aşağıdaki gibi bir ekran çıkışı elde edilecektir.

```
U-Boot SPL 2011.09-00000-gf63b270-dirty (Apr 24 2012 - 09:51:0

 Texas Instruments Revision detection unimplemented
 No AC power, disabling frequency switch
 OMAP SD/MMC: 0
 reading u-boot.img
 reading u-boot.img

U-Boot 2011.09-00000-gf63b270-dirty (Apr 24 2012 - 09:51:01)

I2C:   ready
 DRAM:  256 MiB
 No daughter card present
 NAND:  HW ECC Hamming Code selected
 No NAND device found!!!
 0 MiB
 MMC:   OMAP SD/MMC: 0
 *** Warning - readenv() failed, using default environment

Net:   cpsw
 Hit any key to stop autoboot:  0

U-Boot# help
```

help girişi ile, desteklenen komutların listesi elde edilebilir. Her bordun desteklemiş olduğu komut kümesi farklıdır. Örneğin Beaglebone sisteminde nand mevcut değildir. Bundan dolayı nand ile ilgili bütün komutlar, u-boot programından çıkarılmıştır. Mevcut olsa bile işlemeyecektir. Çünkü cihazda NAND yoktur.

U-Boot çevre değişkenlerinin host tarafında saklanması tavsiye edilir. Saklama işi copy/paste ile veya history özelliği ile yapılabilir. Bazı değişkenlerin tanımları minicom

ekranına tam sığmayabilir. Uzun satırların tamamını görebilmek için, minicom'da Ctrl+Alt+W ile "Linewrap Off" yapılmalıdır.

Mevcut sistemin çevre değişkenleri tanımları aşağıda listelenmiştir.

```
U-Boot SPL 2011.09-00000-gf63b270-dirty (Apr 24 2012 - 09:51:0
Texas Instruments Revision detection unimplemented
No AC power, disabling frequency switch
OMAP SD/MMC: 0
reading u-boot.img
reading u-boot.img

U-Boot 2011.09-00000-gf63b270-dirty (Apr 24 2012 - 09:51:01)

I2C:   ready
DRAM:  256 MiB
No daughter card present
NAND:  HW ECC Hamming Code selected
No NAND device found!!!
0 MiB
MMC:   OMAP SD/MMC: 0
*** Warning - readenv() failed, using default environment

Net:   cpsw
Hit any key to stop autoboot:  0
U-Boot# pr

autoload=yes
baudrate=115200
bootargs_defaults=setenv bootargs console=${console} ${optargs
bootcmd=if mmc rescan; then echo SD/MMC found on device ${mmc_
bootdelay=1
bootenv=uEnv.txt
bootfile=uImage
console=ttyO0,115200n8
ethact=cpsw
ethaddr=d4:94:a1:8e:65:5a
importbootenv=echo Importing environment from mmc ...; env imp
ip_method=none
loadaddr=0x82000000
loadbootenv=fatload mmc ${mmc_dev} ${loadaddr} ${bootenv}
mmc_args=run bootargs_defaults;setenv bootargs ${bootargs} roo
mmc_boot=run mmc_args; run mmc_load_uimage_ext4; bootm 0x80007
mmc_dev=0
mmc_load_uimage=fatload mmc ${mmc_dev}:1 0x80007fc0 ${bootfile
mmc_load_uimage_ext2=ext2load mmc ${mmc_dev}:2 0x80007fc0 /boo
mmc_load_uimage_ext4=ext4load mmc ${mmc_dev}:2 0x80007fc0 /boo
```

```
mmc_root=/dev/mmcblk0p2 ro
mmc_root_fs_type=ext4 rootwait
nand_args=run bootargs_defaults;setenv bootargs ${bootargs} ro
nand_boot=echo Booting from nand ...; run nand_args; nand read
nand_img_siz=0x500000
nand_root=/dev/mtdblock7 rw
nand_root_fs_type=jffs2
nand_src_addr=0x280000
net_args=run bootargs_defaults;setenv bootargs ${bootargs} roo
net_boot=echo Booting from network ...; setenv autoload no; dc
nfsopts=nolock
nor_args=run bootargs_defaults;setenv bootargs ${bootargs} roo
nor_boot=echo Booting from NOR ...; run nor_args; cp.b ${0x080
nor_img_siz=0x280000
nor_root=/dev/mtdblock3 rw
nor_root_fs_type=jffs2
nor_src_addr=0x08080000
rootpath=/export/rootfs
script_addr=0x81900000
spi_args=run bootargs_defaults;setenv bootargs ${bootargs} roo
spi_boot=echo Booting from spi ...; run spi_args; sf probe ${s
spi_bus_no=0
spi_img_siz=0x280000
spi_root=/dev/mtdblock4 rw
spi_root_fs_type=jffs2
spi_src_addr=0x62000
static_ip=${ipaddr}:${serverip}:${gatewayip}:${netmask}:${host
stderr=serial
stdin=serial
stdout=serial

Environment size: 2769/8188 bytes
U-Boot#
```

U-Boot sisteminde her zaman 2 temel değişken vardır. Bunlar bootargs ve bootcmd'dir. Diğer bütün değişkenler, bu iki değişkende toplanırlar. Diğer bir deyişle, diğer bütün değişkenler bu 2 değişken için, yardımcı değişken gibi davranırlar.

bootargs değişkeninin aslında u-boot ile bir ilgisi yoktur. Linux çekirdeğine aktarılacak bütün parametreler bu değişkene atanır. U-Boot sistemi, çekirdeği yüklerken, bootargs içinde ne varsa, gözü kapalı bir biçimde çekirdeğe aktarır. Diğer bir deyişle bootargs değişkeni u-boot için pasif bir değişkendir.

Diğer değişken ise bootcmd değişkenidir. U-Boot çevre değişkenlerini yükledikten sonra, gözü kapalı bir biçimde bootcmd içinde ne varsa çalıştırır. Bundan dolayı otomatik açılış ve diğer bütün işler için bootcmd kullanılır.

console tanımı da çok önemlidir. Bu bir çekirdek parametresidir. Yani açılışta u-boot tarafından doğrudan çekirdeğe aktarılır. Gömülü sistemlerle uğraşılırken karşılaşılan ve insanı kahreden hatalardan biri de konsola hiç bir mesajın gelmemesi veya login lafı geldikten sonra hiç bir girişin yapılamamasıdır. Bunun en büyük sebebi console tanımının eksik olmasıdır. Begalebone sisteminde konsol olarak /dev/ttyO0 tanıtılmıştır. tty'den sonra "O" harfi ve sıfır vardır. Ayrıca hız olarak 115200 tanıtılmıştır. Diğer seri kanal parametreleri n8 olarak, yani "no parity" ve "8 bit data" olarak verilmiştir. Eğer konsol tanımı hatalı girilirse, çekirdek mesajları gözükmez. Eğer konsole tanımı /etc/inittab içinde hatalı girilirse login gelse bile giriş yapılamaz. Konsole tanımı /etc/securetty içine girilmezse, dışarıdan telnet ile bağlantı yapılamaz. Kısaca konsol tanımı çok önemlidir. Her bord markasının farklı bir konsol ismi olabilir.

u-boot açıldığı zaman hemen bootcmd değişkeni içinde ne varsa çalıştırır. Kolay okunması için bootcmd tanımı, aşağıda satır satır verilmiştir. U-Boot sisteminin açılış tekniğini kavrayabilmek için aşağıdaki betiğin anlaşılması çok önemlidir. Aşağıdaki betik bootcmd ifadesinin sağ tarafına aittir. Bu değişken u-boot seviyesinde iken "pr bootcmd" girişi yapılarak ayrıca incelenebilir.

```
01 if mmc rescan; then

02   echo SD/MMC found on device ${mmc_dev};

03   if run loadbootenv; then
04      echo Loaded environment from ${bootenv};
05      run importbootenv;
06      fi;

07   if test -n $uenvcmd; then
08      echo Running uenvcmd ...;
09      run uenvcmd;
10      fi;

11   if run mmc_load_uimage_ext4; then
12      run mmc_args;
13      bootm 0x80007fc0;
14      fi;

15   fi;

16 run nand_boot;
```

Betiğin üzerinden kısaca geçelim.

01. satırda mmc rescan ile MMC'nin takılı olup olmadığı kontrol edilir. MMC yoksa 16. satıra düşülür. Bu satır NAND üzerinden sistemi açar. Ama Beaglebone sisteminde

NAND yoktur. Sonuçta bu satırın da bir anlamı yokur. Eğer sistemde MMC yoksa hiç bir iş yapılmaz ve u-boot promptu gelir.

“run A” nın çok basit bir işlevi vardır. A değişkeninin içindeki betik yürütülür. Buna göre 03..06. satırları arasında uEnv.txt dosyası belleğe yüklenir. Değişkenler yerlerine konularak takip edilirse betiğin yapacağı iş kolaylıkla tespit edilebilir.

07..10. satırlarında ise uenvcmd isimli değişkenin var olup olmadığına bakılır. Varsa bu değişken içindekiler çalıştırılır.

Buradan hemen çok basit bir sonuç çıkarılabilir. U-Boot tarafında çalışması istenilen komutlar, VFAT içindeki uEnv.txt dosyası içine yazılır. Bu komutların bir tanesi de uenvmcd olacaktır. U-Boot sistemi, hemen uEnv.txt dosyasını yükleyecek ve uenvcmd komutunu çalıştıracaktır.

Okuyucunun, betiği dikkatli bir biçimde incelemesi tavsiye edilir.

“bootm 0x80007fc0″ bilgisi de çok önemlidir. Çekirdeğin, VFAT’dan belleğe yükleneceği adresi belirtir. Bu adresin analizi yazı sonunda yapılacaktır. Şimdilik bir köşeye kaydedilmesi yeterlidir.

nand ile başlayan değişkenlerin bu cihaz için bir anlamı yoktur.

Ayrıca sistem açıldıktan sonra dmesg çıktısı saklanmalıdır. “Kernel command line” gibi önemli bilgiler içerir.

Angstrom ile gelen çekirdek hiç bir zaman olduğu gibi kullanılmamalıdır. Projeye uygun yeniden derlenmelidir. dmesg ile elde edilen çekirdek mesajları incelenirse, aslında sizin projenizde hiç kullanmayacağınız pek çok sürücü veya çekirdek özelliği olduğu görülebilir.

Mevcut sistemden yeteri kadar kopya çekildi. Şimdi Angstrom Linux sistemi bir kenara bırakılacak ve MLO’dan kök dosya sistemine kadar aradaki bütün işlemler el yordamı ile sıfırdan yapılacaktır.

Hazır çalışan bir sistem varken niçin böyle bir işe girişiyor, her işi el ile, sil baştan yapıyoruz. Biz mazoist miyiz? Her ne kadar Slackware kullanıyorsak da mazoist değiliz. Daha önce de tekrar ettiğimiz gibi tek bir amacımız vardır, yapılan işlerin farkında olmak.

Böylece önümüze gelen gömülü Linux projesi ile ilgili en doğru kararları alıp, projeye en uygun kuruluşu yapabileceğiz.

3. Çapraz Derleyici

MLO, U-Boot, Çekirdek ve Busybox'ın derlenmesi için çapraz derleyiciye gerek vardır. Beaglebone için tavsiye edilen bir kaç çapraz derleyici mevcuttur. Biz angstrom dağıtımının kullandığı derleyiciyi seçtik. Bu derleyici ve ilgili tool-chain, 32 bitlik x86 makineler için aşağıdaki gibi indirilip, kurulabilir.

```
$ wget http://www.angstrom-distribution.org/toolchains/angstro
$ mkdir arm-angstrom
$ cd arm-angstrom
$ tar jxvf /tmp/ftp/angstrom-2011.03-i686-linux-armv7a-linux-g
```

Bizler çapraz derleyiciyi /cross/arm-angstrom/ dizini altına açtık. Okuyucu kendine uygun bir dizin seçebilir. Çapraz derleyici /cross/arm-angstrom/usr/local/angstrom /arm/bin altındadır. Bu dizin PATH değişkenine eklenmelidir.

Test için derleyici ön-eki girildikten sonra, aşağıdaki gibi, 2 kez TAB tuşuna basılabilir. Eğer PATH değişkeni düzgün ayarlanmışsa, iki kez TAB'dan sonra, bütün tool-chain programları ekrana listelenecektir.

```
$ arm-angstrom-linux-gnueabi-  TAB TAB
```

Kullandığımız derleyicinin sürümü aşağıda verilmiştir.

```
$ arm-angstrom-linux-gnueabi-gcc --version

arm-angstrom-linux-gnueabi-gcc (GCC) 4.3.3
Copyright (C) 2008 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTIC
```

4. U-Boot

U-Boot'un son sürümü aşağıdaki gibi indirilip derlenebilir. Eğer okuyucu farklı bir çapraz derleyici kullanıyorsa derleyici ön-ekini, kendininkiyle değiştirmelidir.

```
$ wget ftp://ftp.denx.de/pub/u-boot/u-boot-latest.tar.bz2
$ tar jxvf /nk/blog/2/u-boot-latest.tar.bz2
$ ln -s /cross/u-boot-2012.10 u-boot
$ cd u-boot
$ make ARCH=arm CROSS_COMPILE=arm-angstrom-linux-gnueabi- am33
$ make ARCH=arm CROSS_COMPILE=arm-angstrom-linux-gnueabi-
```

U-Boot derlendiği zaman 3 temel program elde edilir. Bunlar sırası ile MLO, u-boot.img ve mkimage'dir. MLO, aslında u-boot'tan türetilmiştir. Bundan dolayı genelde u-boot ile birlikte bulunur. mkimage programı ise, dosyaları u-boot imajı haline getirmek için kullanılır. Bir önceki yazımızda bu konulardan çok fazla bahsettiğimizden burada ayrıca inceleme yapılmayacaktır.

mkimage programının /usr/local/bin gibi, genel bir yere kopyalanması tavsiye edilir. Aşağıda derleme sonunda elde edilen dosyalar ve kopyalama işlemi verilmiştir.

```
$ ls -l u-boot.img MLO tools/mkimage

-rw-r--r-- 1 root root  71882 Nov  8 21:47 MLO
-rwxr-xr-x 1 root root  56696 Nov  8 21:46 tools/mkimage*
-rw-r--r-- 1 root root 230496 Nov  8 21:47 u-boot.img

$ cp tools/mkimage /usr/local/bin/
```

u-boot/include/configs dosyası içinde, Beaglebone ile ilgili pek çok çevre değişkeni mevcuttur. Okuyucu bu dosyayı mutlaka incelemelidir. Ayrıca U-Boot'un varsayılan çevre değişkenleri burada tanımlıdır. Gömülü sistem projesinin nihai halinde, bootargs ve bootcmd burada tanıtılarak, uEnv.txt dosyası hiç kullanılmayabilir.

Aşağıda, config dosyasının tek bir satırının değiştirilmesi gösterilmiştir. Bizler her zaman olduğu gibi PROMPT ifadesini değiştirdik. Bu tür değişikliklerden sonra U-Boot'u tekrar derlemek gerekir, hatırlatalım.

```
$ cd u-boot/include/configs

$ vi am335x_evm.h

Burada varsayılan u-boot çevre değişkenleri bulunabilir.
Sadece aşağıdaki değişken yeniden atanmıştır.
#define CONFIG_SYS_PROMPT     "UcanLinux > "
```

5. MMC'nin Bölümlendirilmesi

Bizler kendi MMC kartımızı kendimiz kurmak istiyoruz. Bunun için önce MMC kartı bölümlere ayrılmalıdır. En az 2 bölüm olmalıdır. Birinci bölümde boot yükleyicileri ve çekirdek imajı bulunacaktır. 2. bölümde ise kök dosya sistemi oturacaktır.

Beaglebone sisteminin MMC'yi açılışta görebilmesi için c tipinde ve bootable işaretlenmesi yeterlidir. Bunun için herhangi bir MMC kartı masaüstü sisteme takılır ve aşağıdaki gibi bölümlendirme yapılır.

Aman dikkat! Bölümlendirme yaparken masaüstü sistemin diskini perişan etmeyin. MMC kartını masaüstüne taktıktan sonra fdisk -l veya dmesg|tail komutu ile takılan kartın ismini tam olarak tespit edin. Benim makinede MMC kartı /dev/sdb olarak gözükmektedir. fdisk ile aşağıdaki gibi bölümlendirme yapılır. Bir önceki yazıda fdisk'e geniş biçimde değinilmişti. Burada ayrıca bir açıklama yapılmayacaktır.

```
# fdisk -l

# fdisk /dev/sdb

Aman dikkat. Aşağıdaki komutu yanlış girmeyin!!!

# dd if=/dev/zero of=/dev/sdb count=1024 bs=1024

# fdisk /dev/sdb
 Device contains neither a valid DOS partition table, nor Sun,
 Building a new DOS disklabel with disk identifier 0xee385770.
 Changes will remain in memory only, until you decide to write
 After that, of course, the previous content won't be recovera

 Warning: invalid flag 0x0000 of partition table 4 will be cor

Command (m for help): n
 Command action
 e   extended
 p   primary partition (1-4)
 p
 Partition number (1-4, default 1):  ENTER
 Using default value 1
 First sector (2048-7716863, default 2048): ENTER
 Using default value 2048
 Last sector, +sectors or +size{K,M,G} (2048-7716863, default

Command (m for help): n
 Command action
 e   extended
 p   primary partition (1-4)
 p
 Partition number (1-4, default 2): ENTER
 Using default value 2
 First sector (133120-7716863, default 133120):  ENTER
 Using default value 133120
 Last sector, +sectors or +size{K,M,G} (133120-7716863, defaul
 Using default value 7716863

Command (m for help): t
```

```
 Partition number (1-4): 1
 Hex code (type L to list codes): c
 Changed system type of partition 1 to c (W95 FAT32 (LBA))

Command (m for help): a
 Partition number (1-4): 1

Command (m for help): p

Disk /dev/sdb: 3951 MB, 3951034368 bytes
 122 heads, 62 sectors/track, 1020 cylinders, total 7716864 se
 Units = sectors of 1 * 512 = 512 bytes
 Sector size (logical/physical): 512 bytes / 512 bytes
 I/O size (minimum/optimal): 512 bytes / 512 bytes
 Disk identifier: 0xee385770

Device Boot      Start         End      Blocks   Id  System
 /dev/sdb1   *    2048      133119       65536    c  W95 FAT32
 /dev/sdb2      133120     7716863     3791872   83  Linux

Command (m for help): w
 The partition table has been altered!
 Calling ioctl() to re-read partition table.

#
```

Burada farklı olarak bir adet dd komutu vardır. Bir blok 512 bayt olmak üzere, fdisk sistemi, ilk 2048 bloğu, yani ilk 1MB'lık bölümü boot yükleyicileri için ayırır. Bundan dolayı birinci bölüm her zaman 2048. bloktan başlar. Aynı zamanda fdisk sistemi ilk 2048 bloğa hiç dokunmaz. Sadece birinci bloğu yani MBR'yi günceller. Bizler garanti olsun diye ilk 2048 bloğu sıfırlıyoruz. Aslında sıfırlamasak da cihaz açılıyor. Fakat pek çok yerde bunun yapılması öneriliyor. Biz de geri kalmadık, yaptık.

6. Boot Yükleyicilerinin MMC'ye aktarılması

Daha önce MMC'nin 1. bölümünde MLO, u-boot.img, uEnv.txt ve uImage dosyalarının olması gerektiğinden bahsetmiştik. Henüz çekirdek derlemedik. U-Boot sistemini 4. bölümde derlediğimizden, elimizde MLO ve u-boot.img mevcuttur. Hemen bu ikisini kullanarak bordu u-boot seviyesine kadar açabiliriz. Bunun için 1. bölüme VFAT kurup her iki dosyayı bu bölüme kopyalamak yeterli olacaktır. Bu işlemler aşağıda verilmiştir. Eğer varsa, masaüstü sistemde automounter sisteminin kapatılması tavsiye edilir.

```
# mkdir /mnt/vfat

# mkfs.vfat -F32 -n "UcanLinux" /dev/sdb1
```

```
# mount /dev/sdb1 /mnt/vfat

# cp u-boot/MLO        /mnt/vfat
# cp u-boot/u-boot.img /mnt/vfat

# df /mnt/vfat
 Filesystem     1K-blocks  Used Available Use% Mounted on
 /dev/sdb1          64511   297     64215   1% /mnt/vfat

# ls -l /mnt/vfat
 total 296
 -rwxr-xr-x 1 root root  71882 Nov  9 19:52 MLO
 -rwxr-xr-x 1 root root 230500 Nov  9 19:52 u-boot.img

# umount /mnt/vfat
```

Burada birkaç hususa tekrar dikkat çekmek isteriz. Beaglebone sisteminin MMC'den açabilmesi için aşağıdaki özelliklerin sağlanması tavsiye edilir.

- MMC'nin 1. bölümü c tipinde, yani Win95,Fat32 olmalıdır.
- Bootable olmalıdır.
- 1. bölüm VFAT formatlanmalıdır. Formatlama yapılırken -F32 ile 32 bitlik fat tablosu seçilmelidir.
- MLO ilk dosya olarak kopyalanmalıdır.

7. İlk Açılış

MMC kartı masaüstünden çıkarılır ve borda takılır. USB kablosu masaüstü sistemine takılınca bord u-boot seviyesine kadar aşağıdaki gibi açılacaktır. Bordda artık kendi derlediğimiz boot yükleyicileri, MLO ve u-boot.img koşmaktadır.

```
U-Boot SPL 2012.10 (Nov 08 2012 - 21:58:26)
 OMAP SD/MMC: 0
 reading u-boot.img
 reading u-boot.img

U-Boot 2012.10 (Nov 08 2012 - 21:58:26)

I2C:   ready
 DRAM:  256 MiB
 WARNING: Caches not enabled
 MMC:   OMAP SD/MMC: 0, OMAP SD/MMC: 1
 Using default environment

Net:   cpsw
```

```
 Hit any key to stop autoboot:  0
 SD/MMC found on device 0
 reading uEnv.txt

** Unable to read "uEnv.txt" from mmc 0:1 **
 Loading file "/boot/uImage" from mmc device 0:2
 ** File not found /boot/uImage
 ext2load - load binary file from a Ext2 filesystem

Usage:
 ext2load <interface> <dev[:part]> [addr] [filename] [bytes]
 - load binary file 'filename' from 'dev' on 'interface'
 to address 'addr' from ext2 filesystem
 UcanLinux >
```

U-Boot seviyesine kadar gelmek, bizce çok önemlidir. Çünkü bu seviyeye kadar yapılan işler daha çok borda bağımlıdır. Hata durumları genelde çok zor tespit edilir. Cihaz tıkanıp kalır. U-Boot seviyesinden sonra çekirdek ve kök dosya sistemi bulunur. Her iki sistem de artık borda çok bağımlı değildir ve u-boot gibi çok etkileşimli bir ortamda çalışılmaktadır. Hatalar çok kolay tespit edilebilir.

Bordda bulunan MMC kartı, “mmc 0″ ile gösterilir. Eğer ikinci bir MMC kartı olsaydı, bu da “mmc 1″ ile gösterilecekti. Kartın üzerindeki her bir bölüm de “:1″, “:2″ vs şeklinde gösterilir. Buna göre MMC kartın 1. bölümü “mmc 0:1″ ile gösterilir.

Açılışta ekrana düşen mesajlardan hemen kopya çekebiliriz. U-Boot sistemi “mmc 0:1″ den uEnv.txt dosyasını okumaya çalışmaktadır. O halde bizim hemen VFAT bölümüne uygun bir env dosyası koymamız gerekecektir. 11. bölümde uEnv.txt dosyası kurulacaktır.

Yukarıda verilen açılış mesajları 2. bölümde incelemiş olduğumuz bootcmd betiğinden gelmektedir. Şimdi en büyük eksiğimiz çekirdektir. Eğer çekirdeği derleyip, uEnv.txt dosyası ile beraber MMC’nin 1. bölümüne atarsak, kök dosya sistemine kadar açılışı gerçekleştirmiş oluruz. MMC’nin 2. bölümüne de kök dosya sistemini kurarsak, takla atmasa da bize login veren basit bir gömülü sistemimiz olacaktır. Şimdi bu adımları tek tek uygulayalım.

8. Çekirdeğin Derlenmesi

Beaglebone için gerekli firmware ve yamalar, esas çekirdek ağacına henüz eklenmemiştir. Bundan dolayı Beaglebone için özel olarak hazırlanan çekirdek aşağıdaki gibi indirilir.

```
# git clone git://arago-project.org/git/projects/linux-am33x.g
```

```
# ln -s linux-am33x linux
# cd linux
# git checkout -f v3.2-staging
# wget "http://arago-project.org/git/projects/?p=am33x-cm3.git
```

Son wget komutu, linux/firmware altına am335x-pm-firmware.bin dosyasını kurar. Çekirdek linux-am33x/ dizini altına yüklenir. Bu dizin linux ile sembolik olarak bağlanır. Böylece komut girişi daha kolay olmaktadır. Çekirdek aşağıdaki gibi, çapraz olarak derlenir.

-jN ile sistemdeki CPU sayısı verilir. Sisteminizde 4 CPU varsa -j4 girilebilir. Derleme hızı çok artacaktır.

```
# cd linux

# make ARCH=arm CROSS_COMPILE=arm-angstrom-linux-gnueabi- am33

# make ARCH=arm CROSS_COMPILE=arm-angstrom-linux-gnueabi- menu

# make -j2 ARCH=arm CROSS_COMPILE=arm-angstrom-linux-gnueabi-

...
 LD      arch/arm/boot/compressed/vmlinux
 OBJCOPY arch/arm/boot/zImage
 Kernel: arch/arm/boot/zImage is ready
 UIMAGE  arch/arm/boot/uImage

Image Name:   Linux-3.2.0-ucan-linux-00078-g63
 Created:      Mon Nov 12 20:16:06 2012
 Image Type:   ARM Linux Kernel Image (uncompressed)
 Data Size:    2421808 Bytes = 2365.05 kB = 2.31 MB
 Load Address: 80008000
 Entry Point:  80008000

Image arch/arm/boot/uImage is ready
```

Çekirdek için nelerin seçilmesi gerektiğinden burada bahsedilmeyecektir. Şu an için çok da önemli değildir. Yazı dizisinin ilerleyen bölümlerde ne tür özelliklerin çekirdeğe ekleneceğinden bahsedilecektir. Fikir vermesi açısından çekirdek için gerekli .config dosyası çalışma paketi içinde kernel.config adı ile verilmiştir. Derleme işine girmeden evvel bu dosya linux/ altında .config ismi ile kopyalanabilir.

Derleme bittikten sonra linux/ altında bulunan .config dosyası, aşağıdaki gibi, mutlaka, çekirdek kodunun dışında, farklı bir yerde, saklanmalıdır.

```
# cp linux/.config  /uygun/bir/yer/kernel.config
```

Yeni çekirdeğimizin boyu aşağıdaki gibi listelenebilir. Ayrıca u-boot başlık bilgileri mkimage ile elde edilebilir.

```
# ls -l linux/arch/arm/boot/uImage
-rw-r--r-- 1 root root 2421872 Nov 12 20:16 linux/arch/arm/boo

# mkimage -l linux/arch/arm/boot/uImage

Image Name:   Linux-3.2.0-ucan-linux-00078-g63
Created:      Mon Nov 12 20:16:06 2012
Image Type:   ARM Linux Kernel Image (uncompressed)
Data Size:    2421808 Bytes = 2365.05 kB = 2.31 MB
Load Address: 80008000
Entry Point:  80008000
```

Çekirdeği hemen test edebiliriz. Bunun için çekirdeği MMC kartının 1. bölümüne aşağıdaki gibi kopyalamamız yeterli olacaktır.

```
# mount /dev/sdb1 /mnt/mmc

# cp linux/arch/arm/boot/uImage /mnt/mmc

# ls -l /mnt/mmc
 total 2662
 -rwxr-xr-x 1 root root   71882 Nov  9 19:52 MLO*
 -rwxr-xr-x 1 root root  230500 Nov  9 19:52 u-boot.img*
 -rwxr-xr-x 1 root root 2421872 Nov 15 09:37 uImage*

# umount /mnt/mmc
```

MMC kart borda takılır ve bord reset edilir.

```
U-Boot 2012.10 (Nov 08 2012 - 21:58:26)

I2C:   ready
 DRAM:  256 MiB
 WARNING: Caches not enabled
 MMC:   OMAP SD/MMC: 0, OMAP SD/MMC: 1
 Using default environment

Net:   cpsw
```

```
 Hit any key to stop autoboot:  0
 SD/MMC found on device 0
 reading uEnv.txt

** Unable to read "uEnv.txt" from mmc 0:1 **
 Loading file "/boot/uImage" from mmc device 0:2
 ** File not found /boot/uImage
 ext2load - load binary file from a Ext2 filesystem

Usage:
 ext2load <interface> <dev[:part]> [addr] [filename] [bytes]
 - load binary file 'filename' from 'dev' on 'interface'
 to address 'addr' from ext2 filesystem
```

Buraya kadar, yani u-boot seviyesindeki açılışa kadar zaten gelmiştik. Şimdi aşağıdaki gibi adım adım, çekirdek belleğe yüklenecek ve işletilecektir.

```
UcanLinux > mmc rescan
```

mmc rescan ile MMC kartın olup olmadığı kontrol edilebilir. Eski u-boot sürümlerinde, MMC kartı ile işlem yapabilmek için en az 1 kez “mmc init” veya “mmc rescan” yazmak gerekliydi. Artık buna gerek yoktur.

```
UcanLinux > fatinfo mmc 0:1

Interface:  MMC
 Device 0: Vendor: Man 02544d Snr 10d2f0d0 Rev: 0.6 Prod: SA04
 Type: Removable Hard Disk
 Capacity: 3768.0 MB = 3.6 GB (7716864 x 512)
 Partition 1: Filesystem: FAT32 "UcanLinux  "
```

fatinfo komutu ile MMC kartın 1. bölümünde bulunan fat dosya sistemi konusunda, yukarıdaki gibi, bilgi alınabilir.

```
UcanLinux > fatls mmc 0:1
   71882   mlo
  230500   u-boot.img
 2421872   uimage

3 file(s), 0 dir(s)
```

fatls komutu ile vfat içeriği listelenebilir.

```
UcanLinux > fatload mmc 0:1 82000000 uImage

reading uImage
2421872 bytes read
```

fatload komut, mmc 0:1′den yani MMC kartın 1. bölümünden, uImage dosyasını, 82,000,000 RAM adresine yükler. Bu adresin bir önemi yoktur. Geçerli bir adres ve sonrasında çekirdeği alacak kadar yerin olması yeterlidir. Bu adresin analizi 12. bölümde yapılacaktır. Bütün adresler 16′lık sistemdedir. Ayrıca önlerine 0x yazılmamıştır.

```
UcanLinux > bootm 82000000
```

bootm (boot from memory) komutu ile 82,000,000 adresinde bulunan çekirdek yüklenir ve işletilir. U-Boot burada, arka arkaya pek çok işlem uygular ve çekirdeği başlatır. U-Boot'un açılış mantığını anlama açısından bu adımlara kısaca değinecek olursak…

- bootm 82000000 girildiğinde, u-boot sistemi, verilen adreste bir imaj olup olmadığına bakar. Diğer bir deyişle kendisine ait 64 baytlık bir başlık arar. Bulamazsa hata verir ve durur. Okuyucu, keyfi bir adres vererek deneme yapabilir. Boot işlemi hemen duracaktır. 64 baytlık başlığın geçerli olup olmadığını anlamak için başlığa ait checksum değerine bakılır. Doğru ise yürütme devam eder.
- U-Boot sistemi, mimarinin uygun olup olmadığını kontrol eder. ppc için derlenmiş bir imaj arm makinede çalıştırılamaz.
- İmaj tipinden hareketle boot edilip edilemeyeceğine bakar. Örneğin bu imajın içinde bir sh betiği varsa, hemen durur, boot etmeye teşebbüs etmez.
- 64 baytlık başlık bilgisi içinde ilgili dosyanın, yani zImage dosyasının da checksum değeri bulunur. Hemen bu checksum değeri de kontrol edilir. Diğer bir deyişle, u-boot başlığı içinde, hem başlığın kendisi hem de veri için checksum değeri mevcuttur.
- Bütün bu adımlardan geçtikten sonra, -a ile verilen, "load address" değeri alınır. Yukarıdaki "mkimage -l" çıkışları incelenirse, "-a" ile verilen değerin 80008000 olduğu görülür. U-Boot sistemi, zImage dosyasını, yani header hariç, uImage dosyasını, -a ile verilen, 80008000 adresine kopyalar. Şekil 2.1′de bu durum resmedilmiştir.
- Kopyalama bittikten sonra, yürütme "-e" ile verilen (entry point) adrese yönlendirilir ve u-boot kendi işini bitirir. Çekirdek her zaman yükleme adresinden başlatılır. Bundan dolayı -a ve -e adresleri her zaman birbirinin aynı olmak zorundadır.

bootm girişinden sonra ekrana düşen açılış mesajlarının bir kısmı aşağıda verilmiştir. Önemli bazı satırlar kırmızı ile işaretlenmiştir. Ayrıca aralara not da düşülmüştür.

```
## Booting kernel from Legacy Image at 82000000 ... <== fatloa
 Image Name:   Linux-3.2.0-ucan-linux-00078-g63
 Image Type:   ARM Linux Kernel Image (uncompressed)
 Data Size:    2421808 Bytes = 2.3 MiB
 Load Address: 80008000
 Entry Point:  80008000
 Verifying Checksum ... OK            <== U-Boot imajı hatasız
 Loading Kernel Image ... OK    <== zImage, 80008000'a yüklend
 OK

Starting kernel ...  <== Yürütme 80008000 adresine geçirildi.

Uncompressing Linux... done, <== zImage, Image haline getirili
booting the kernel.  <== Çekirdek açılmaya başlar.

 Linux version 3.2.0-ucan-linux-00078-g63c1ae3 (root@nkoc_ev)
 CPU: ARMv7 Processor [413fc082] revision 2 (ARMv7), cr=10c53c
 CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instru
 Machine: am335xevm
 Memory policy: ECC disabled, Data cache writeback
 AM335X ES1.0 (sgx neon )
 Built 1 zonelists in Zone order, mobility grouping on.  Total

 # U-Boot'da bootargs ile verilen bütün ifade, burada "Kernel
 # içinde bulunur.

 Kernel command line: root=/dev/mmcblk0p2 rootwait console=tty

 PID hash table entries: 1024 (order: 0, 4096 bytes)
 Dentry cache hash table entries: 32768 (order: 5, 131072 byte
 Inode-cache hash table entries: 16384 (order: 4, 65536 bytes)
 Memory: 256MB = 256MB total
 Memory: 254720k/254720k available, 7424k reserved, 0K highmem
 Virtual kernel memory layout:
 vector  : 0xffff0000 - 0xffff1000   (   4 kB)
 fixmap  : 0xfff00000 - 0xfffe0000   ( 896 kB)
 vmalloc : 0xd0800000 - 0xff000000   ( 744 MB)
 lowmem  : 0xc0000000 - 0xd0000000   ( 256 MB)
 ...
 at24 1-0051: 32768 byte 24c256 EEPROM, writable, 64 bytes/wri
 No daughter card found
 at24 1-0050: 32768 byte 24c256 EEPROM, writable, 64 bytes/wri
 Board name: A335BONE
 Board version: 00A6
```

```
The board is a AM335x Beaglebone.
tps65217 1-0024: TPS65217 ID 0xf version 1.1
print_constraints: DCDC1: 900 <--> 1800 mV at 1800 mV
print_constraints: DCDC2: 900 <--> 3300 mV at 1100 mV
...
mux: Failed to setup hwmod io irq -22
Power Management for AM33XX family
Trying to load am335x-pm-firmware.bin (60 secs timeout)
Copied the M3 firmware to UMEM
clock: disabling unused clocks to save power
Detected MACID=d4:94:a1:8e:65:5a
cpsw: Detected MACID = d4:94:a1:8e:65:5b
omap_rtc omap_rtc: setting system clock to 2000-01-01 00:05:4
Waiting for root device /dev/mmcblk0p2...  <== MMC'nin çekird
mmc0: host does not support reading read-only switch. assumin
mmc0: new high speed SDHC card at address 1234
mmcblk0: mmc0:1234 SA04G 3.67 GiB <== 4GB'lik MMC'yi tanıdı.
mmcblk0: p1 p2 <== MMC'deki 2 bölümü de gördü.

EXT2-fs (mmcblk0p2): warning: mounting unchecked fs, running
VFS: Mounted root (ext2 filesystem) on device 179:2.
EXT2-fs (mmcblk0p2): error: ext2_lookup: deleted inode refere
devtmpfs: error mounting -5 <== Kök dosya sistemi ve /dev olm
Freeing init memory: 220K

Kernel panic - not syncing: No init found.  <== Kök dosya sis

Try passing init= option to kernel. See Linux Documentation/i
Backtrace:
[<c00175c0>] (dump_backtrace+0x0/0x10c) from [<c0354cac>] (du
r7:00000013 r6:c003c118 r5:c0494598 r4:c04e06d8
[<c0354c94>] (dump_stack+0x0/0x1c) from [<c0354d04>] (panic+0
[<c0354cb0>] (panic+0x0/0x190) from [<c000887c>] (init_post+0
r3:0000000d r2:0000000a r1:cf8002c0 r0:c03f308d
[<c0008768>] (init_post+0x0/0x144) from [<c045e2f8>] (kernel_
r4:c04df714
[<c045e208>] (kernel_init+0x0/0x120) from [<c003c118>] (do_ex
```

Yukarıdaki çıkıştan da görüleceği gibi kök dosya sistemi olmadığı için çekirdek paniklemiş ve bord tıkanmıştır. Son olarak kök dosya sistemini MMC'nin 2. bölümüne kuracak ve açılışı tamamlayacağız.

Gömülü sistem kurarken, her zaman bu şekilde adım adım test edilmesi tavsiye edilir. U-Boot seviyesine gelindikten sonraki işler nispeten daha kolaydır.

10. Kök Dosya Sisteminin Kurulması

Kök dosya sistemimiz busybox destekli olacaktır. Bir önceki yazımızda busybox sisteminden bolca bahsettik. Burada sadece ara adımlar verilecektir.

En son sürüm busybox.net adresinden indirilir. Bu yazı yazılırken son kararlı sürüm 1.20.2 idi. Aşağıdaki gibi çapraz derleme yapılır.

```
$ cd busybox

$ make menuconfig

Busybox Settings  --->
 Build Options  --->
 [ ] Build BusyBox as a static binary (no shared libs)
 [ ]   Build BusyBox as a position independent executable
 [ ] Force NOMMU build
 [ ] Build shared libbusybox
 [ ] Build with Large File Support (for accessing files > 2 GB
 (arm-angstrom-linux-gnueabi-) Cross Compiler prefix
 ()  Path to sysroot
 ()  Additional CFLAGS
 ()  Additional LDFLAGS
 ()  Additional LDLIBS

Busybox Settings  --->
 Installation Options ("make install" behavior)  --->
 What kind of applet links to install (as soft-links)  --->
 (/nk/blog/2/RootFS) BusyBox installation prefix

 # make
 # make install

 # cp .config /uygun/bir/yer/busybox.config
```

Burada busybox dinamik olarak derlenmektedir. make install ile /nk/blog/2/RootFS dizini altına kurulum yapılır. Okuyucu kendisine uygun herhangi bir dizin adı seçebilir. ASLA /usr/local gibi genel isimler seçilmemelidir. Mevcut host sistemi kaybedilir.

install işleminden sonra RootFS/ altında /bin, /sbin ve /usr dizinleri kurulur. Ayrıca linuxrc sembolik linki kurulur. Bu linke gerek yoktur, silinebilir.

Bizler busybox'ı derlerken hemen hemen hiç bir varsayılan seçime dokunmadık. Gerçek uygulamada, müşteriye verilen sistemde sadece ve sadece kullanılan komutlar busybox'a dahil edilmelidir. Kullanılan komutları tespit etmenin en basit yolu /etc/rcS içine bakmaktır. Nihai sistemde /etc/rcS içinde bulunan komutlar busybox apleti olarak seçilmelidir.

Derlediğimiz busybox'ın özellikleri aşağıdaki gibi incelenebilir.

```
# ls -l busybox

 -rwxr-xr-x 1 root root 769296 Nov 15 10:02 busybox*

# file busybox

 busybox: ELF 32-bit LSB executable,
          ARM,
          version 1 (SYSV),
          dynamically linked (uses shared libs),
          stripped
```

Şimdi kök dosya sistemi MMC'nin 2. bölümüne kurulabilir. Bizim kök dosya sistemimiz sadece busybox destekli komutlardan oluşmaktadır. Bundan dolayı çok ufaktır. Öncelikle MMC'nin 2. bölümü aşağıdaki gibi silinip, 16MB'lık yeni ve daha ufak bir bölüm yapılabilir.

```
# fdisk /dev/sdb

Command (m for help): p

Disk /dev/sdb: 3951 MB, 3951034368 bytes
 122 heads, 62 sectors/track, 1020 cylinders, total 7716864 se
 Units = sectors of 1 * 512 = 512 bytes
 Sector size (logical/physical): 512 bytes / 512 bytes
 I/O size (minimum/optimal): 512 bytes / 512 bytes
 Disk identifier: 0xee385770

Device Boot      Start         End      Blocks   Id  System
 /dev/sdb1   *        2048      133119       65536    c  W95 F
 /dev/sdb2          133120     7716863     3791872   83  Linux

Command (m for help): d
 Partition number (1-4): 2  <== 2. bölümü sil.

Command (m for help): n
 Command action
 e   extended
 p   primary partition (1-4)
 p
 Partition number (1-4, default 2):
 Using default value 2
 First sector (133120-7716863, default 133120):
```

```
 Using default value 133120
 Last sector, +sectors or +size{K,M,G} (133120-7716863, defaul

Command (m for help): w
```

16 MB'lık 2. bölüm kurulmuştur. Şimdi bu bölüme aşağıdaki gibi ext2 dosya sistemi kurup mount edebiliriz. mkfs.ext3 veya mkfs.ext4 ile jurnal destekli dosya sistemi de kurulabilir.

```
# mkfs.ext2 /dev/sdb2

# mount /dev/sdb2 /mnt/mmc

# df /mnt/mmc
Filesystem     1K-blocks  Used Available Use% Mounted on
/dev/sdb2          15863   140     14904   1% /mnt/mmc
```

Bizler MMC'nin 2. bölümünü /mnt/mmc dizinine bağladık. Okuyucu istediği dizini kullanabilir. Kök dosya sistemi sadece /bin, /sbin ve /usr dizinlerinden oluşmaz. Diğer dizinler, kütüphaneler ve /etc dizini RootFS altında daha önce kurulmuştur. Çalışma paketi içinde RootFS'in tam hali bulunabilir. Bir önceki bölümde RootFS üzerinde bolca laf edildiği için bu bölümde sükut geçilecektir. Örnek kök dosya sistemi aşağıdaki gibi kurulabilir.

```
 # cd RootFS/

 # rm -f linuxrc
 # cp -a * /mnt/mmc

 # ls -l /mnt/mmc
 total 26
 drwxr-xr-x 2 root root  2048 Nov 15 10:02 bin
 drwxr-xr-x 2 root root  1024 Aug 24 13:36 dev
 drwxr-xr-x 2 root root  1024 Nov 15 15:47 etc
 drwxr-xr-x 2 root root  1024 Aug 24 13:36 home
 drwxr-xr-x 3 root root  1024 Aug 24 13:45 lib
 drwx------ 2 root root 12288 Nov 15 16:20 lost+found
 drwxr-xr-x 2 root root  1024 Aug 24 13:36 mnt
 drwxr-xr-x 2 root root  1024 Aug 24 13:36 proc
 drwxr-xr-x 2 root root  1024 Aug 24 13:36 root
 drwxr-xr-x 2 root root  1024 Nov 15 10:02 sbin
 drwxr-xr-x 2 root root  1024 Aug 25 20:12 sys
 drwxr-xr-x 2 root root  1024 Aug 24 13:36 tmp
 drwxr-xr-x 4 root root  1024 Aug 25 14:12 usr
```

```
 drwxr-xr-x 2 root root  1024 Aug 24 13:36 var

 # df /mnt/mmc
 Filesystem     1K-blocks  Used Available Use% Mounted on
 /dev/sdb2          15863  3080     11964  21% /mnt/mmc

# umount /mnt/mmc
```

df çıktısından da görüleceği gibi kök dosya sistemimiz sadece 3MB yer kaplamaktadır. Okuyucu projenin amacına göre kök dosya sistemini genişletebilir. Buradaki sistem minimal bir sistemdir ve inanılmaz derecede çok işe yaramaktadır.

Kök dosya sistemimiz de tamamlanmıştır. Şimdi tek eksiğimiz, el ile yaptığımız açılışın otomatik olarak yapılabilmesidir. İşte tam bu anda MMC'nin 1. bölümünde olması gereken uEnv.txt dosyası imdada yetişir.

11. uEnv.txt Dosyasının Kuruluşu

Daha öncede belirttiğimiz gibi, uEnv.txt içine çalışmasını istediğimiz komutları yazabiliriz. uenvcmd için yazacağımız bütün ifadler u-boot tarafından hemen çalıştırılacaktır. Aşağıdaki satır uEnv.txt içine bir editör ile veya echo ile yazılabilir. 82000000 adresi yerine 80007FC0 adresi de yazılabilir.

```
uenvcmd=fatload mmc 0:1 82000000 uImage && bootm 82000000
```

Bu ifade daha önce el yordamı ile yaptığımız girişin aynısıdır. uEnv.txt dosyası aşağıdaki gibi MMC'nin 1. bölümüne yazılabilir.

```
# mount /dev/sdb1 /mnt/mmc

# cp uEnv.txt /mnt/mmc

# ls -l /mnt/mmc
 total 2662
 -rwxr-xr-x 1 root root   71882 Nov  9 19:52 MLO
 -rwxr-xr-x 1 root root  230500 Nov  9 19:52 u-boot.img
 -rwxr-xr-x 1 root root      59 Nov 15 16:24 uEnv.txt
 -rwxr-xr-x 1 root root 2421872 Nov 15 09:37 uImage

# umount /mnt/mmc
```

Yukarıdaki ls -l çıkışından da görüleceği gibi MMC'nin 1. bölümünde 4 adet dosyamız mevcuttur. Bu dosyalar çekirdek seviyesine kadar açılış yapılır. uImage dosyası burada bulunmak zorunda değildir. MMC'nin 2. bölümünde veya ağ üzerindeki bir

makine de olabilir. Yazı dizimizin devamında, uImage dosyası farklı ortamlarda bulunacaktır.

Test için MMC kartı borda takılır ve bord reset edilir. root/root ile giriş yapılabilir. IP değeri 10.0.1.3 ile verilmiştir. Bu değer /etc/rcS içinde değiştirilebilir.

telnet ile başka bir makineden aşağıdaki gibi giriş yapılabilir.

```
$ telnet 10.0.1.3
 Trying 10.0.1.3...
 Connected to 10.0.1.3.
 Escape character is '^]'.

UcanLinux login: root
Password: root

Linux UcanLinux 3.2.0-ucan-linux-00078-g63c1ae3 #1 Mon Nov 12

root@UcanLinux:~ # exit
```

Bordun açılışı aşağıda özetlenmiştir.

```
U-Boot 2012.10 (Nov 08 2012 - 21:58:26)

I2C:   ready
 DRAM:  256 MiB
 WARNING: Caches not enabled
 MMC:   OMAP SD/MMC: 0, OMAP SD/MMC: 1
 Using default environment

Net:   cpsw
 Hit any key to stop autoboot:  0
 SD/MMC found on device 0
 reading uEnv.txt

59 bytes read
 Loaded environment from uEnv.txt
 Importing environment from mmc ...
 Running uenvcmd ...

reading uImage

2421872 bytes read

## Booting kernel from Legacy Image at 82000000 ...
```

```
 Image Name:   Linux-3.2.0-ucan-linux-00078-g63
 Image Type:   ARM Linux Kernel Image (uncompressed)
 Data Size:    2421808 Bytes = 2.3 MiB
 Load Address: 80008000
 Entry Point:  80008000
 Verifying Checksum ... OK
 Loading Kernel Image ... OK
 OK

Starting kernel ...

Uncompressing Linux... done, booting the kernel.
 Linux version 3.2.0-ucan-linux-00078-g63c1ae3 (root@nkoc_ev)
 CPU: ARMv7 Processor [413fc082] revision 2 (ARMv7), cr=10c53c
 ...
 omap_rtc omap_rtc: setting system clock to 2000-01-01 00:30:4
 Waiting for root device /dev/mmcblk0p2...
 mmc0: host does not support reading read-only switch. assumin
 mmc0: new high speed SDHC card at address 1234
 mmcblk0: mmc0:1234 SA04G 3.67 GiB
 mmcblk0: p1 p2
 VFS: Mounted root (ext2 filesystem) on device 179:2.
 devtmpfs: mounted
 Freeing init memory: 220K
 net eth0: CPSW phy found : id is : 0x7c0f1

 Filesystem           1K-blocks      Used Available Use% Mount
 /dev/root                15863      3080     11964  20% /
 devtmpfs                127360         0    127360   0% /dev
 tmpfs                   127468         0    127468   0% /dev/

http://ucanlinux.com

UcanLinux login: root
Password: root

 Linux UcanLinux 3.2.0-ucan-linux-00078-g63c1ae3 #1 Mon Nov 12

root@UcanLinux:~ # df
 Filesystem           1K-blocks      Used Available Use% Mount
 /dev/root                15863      3081     11963  20% /
 devtmpfs                127360         0    127360   0% /dev
 tmpfs                   127468         0    127468   0% /dev/

root@UcanLinux:~ # free
 total         used         free       shared      buffers
 Mem:        254940         5576       249364            0
 -/+ buffers:               5488       249452
```

```
Swap:            0             0             0
root@UcanLinux:~ #
```

Yukarıdaki df çıktısından görüleceği gibi kök dosya sistemi sadece 3MB yer tutmuştur. free çıkışından da görüleceği gibi, harcanan toplam bellek 6MB'tan azdır. Tabii ki bu sistem takla atamaz ama olsun, login vermektedir ki bu da çok güzel bir başlangıçtır.

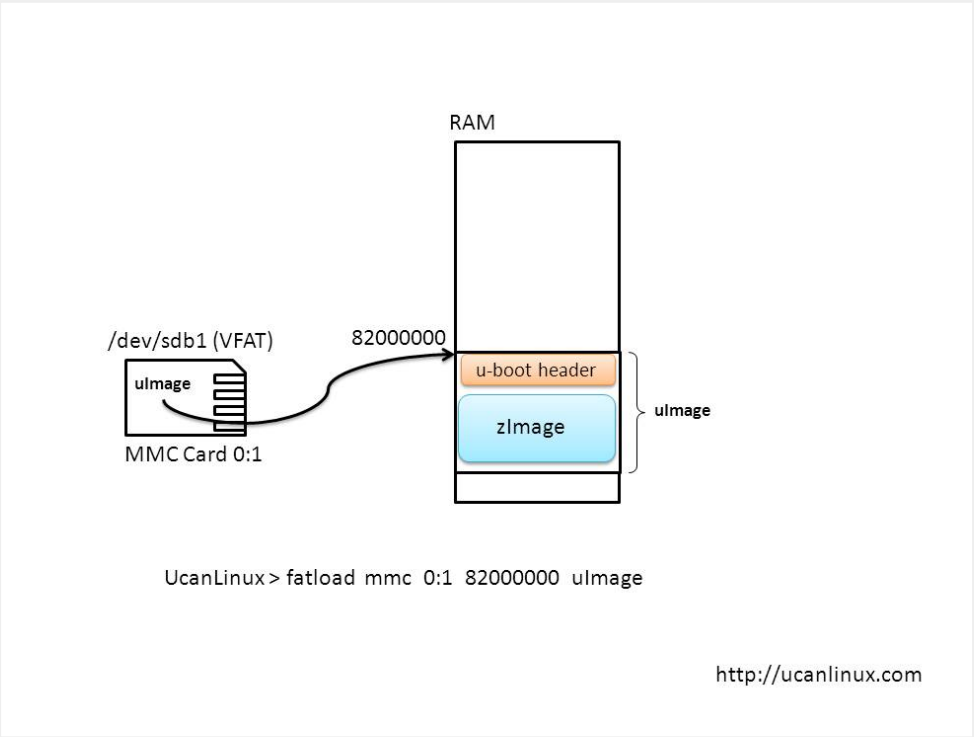
Çekirdek yüklendikten sonra, /dev sistemini otomatik olarak bağlar. Çünkü çekirdek derlemesi sırasında, kök dosya sistemini bağladıktan hemen sonra /dev'i bağlaması için gerekli seçimler yapılmıştır.

Kök dosya sistemi bağlandıktan sonra yürütme /sbin/init programına geçer. /sbin/init programı /etc/inittab'da verilen satırlara göre sistemi ayağa kaldırır. Sistemde tek açılış betiği vardır, o da /etc/rcS'dir. Okuyucu bu betiği kendi amaçlarına göre geliştirebilir.

Angstrom sisteminde inanılmaz karışık bir açılış sistemi mevcuttur. Ayrıca açılış için run level kullanılmıştır. Gömülü bir sistemde bunlara hiç gerek yoktur.Ayrıca kök dosya sistemi read/write bağlanmıştır. Bu da çok sakıncalıdır. İlerleyen yazılarımızda bu konuya genişçe değinilecektir.

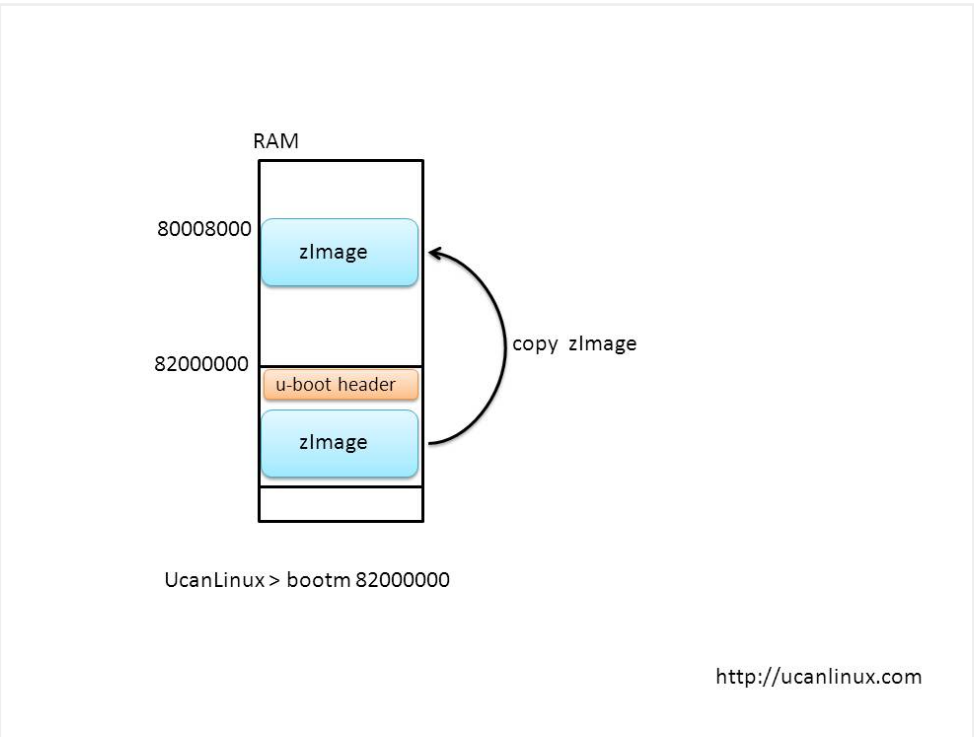
12. XIP kavramı

Daha önce, fatload ve bootm komutları ile çekirdeği yüklemiştik. Bu iki komutun u-boot tarafından nasıl işlendiği Şekil 2.1 ve 2.2′de resmedilmiştir.



— Şekil 2.1: fatload komutu ile imajın belleğe yüklenmesi.

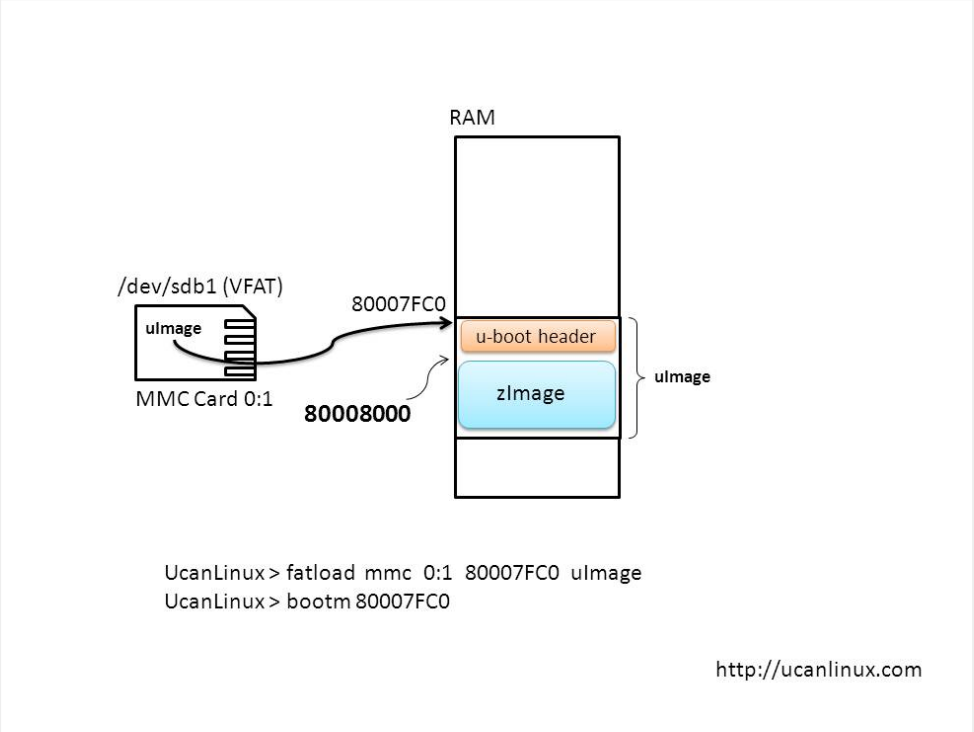
fatload komutu, MMC'nin 1. bölümünden, yani "mmc 0:1″ den, uImage dosyasını okur ve RAM belleğin 82,000,000 adresine yükler. uImage dosyası esas itbari ile 64 baytlık bir u-boot başlığından ve zImage dosyasından oluşur. Şekil 2.2'den bootm komutundan sonra gerçekleşen işler resmedilmiştir.



— Şekil 2.2: bootm komutunun işlemesi.

Şekil 2.2'de gösterildiği gibi, bootm komutu u-boot header içinde bulunan ve -a ile verilen adresi elde eder. Bu adres örneğimizde 80008000'dir. Hemen zImage dosyasını bu adrese yani 80008000 adresine kopyalar. Sonra da yürütmeyi -e ile verilen adrese geçirir.

Buradaki kopyalama işi son derece gereksiz gibi gözükmektedir. fatload ile uygun bir adres verildiği takdirde zImage dosyası hiç bir yere kopyalanmayabilir. Kopyalamaya gerek kalmaz. Bu çok özel durum Şekil 2.3'de resmedilmiştir.



— Şekil 2.3: Kopyalama yapmadan yürütme.

uImage dosyası, fatload komutu ile 80007FC0 adresine yüklenir. Header boyu tam 64 bayttır. 80007FC0 adresinden itibaren 64 bayt sonrasının adresi 80008000'dir. zImage için verilen başlangıç adresi de zaten 80008000'dir. U-Boot bunu anlar ve zImage dosyasını hiç bir yere kopyalamaz ve olduğu yerde yürütür (execute in place, xip).

Yürütmeden önce kopyalanmayan ve olduğu yerde çalıştırılan imajlara, XIP imajı denir. Eğer bir imaj XIP olarak adreslenmişse, u-boot açılış mesajlarında bu belirtilir. 80008000 yerine 80007FC0 adresi ile yapılan test aşağıda verilmiştir. Açılış u-boot seviyesinde durdurulur ve el yordamı ile test yapılır. Okuyucu aynı değişikliği uEnv.txt dosyasında da yapabilir.

```
U-Boot SPL 2012.10 (Nov 08 2012 - 21:58:26)
 ...
 Hit any key to stop autoboot:  0 <== açılışı bir tuşa basarak

UcanLinux > fatload mmc 0:1 80007FC0 uImage
 reading uImage
2421872 bytes read

UcanLinux > bootm 80007FC0
 ## Booting kernel from Legacy Image at 80007fc0 ...
 Image Name:   Linux-3.2.0-ucan-linux-00078-g63
 Image Type:   ARM Linux Kernel Image (uncompressed)
 Data Size:    2421808 Bytes = 2.3 MiB
 Load Address: 80008000
 Entry Point:  80008000
 Verifying Checksum ... OK
 XIP Kernel Image ... OK
 OK

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

13. Kök Dosya Sistemini rw Bağlanması Hakkında

Bu çalışmada kök dosya sistemi MMC üzerine kurulmuş ve read/write olarak mount edilmiştir. Ciddi bir sistemde MMC üzerine kurulu kök dosya sistemi read/write mount edilmemelidir. Ani kapanmalarda, büyük ihtimalle dosya sistemi kaybedilecektir.

Burada verilen örnek sistemimizde kök dosya sistemi ext2 olarak seçilmiştir. ext2 sistemi ani kapanmalara karşı çok hassastır. Genelde ext3 veya ext4 gibi jurnal tabanlı dosya sistemleri seçilir. Okuyucunun yapacağı tek iş, yukarıdaki kuruluşlarda, mkfs.ext2 yerine mkfs.ext3 yazmaktır. Yine de jurnal tabanlı kök dosya sistemleri de ani kapanmalara karşı %100 koruma sağlamaz.

Ani kapanmalara karşı bir başka yol, kök dosya sisteminin önce ro sonra rw bağlanmasıdır. Bunun için çekirdek parametrelerine ro eklenmelidir. Böylece kök dosya sistemi çekirdek tarafından, başlagıçta ro olarak bağlanacaktır. Daha sonra /etc/rcS içinde kök dosya sistemi için fsck başlatılmalı ve hata varsa dosya sistemi kurtarılmaya çalışılmalıdır. Hata yoksa kök dosya sistemi read/write olarak mount edilmeli ve açılış bilinen yolla devam ettirilmelidir. Host sistemler için yapılan bu teknik gömülü sistemlere de uygulanabilir. Çok basit bu yöntem okuyucuya alıştırma olarak bırakılmıştır.

jffs2 ve özellikle UBIFS gibi flash destekli dosya sistemleri kapanmaya karşı çok

dayanıklıdır. Bu dosya sistemleri doğrudan read/write mount edilebilir. Ama yine de bu tavsiye edilmez. Örneğin UBIFS sisteminin fsck programı mevcut değildir. Sistem kendi kendini onarmaya çalışır. Eğer onaramazsa ne olacaktır? Bu bile başlı başına proje için stratejik bir karardır. Beaglebone sisteminde flash disk mevcut değildir. Bundan dolayı üretilecek çözümler MMC destekli olmalıdır.

Elimizde MMC varsa kök dosya sistemi nasıl bağlanmalıdır?

Öncelikle kök dosya sistemi her zaman read/only bağlanmalıdır. Asla read/write bağlanmamalıdır. read/write olması gereken dizinler için bazı çözümler mevcuttur.

Öncelikle /dev dizini her zaman read/write olmalıdır. Bu işi çekirdek zaten halleder. Burada bir sorun yoktur.

/var/pid gibi bilgilerin geçici olarak saklandığı dosya veya dizinler /dev/shm'ye yönlendirilmelidir.

/tmp, /var gibi dizinler olduğu gibi /dev/shm'ye yönlendirilebilir. Bu yönlendirme işi sembolik linkler veya "mount -o bind" ile yapılabilir.

config bilgileri gibi bazı bilgilerin kalıcı olması gerekir. Bu tür bilgiler başka bir MMC bölümüne yönlendirilir. Örneğin MMC'de 3. bir bölüm açılır ve bu bölüm /etc/rcS tarafından açılışta mount edilebilir. Buraya yazım yapacak programlar "atomik güncelleme" denilen bir yöntemle dosyaları, ani kapanmalardan etkilenmeden veya çok az etkilenerek güncelleyebilirler. En nihayetinde, sistem nasıl kapanırsa kapansın kök dosya sistemi her zaman ayakta kalacaktır. En kötü ihtimalle rw bağlı olan bölüm kaybedilecektir. Kök dosya sistemi her zaman ayakta olacağı için, buradaki bilgiler başka bir biçimde yeniden temin edilebilecektir.

Okuyucu alıştırma olarak aşağıdakileri yapabilir. Şu anki mevcut sisteme uygulaması son derece kolay olacaktır.

- kök dosya sistemi ro bağlanmalıdır.
- /tmp ve /var gibi dizinler /dev/shm'ye sembolik olarak bağlanmalıdır.
- /etc/rcS'de gerekli değişiklikler yapılmalıdır.

14. Gelecek Bölümler

Gelecek yazıların aşağıdakiler gibi olması planlanmaktadır.

2. Bölüm

Kök dosya sistemi çok ufaksa MMC'ye yüklemeye gerek yoktur. Ufak kök dosya sistemleri doğrudan ramfs şeklinde bağlanabilirler. Ani kapanmalarda bozulma

ihtimalleri yoktur. Çünkü her açılışta yeni baştan kurulurlar. Fakat burada da yine kalıcı bilgilerin saklanması sorunu vardır. İkinci bölümün konusunu ramfs destekli kök dosya sistemi olacaktır.

3. Bölüm

Bu bölümde ağ üzerinden boot tekniklerinden bahsedilecektir. bootp, dhcp ve NFS kullanılarak boot tekniği ve kök dosya sistemlerinin bağlanılmasından bahsedilecektir.

4. Bölüm

Gömülü sistemler için ideal test ortamının hazırlanılmasından bahsedilecektir ki bizce bu işle uğraşan herkesin illa ki kullanması gereken bir yöntemdir.

15. Diğer

Bu çalışmada elde edilen bazı dosyalar buradan indirilebilir. Özellikle config dosyaları yazıyı okurken yapılacak denemelerde vakit kazandıracaktır.

Yazara **nazim** at **ucanlinux.com** adresinden ulaşılabilir. Ya da bu yazı doğrudan blog'dan okunuyursa blogun altına bulunan mesaj kutusu kullanılabilir. Bu kutu belirli bir süre açık kalacaktır.

Bu yazının en güncel hali her zaman http://ucanlinux.com adresinin blog sekmesinden elde edilebilir. Mevcut güncellemeler aşağıda verilmiştir.

17.Kasım.2012: ilk yayın

16. Kullanım Hakları

— 1. bölümün sonu —