Gömülü Linux Sistemleri

Login'e Kadar Linux

Beaglebone İle Çalışmak, 2. Bölüm

Giriş

Bu yazı dizisinin 1. bölümünde Beaglebone üzerine basit bir gömülü sistem kurulmuştur. Bu gömülü sistemin kök dosya sistemi MMC kartı üzerinde oturmaktadır. Eğer yeteri kadar ufaksa, kök dosya sistemi MMC gibi fiziksel bir ortam yerine initramfs şeklinde, doğrudan RAM bellekte kurulabilir.

1. RAM Disk Nedir?

Dosya sistemleri MMC veya Flash Disk gibi kalıcı fiziksel cihazlara kurulabilirler. Bu cihazların en büyük özelliği elektrik gittikten sonra verilerin kaybolmamasıdır.

Dosya sistemleri aynı zamanda doğrudan RAM içinde de kurulabilir. Diğer bir deyişle, RAM bellek, disk bölümü (disc partition) gibi kullanılabilir.

Hemen örnek olarak, masaüstü sistemde, aşağıdaki gibi RAM bellekte bir dosya sistemi kurulabilir. Şu an için gömülü sistemle bir ilgimiz yoktur.

```
01 # mkfs.ext2 /dev/ram0
02 # mkdir /mnt/ramdisk
03 # mount /dev/ram0 /mnt/ramdisk

04 # df /mnt/ramdisk
   Filesystem     1K-blocks  Used Available Use% Mounted on
   /dev/ram0          15863   140     14904   1% /mnt/ramdisk

05 # cp falan filan /mnt/ramdisk
```

```
06 # umount /mnt/ramdisk
```

Yukarıdaki adımların üzerinden kısaca geçelim.

RAM belleği disk bölümü gibi kullanabilmek için öncelikle, derleme sırasında, aşağıdaki özellikler çekirdeğe katılmalıdır.

```
General setup  --->
   [*] Initial RAM filesystem and RAM disk (initramfs/initrd)

Device Drivers  --->
    [*] Block devices  --->
         <*>   RAM block device support
         (16)     Default number of RAM disks
         (16384)  Default RAM disk size (kbytes)
```

“Default number of RAM disks” satırı ile, çekirdeğin destekleyeceği cihaz sayısı verilir. Bu değer, örnekte 16 olarak verilmiştir. Bu durumda /dev/ram0, /dev/ram1, … /dev/ram15′e kadar, toplam 16 adet cihaz ismi kullanılabilir.

Her bir cihazın kapasitesi “Default RAM disk size” satırında, 16MB olarak verilmiştir. Bu durumda, örneğin /dev/ram0 cihazı için en fazla 16MB’lık dosya sistemi kurulabilir.

RAM diskin kapasitesi, istenirse açılış sırasında ramdisk_size parametresi ile dinamik olarak da değiştirilebilir. Bakınız: linux/Documentation/blockdev/ramdisk.txt

Çalışan bir sistemde, /proc/config.gz mevcut ise, aşağıdaki gibi RAM disk özellikleri elde edilebilir.

```
$ zcat /proc/config.gz | grep DEV_RAM

CONFIG_BLK_DEV_RAM=y
CONFIG_BLK_DEV_RAM_COUNT=16
CONFIG_BLK_DEV_RAM_SIZE=16384
```

Şimdi, yukarıdaki 6 satırlık örnek girişe dönebiliriz. 01. satırda mkfs.ext2 /dev/ram0 komutu mevcuttur. /dev/ram0 cihazına, RAM’daki bir disk bölümü gözü ile bakılabilir. Bu durumda bu bölüme ext2 dosya sistemi kurulacaktır. Derleme sırasında ramdisk kapasitesi olarak 16MB verildiği için, dosya büyüklüğü de 16MB olacaktır. 04. satırda dosya sisteminin büyüklüğü rapor edilmektedir.

03. satırda, ram0 üzerine kurulan dosya sistemi, /mnt/ramdisk dizinine

bağlanmaktadır.

/mnt/ramdisk dizinine geçildiği an, artık RAM belleke kurulan ext2 dosya sisteminin içindeyiz demektir. Bütün komutlar, standard bir disk bölümü gibi, RAM bellekteki dosya sistemi için de geçerlidir.

ram0 üzerinde ext2 dosya sistemi yaratırken, dosya boyu açıkça verilmemiştir. Bu durumda ram0′ın kapasitesi, dosya boyu olarak kabul edilmiştir. Dosya boyu istenirse açıkça verilebilir. Aşağıdaki örnekte ram1′e 8MB'lık ext2 dosya sistemi kurulmaktadır.

```
$ mkfs.ext2 /dev/ram1 8192
```

Initramfs sistemi ile doğrudan ilgili olmamasına rağmen niçin bu kadar laf ettik? Aslında lafı getireceğimiz yer initramfs'e niçin gerek duyulduğudur.

Şimdi, yukarıdaki 6 satırlık örnek girişte, 05. satıra bakalım. Burada basit bir kopyalama işlemi mevcuttur. Bu kopyalama işlemi nasıl yapılır?

Her ne kadar RAM üzerinde otursa da /dev/ram0 sistemi de aynen, standard bir blok cihaz gibi çalışır. Örneğin yukarıdaki cp işlemi yapılırken veriler önce disk ön-belleğine (disc cache) sonra da dosya sistemine yazılır. Eğer SCSI veya NAND flash disk gibi, gerçek fiziksel bir cihaz kullanıyorsak, bu çok mantıklıdır. Ama zaten bütün işlerin sadece RAM'da döndüğü bir ortamda, bu 2 kez yazma işine gerek var mıdır? Tabii ki yoktur.

Ayrıca dosya sistemi yaratıldıktan sonra, kendisine tahsis edilen bütün belleğe oturması gerekli midir? Örneğin yukarıdaki 6 satırlık örnekte, 16MB'lık ext2 dosya sistemi kurduk. mount işlemini yaptıktan hemen sonra 16MB'lık RAM kapasitesi, anında kullanım dışı kalmıştır, çekirdekten çalınmıştır. Buna da gerek yoktur. İçeri dosya girdikçe RAM bellek kullanılmalı ve dosya silindikçe RAM bellek bırakılmalıdır.

Bu sorunların çözümü için çok basit bir öneri getirilmiştir. Dosya sistemine ait veriler her zaman disk ara-belleğinde otursun. Eğer disk ara belleği bir biçimde, dosya sistemi gibi mount edilebilirse yukarıdaki 2 temel iyileştirme de sağlanmış olacaktır.

Çekirdeğin disk ara-bellek mekanizmasında gerekli güncellemeler yapılmış ve yukarıda bahsettiğimiz iyileştirmeler çekirdeğe eklenmiştir. İşte çekirdeğin, disk ara-belleğini dosya sistemi gibi mount etmesini sağlayan dosya sistemine ramfs denir.

Fakat ramfs sisteminin, dolduğunda hata vermemesi gibi garip özellikleri vardır. ramfs sistemi daha da geliştirilmiş ve tmpfs dosya sistemi ortaya çıkmıştır. tmpfs dosya sistemi, şu anda kullanılan en yeni nesil RAM tabanlı sistemdir.

Açılışta çekirdek tarafından mount edilen tmpfs dosya sistemine initramfs denir. Diğer bir deyişle, initramfs sistemi tmpfs sisteminin bir örneğidir. Initramfs sistemine doğrudan tmpfs olarak bakılabilir.

Peki niçin Inittmpfs denilmemiş de, initramfs denilmiştir? Çünkü henüz tmpfs dosya sistemi geliştirilmemiş, ramfs dosya sistemi varken, açılışta kullanılan dosya sistemine de initramfs denilmiştir. Fakat daha sonra tmpfs geliştirilince, initramfs ismi değiştirilmemiştir.

2. TmpFS Örnekleri

Güncel Linux sürümlerinde, tmpfs dosya sistemi pek çok yerde kullanılmaktadır. Örneğin /dev/shm sistemi aslında tmpfs dosya sistemidir. Açılışta çekirdek tarafından bağlanan /dev sistemi de devtmpfs olarak adlandırılır. Hepsinin temeli tmpfs dosya sistemidir.

tmpfs dosya sistemi el ile nasıl kurulur? Yanıtı çok basittir. Önce yanıta geçmeden önce, yukarıda örnek olarak verdiğimiz 6 satırlık girişe tekrar dönelim. Burada herşeyden önce RAM belleğin disk bölümü gibi kullanılmasını sağlayan ve /dev/ramN ile gösterilen bir cihaza gerek vardır. Ayrıca bu cihaz yardımı ile yapılan bölümlendirme üzerinde, ext2, ext3, minix, vfat vs gibi bir dosya sistemi kurmak gerekir. Özetle, /dev/ramN gibi bir bölümlendirme sürücüsüne ve bir dosya sistemine ihtiyacımız vardır.

tmpfs sisteminde ise herhangi bir dosya sistemine gerek yoktur. Öyle ki çekirdeği derlerken hiç bir dosya sistemine destek vermesek de olur. Çünkü çekirdeğin disk ara-bellek mekanizması dosya sistemi gibi davranır. Ayrıca RAM belleği bölümlendirmek için /dev/ramN gibi cihazlara da gerek yoktur.

Aşağıdaki gibi, tmpfs destekli bir dosya sistemi mount edilebilir!

```
07 # mkdir /mnt/disk

08 # mount -t tmpfs tmpfs /mnt/disk

09 # df /mnt/disk
   Filesystem     1K-blocks  Used Available Use% Mounted on
   tmpfs             998056     0    998056   0% /mnt/disk

10 # free
                 total       used       free     shared    buff
   Mem:       1996116    1158060     838056          0      61
   -/+ buffers/cache:     419036    1577080
   Swap:      1959892          0    1959892
```

```
11 # cp falan filan /mnt/disk

12 # umount /mnt/disk
```

08. satırda gösterildiği gibi tmpfs dosya sistemi mount edilir. Dikkat edilirse burada dosya sistemi kullanılmamıştır. Yani mkfs.ext2 gibi bir komut kullanılmamıştır. Sanki daha önce bu dosya sistemi kurulmuş gibi mount edilmiştir. Çünkü tmpfs dosya sistemi, hiç kullanılmasa bile çekirdeğin ara bellek mekanizmasında her daim hazır bulunmaktadır. Hatta Initramfs sistemini hiç kullanmasak bile, nerede ise hiç yer kaplamadığı için, çekirdek içinde içi tamamen boş bir initramfs sistemi mevcuttur. Diğer bir deyişle Initramfs özelliği çekirdeğin içinde her daim mevcttur. Ne güzel.

Burada hemen şu soru sorulabilir. Dosya sisteminin kapasitesi nedir? tmpfs dosya sistemlerinde kapasite her zaman fiziksel belleğin, yani RAM belleğin tam yarısı kadardır.

Yukarıdaki 10. satırda, free komutu ile RAM belleğin 2GB olduğu hemen görülebilir. “mem:” satırı ve “total” sütununa bakılmalıdır. Bu değerin yarısı 1GB’tır. 09. satırda tmpfs dosya sisteminin kapasitesinin 1GB olduğu görülebilir. Yani fiziksel belleğin tam yarısı kadardır.

Bir Linux dağıtımında bir kaç adet bu şekilde tmpfs dosya sistemi bulunabilir. Her biri 0.5 RAM’lık kapasiteye sahiptir. 2′den fazla tmpfs sisteminin kapasitesi, toplam RAM kapasitesini aşar. Bu bir çelişki değildir.

tmpfs dosya sistemleri içine dosya atıldıkça büyürler. İçindeki dosyalar silindikçe ufalırlar. Ayrıca gerekirse swap edilebilirler.

/dev/ram0 örneğinde verildiği gibi, mevcut disk kapasitesinin tamamı RAM bellekten çalınıp üzerine oturulmaz. Buradaki kapasite aslında sadece üst sınır için bir değerdir. Alt sınır her zaman 0 bayttır. Yani dosya sisteminde dosya/dizin mevcut değilse, kullanılan RAM miktarı sıfırdır. Eğer dosya sisteminin kapasitesi aşılırsa, örneğimizde 1GB olmak üzere, dosya sistemi, “yeterli yer kalmadı” hatası verir.

Dosya sistemi yaratılırken, istenirse aşağıdaki gibi açıkça kapasite verilebilir.

```
13 # mount -o size=16M -t tmpfs tmpfs /mnt/disk

14 # df /mnt/disk
   Filesystem     1K-blocks  Used Available Use% Mounted on
   tmpfs              16384     0     16384   0% /mnt/disk
```

```
15 # mount | grep "/mnt/disk"
   tmpfs on /mnt/disk type tmpfs (rw,size=16M)

16 # mount -o remount -o size=32M /mnt/disk

17 # df /mnt/disk
   Filesystem     1K-blocks  Used Available Use% Mounted on
   tmpfs              32768     0     32768   0% /mnt/disk

18 # umount /mnt/disk
```

Yukarıdaki örnekte, 13. satırda size=16M seçeneğiyle, dosya sisteminin kapasitesi 16MB ile sınırlandırılmıştır. 13. satırdaki “-o ” seçeneği option anlamındadır. Her dosya sistemi için option değerleri birbirlerinden farklıdır. Diğer bir deyişle buradaki size=… seçeneği sadece tmpfs dosya sistemi için geçerlidir. Örneğin bu seçenek vfat veya başka bir dosya sisteminde kullanılamaz. Her dosya sistemi için geçerli olan seçenekler “$ man mount” komutu ile incelenebilir.

13. satırda, -t ile dosya tipi verilmiştir. “-t tmpfs” satırında hemen sonra tekrar bir tmpfs kelimesi bulunur. Burada normalde /dev/ram0, /dev/mmcblk0p1, /dev/sda1 gibi cihaz isimleri bulunmalıdır. Fakat burada bir cihaz yoktur. Bundan dolayı buraya ya tmpfs ya da none kelimeleri girilir. Böylece çekirdek nasıl bir cihaz üzerinde çalışması gerektiğini bilir.

14 veya 15. satırlarda verilen komutlar ile mevcut tmpfs sisteminin kapasitesi öğrenilebilir. Aynı zamanda bu kapasite mount edildikten sonra da 16. satırda verildiği gibi değiştirilebilir. Örnekte mevcut kapasite 16MB’tan 32MB’ta çıkarılmaktadır. Kapasite eksiltme de yapılabilir.

Dikkat edilmesi gerek bir başka husus daha vardır. tmpfs üzerinde açıkça bir dosya sistemi kurmadığımız için, 14. satırdaki df komutunda gösterildiği gibi boş dosya sistemindeki başlangıç kullanım miktarı, yani “Used” kolonunda gösterilen miktar 0′dır. Yani dosya sistemi kendi iç yapısı için bir alan tahsis etmemiştir. ext2 veya ext3 gibi bir sistemde dosya sistemi boş dahi olsa, iç yapılardan dolayı, oturulan alanın belirli bir yüzdesi hemen doldurulur. Bu da kapasite kaybına sebep olur.

2. Initramfs

Çekirdeğin, açılış sırasında kullandığı tmpfs dosya sisteminin initramfs olduğundan daha önce bahsetmiştik. Initramfs sisteminden daha önce initrd sistemi mevcut idi. Fakat initrd sistemi artık tamamı ile kullanımdan kalkmış ve yeni nesil tmpfs sistemleri kullanılmaya başlamıştır. Fakat eski ile uyumluluk adına çekirdek halen initrd tabanlı dosya sistemlerini de yüklemektedir. Fakat yeni dağıtımlarda initrd sistemi artık kullanılmamaktadır.

Initramfs sisteminin amacı, esas kök dosya sistemine geçmeden evvel gerekli bazı hazırlıkları yapmaktır. Örneğin gerekli modüllerin önceden yüklenmesi gibi. Ama kök dosya sistemi yeteri kadar ufaksa, esas kök dosya sistemi olarak Initramfs sistemi kullanılabilir. Initramfs sistem iki farklı biçimde kurulabilir.

Birinci durumda bütün kök dosya sistemi çekirdeğe doğrudan gömülür. Derleme sırasında kök dosya sistemi çekirdek kodunun içine eklenir. Böylece sadece çekirdeği taşımak yeterlidir. Kök dosya sistemi çekirdek içinde yapışık olarak gezer. Fakat bu yöntemde kök dosya sistemindeki en ufak bir güncellemede çekirdeği yeniden derlemek gerekir. Ayrıca kök dosya sistemindeki bütün programların GPL veya benzeri lisanslara sahip olması gerekir. Bundan dolayı bu tekniği başka bir yazıya bırakacağız.

İkinci durumda, bütün kök dosya sistemi çekirdekten bağımsız bir dosya olarak kurulur. Açılış sırasında çekirdek bu dosyayı tmpfs içine açar ve mount eder. Kurulumu biraz zor olsa da, bu yazıda bu çeşit kurulumdan bahsedilecektir.

İster çekirdekten bağımsız, ister çekirdek ile beraber kurulsun, bütün güzeller gibi, tmpfs destekli initramfs sisteminin de kusurları vardır. Önce iyi taraflarını özetlersek,

- initramfs şeklinde kurulan kök dosya sisteminin ani kapanmalarda bozulma ihtimali hiç yoktur.
- Kök dosya sistemi her zaman read/write bağlanmıştır. Böylece yazılabilir dizinler için ayrı bir çalışma yapmaya gerek yoktur.
- Bütün kök dosya sistemi tek bir dosya içinde paketlenmiştir. Taşınması, yedeklenmesi, acil durumlarda yedeğinin bulundurulması ve kullanıma alınması çok kolaydır.
- Sistem kapandığında, kök dosya sistemi dahil bütün bilgiler yok olur. Gereksiz log'lar ve gereksiz geçici dosyalar da kendiliğinden yok olur. Bunları temizlemek veya yönetmek için ayrı bir düzen kurmaya gerek yoktur.
- Eğer varsa, ki gömülü sistemlerde genelde yoktur, dosya sistemi swap edilebilir. Diğer bir deyişle, yetersiz bellek durumunda, RAM'daki bilgiler swap alanına geçici olarak yazılabilir.
- Gerekli olan bellek miktarını RAM'dan hemen çalmazlar. İhtiyaç oldukça büyür, ihtiyaç oldukça küçülürler.
- Okuma/Yazma hızı inanılmaz derecede yüksektir. Özellikle aşırı I/O yapan prosesler için tmpfs sistemi çok uygundur.
- Çekirdek içinde, initramfs'i destekleyen çok özel bir yapı bulunur. Ayrı bir dosya sistemine gerek duyulmaz.
- Esas kök dosya sistemine geçişi çok basittir. Muallak işlemler mevcut değildir.

Yukarıda dizilen faydaların bir kısmı, RAM tabanlı bütün dosya sistemleri için geçerlidir. Fakat tamamı sadece tmpfs/initramfs için geçerlidir. Initramfs yönteminin

bazı sakıncaları da aşağıdaki gibi özetlenebilir…

- Kök dosya sistemi doğrudan RAM'da oturduğu için yeterli RAM bellek mevcut olmalıdır. Bundan dolayı çok büyük kök dosya sistemleri bu yöntem ile kurulmaz.
- Tek bir satırlık güncelleme için bile bütün kök dosya sistemi yeniden kurulmalıdır.
- Kalıcı bilgilerin saklanabilmesi için MMC veya Flash bellek gibi kalıcı veri saklayan ortamlara ihtiyaç duyarlar. Çünkü sistem kapandığında bütün bilgiler yok olacaktır.

3. Kök Dosya Sisteminin Initramfs İçin Hazırlanması

Bu yazı dizisinin bir önceki bölümünde kök dosya sistemi MMC kartın 2. bölümünde oturmaktaydı. Kök dosya sistemi initramfs şeklinde olacağı için artık MMC kartın 2. bölümüne gerek olmayacaktır. Diğer bir deyişle bütün işler MMC kartın vfat ile formatlanmış 1. bölümünde yapılacaktır.

Bir önceki bölümde MMC kartın 1. bölümünde, MLO, u-boot.img, uEnv.txt ve uImage dosyaları bulunmaktaydı. Initramfs destekli açılışta bu dosyalardan uEnv.txt hariç, bütün dosyalar aynen korunacaktır.

Özetle, u-boot ve çekirdek derlenmeyecektir. Bir önceki bölümde elde edilen MLO, u-boot.img ve uImage dosyaları aynen kullanılacaktır. Fakat u-boot sistemi initramfs yöntemine göre Linux sistemini açacağından u-boot için açılış komutları farklı olacaktır. Bundan dolayı uEnv.txt dosyası yeniden yazılacaktır.

Ayrıca kök dosya sistemi cpio arşivi olarak paketlenecek ve uInitrd ismi ile MMC kartın 1. bölümüne kopyalanacaktır. Böylece bir önceki sistemden farklı olarak uInitrd dosyası ilk defa vfat sistemine eklenecektir. Ayrıca uEnv.txt güncellenecektir.

Şimdi öncelikle kök dosya sistemi initramfs tekniğine göre aşağıdaki gibi hazırlanmalıdır. Örnek kök dosya sisteminin RootFS içinde olduğu kabul edilmektedir.

```
19 # cd RootFS
20 # find . | cpio -H newc -o | gzip -v -9 > ../initramfs.cpio
21 # cd ..
22 # mkimage -A arm -O linux -T ramdisk -C none -a 0 -e 0 -n U
             -d initramfs.cpio.gz uInitrd
22 # ls -l uInitrd
   -rw-r--r-- 1 root root 1528595 Nov 25 19:27 uInitrd

23 # rm initramfs.cpio.gz
```

19. satırda, örnek kök dosya sisteminin içine girilmektedir. RootFS ile verilen kök dosya sisteminin 1. bölümde verilen kök dosya sistemi ile çok az farklılıkları

bulunmaktadır. Bu farklılıklar aşağıdaki paragraflarda ayrıca incelenecektir.

20. satırda RootFS içindeki bütün dosyalar cpio arşivi şeklinde paketlenmekte ve sonra sıkıştırılmaktadır. Bu komutu açıklaması için bu yazının 9 numaralı bölümüne bakılabilir. Burada ayrıca açıklanmayacaktır.

U-Boot'un kullanacağı bütün dosyalar, ASCII tabanlı dahi olsa, u-boot imajı haline getirilmelidir. 22.satırda initramfs.cpio.gz dosyası, mkimage ile u-boot imajı haline getirilir. Bu komutu açıklaması için bu yazının 9 numaralı bölümüne bakılabilir.

mkimage komutu, initramfs.cpio.gz dosyasının baş tarafına 64 baytlık u-boot başlığı ekler. Elde edilen bu yeni dosyaya uInitrd ismi verilmiştir. Genelde ramdisk destekli açılışlarda, eskiden kalma bir alışkanlık olarak initrd ismi verilir. uInitrd dosyası, yani paketlenmiş, sıkıştırılmış ve u-boot imajı haline getirilmiş kök dosya sistemi, 22. satırda görüldüğü gibi ortalama 1.5MB yer kaplamaktadır. Eğer bu değer çok büyükse, initramfs destekli açılış kullanılmamalıdır.

Sıkıştırılmış arşiv ile işimiz kalmadığı için, 23. satırda ilgili dosya silinebilir.

Bir sonraki adımda, uInitrd dosyası MMC kartın 1. bölümüne kopyalanacaktır. Fakat bu kopyalama işlemine geçmeden evvel yeni kök dosya sisteminde, Initramfs için bir kaç değişiklik yapılmıştır. Bu değişikliklerden kısaca bahsedelim.

Öncelikle, initramfs destekli kök dosya sistemleride, çekirdek hemen /init dosyasını arar ve işletir. Bizler /init dosyasını /sbin/init'e sembolik olarak yönlendirdik. Böylece çekirdek otomatik olarak /sbin/init dosyasını çalıştıracaktır. "$ ls -l RootFS" girilerek sembolik bağ görülebilir.

Eğer /init programı bulunamazsa, ya da mevcut olup da çalıştırılamazsa ne olur? initramfs sistemi, esas kök dosya sistemi değildir. Çekirdek init'i çalıştıramazsa esas kök dosya sistemini bağlamaya çalışır. Esas kök dosya sistemi açılış parametrelerinde "root=…" ile verilen bölüm üzerinden mount edilmeye çalışılır. Onu da mount edemezse panikler. Bu yazımızda aynı anda hem initramfs hem de esas kök dosya sistemi kullanılmayacaktır. Standard Linux dağıtımlarında çok fazla kullanılmasına rağmen, gömülü sistemlerde her ikisini birden, yani hem initramfs hem de esas kök dosya sistemini beraber kullanmak pek mantıklı değildir. Bizler initramfs sistemini, esas kök dosya sistemi olarak kullanacağız.

Kök dosya sistemindeki diğer değişiklik ise /etc/rcS üzerinde yapılmıştır. Yapılan bu değişiklik kısaca aşağıda verilmiştir.

```
mount -t sysfs  sysfs  /sys
mdev -s
```

mount komutunda sysfs dosya sistemi bağlanmaktadır. sys dosya sistemi, aynı proc gibi sözde dosya sistemidir (psuedo file system). proc dosya sisteminden farklı olarak, daha çok donanım verisi barındırır. proc dosya sistemi ise daha çok prosesler hakkında bilgi barındırır. Ayrıca pek çok yerde her iki dosya sistemi birbirinin içine geçmiştir.

mount komutundan sonra mdev komutu gelir. mdev komutu sys dosya sistemini tarar ve sistemde tanımlı bütün cihazları /dev dizini altında yaratır. Burada saç baş yolduran pek çok incelik yatar. Bunlardan kısaca bahsedecek olursak…

- Çekirdek aslında devtmpfs dosya sistemini kendi içinde kurar. Hatta çekirdek derlenirken devtmpfs dosya sistemini mount etmesini söyleriz. Fakat çekirdek burada devtmpfs'i asla mount etmez. Çünkü, uylaşım gereği, devtmpfs sistemi, initramfs'in işi bittikten ve esas kök dosya sistemine geçildikten sonra bağlanır. Fakat bizim esas kök dosya sistemimiz initramfs olacağı için hiç bir zaman devtmpfs bağlanmayacaktır. Bundan dolayı, eğer initramfs kullanılacaksa, çekirdekte "devtmpfs sistemini bağla" babından yapılan seçimin bir anlamı olmayacaktır ki help kısmında bu açıkça söylenir.
- mdev çalışmadan önce çalışan bazı programların /dev/null, /dev/console gibi cihazlara ihtiyaçları olur. Garanti olması için kök dosya sisteminde bu cihazlar önceden el ile kurulmuştur. Böylece mdev komutundan önce /dev dizininde bir cihaza ihtiyaç duyan programlar hata vermeyecektir. "$ ls -l RootFS/dev" ile önceden tanımlı cihazlar incelenebilir.
- Açılış sırasına /dev altında tanımlı bazı cihaz adlarına gerek olabilir. Bu cihazlar henüz sys içinde mevcut olmayabilirler. Bu durumda, açılışta ve ilerde ihtiyaç duyulabilecek cihazlar mknod komutu ile rcS içinde tek tek tanıtılmalıdır.

Artık kök dosya sistemimiz açılışa hazırdır. Sonraki adımda u-boot komutları uEnv içine girilecek ve bütün dosyalar MMC kartın 1. bölümüne yazılarak açılış gerçeklenecektir.

4. uEnv.txt'nin Kuruluşu

Açılışı gerçekleştirecek u-boot komutları uEnv.txt dosyasına yazılır. Dosya içeriği aşağıda verilmiştir.

```
param=setenv bootargs console=ttyO0,115200n8 root=/dev/ram0
files=fatload mmc 0:1 80007FC0 uImage && fatload mmc 0:1 81000
uenvcmd=run param ; run files ; bootm 80007FC0 81000000
```

initramfs'den boot etmenin pek çok yolu vardır.Yukarıda verilen yöntem en iyisi değildir ama en açık olanıdır.

Ayrı bir iş olmasın diye bu bölümde çekirdeği derlemedik. Çekirdek derlemesi sırasında “Kernel command line” satırında açılış parametreleri bulunur. Bu parametreler ile çekirdek initramfs tekniğine göre açılamaz. Çünkü root tanımı hatalıdır. Burada root olarak mmcblk0p2, yani MMC kartın 2. bölümü tanımlıdır. Fakat biz initramfs ile açmak istiyoruz. Kök dosya sisteminin, çekirdek tarafından initramfs olarak anlaşılması için root=/dev/ram0 olmalıdır. Çekrideği de yeniden derlemek istemediğimiz için, çekirdek parametrelerini u-boot sisteminin bootargs değişkeni ile atayabiliriz.

Eğer el yordamı ile sistemi açıyor olsaydık, u-boot promptundan

setenv bootargs console=ttyO0,115200n8 root=/dev/ram0

girmemiz yeterli olacaktı. Böylece “console=ttyO0,115200n8″ ve “root=/dev/ram0″ parametreleri u-boot tarafından, açılış sırasında çekirdeğe aktarılırdı. Aynı işi uEnv.txt içinde yapabilmek için ilgili satırı bir değişkene atayıp, run ile çalıştırmak yeterli olacaktır. Örnekte param ve “run param” ile bu işlem yapılmaktadır.

Benzer şekilde “run files” ile de files değişkeni içinde bulunan komutlar yürütülür. Bu komutların önceki fatload komutlarından bir farkı yoktur.

“fatload mmc 0:1 80007FC0 uImage” komutu ile uImage imajı XIP adresine yüklenir.

“fatload mmc 0:1 81000000 uInitrd” komutu ile de kök dosya sisteminin imajı 81000000 adresine yüklenir. İki adresin içeriğinin birbirlerini ezmemesi gerekir. Genelde initramfs açılışı için 81000000 adresi tercih edilir.

Nihayetinde “bootm 80007FC0 81000000″ komutu ile açılış başlatılır. bootm’nin önündeki ilk adres çekirdeğe ikincisi ise initrd’ye aittir. Her iki adresin de fatload komutlarında kullanıldığını hatırlatalım. Genelde bu tür adresler değişkenlere atanır, öyle de yapılmalıdır. Fakat betiği kısa ve kolay okunabilir tutmak için adresler 2 kez yazılmıştır. Bu iyi bir yöntem değildir.

bootm (boot from memory) komutu ne yapar?

bootm komutu öncelikle çekirdeği 80007FC0 adresine yükler. Sonra 81000000 ile gösterilen initrd imajının doğru olup olmadığını kontrol eder. Her ikisi de checksum denetiminden geçtikten sonra, 81000000 adresi, çekirdeğe “initrd” adresi olarak gönderilir. Sonra da yürütme çekirdeğe geçer. Çekirdek root=/dev/ram0 parametresi sayesinde initramfs veya initrd ile açılacağını anlar. Sonra u-boot’un kendisine geçirdiği initrd adresi sayesinde initrd dosyasını bellekte bulur. gunzip ile açar. cpio ile paketi çözer ve kendi içinde bulunan tmpfs dosya sistemine bunları kopyalar. Sonra /init programını yürütür ve login gelene kadar klasik aşamalardan geçilir.

5. MMC Kartın Hazırlanması ve Açılış.

Şu ana kadar bahsi geçen bütün dosyalar MMC kartın 1. bölümüne aşağıdaki gibi yazılır. Eğer 1. bölümde vfat varsa 24. satır atlanabilir.

```
24 # mkfs.vfat -F32 -n "UcanLinux" /dev/sdb1
25 # mount /dev/sdb1 /mnt/vfat

26 # cp u-boot/MLO                  /mnt/vfat
27 # cp u-boot/u-boot.img           /mnt/vfat
28 # cp linux/arch/arm/boot/uImage /mnt/vfat
29 # cp uInitrd                     /mnt/vfat
30 # cp uEnv.txt                    /mnt/vfat

31 # df /mnt/vfat
   Filesystem     1K-blocks  Used Available Use% Mounted on
   /dev/sdb1          64511  4156     60356   7% /mnt/vfat

32 # ls -l  /mnt/vfat

   total 4155
   -rwxr-xr-x 1 root root   71882 Nov 25 20:33 MLO
   -rwxr-xr-x 1 root root  230500 Nov 25 20:33 u-boot.img
   -rwxr-xr-x 1 root root     191 Nov 25 20:33 uEnv.txt
   -rwxr-xr-x 1 root root 2421872 Nov 25 20:33 uImage
   -rwxr-xr-x 1 root root 1528595 Nov 25 20:33 uInitrd

33 # umount /mnt/vfat
```

MMC kart, Beaglebone'a takılır ve cihaz açılır. Açılışın özeti aşağıda verilmiştir.

```
U-Boot SPL 2012.10 (Nov 08 2012 - 21:58:26)
...
Hit any key to stop autoboot:  0
SD/MMC found on device 0
reading uEnv.txt      <== uEnv.txt dosyası okunuyor.

191 bytes read
Loaded environment from uEnv.txt
Importing environment from mmc ...
Running uenvcmd ...    <== uenvcmd çalıştırılıyor.
reading uImage       <== 1. fatload komutu çalışıyor.

2421872 bytes read
reading uInitrd    <== 2. fatload komutu çalışıyor.
```

```
1528595 bytes read
## 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
## Loading init Ramdisk from Legacy Image at 81000000 ...
   Image Name:   UcanLinux
   Image Type:   ARM Linux RAMDisk Image (uncompressed)
   Data Size:    1528531 Bytes = 1.5 MiB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
   XIP Kernel Image ... OK
OK

Starting kernel ...

Uncompressing Linux... done, booting the kernel.
...
Kernel command line: console=ttyO0,115200n8 root=/dev/ram0
...
mmcblk0: mmc0:1234 SA04G 3.67 GiB
 mmcblk0: p1 p2

Filesystem           1K-blocks      Used Available Use% Mounte
tmpfs                   127468         0    127468   0% /dev/s

http://ucanlinux.com
UcanLinux login: root
Password: root
```

root/root ile sisteme giriş yapılır. free komutu ile kullanılan bellek miktarı aşağıdaki gibi incelenebilir. Çıkşın Used satırından da görülebileceği gibi çekirdek ve kök dosya sisteminin tamamı 9MB'tan az yer kaplamaktadır.

```
root@UcanLinux:~ # free
             total         used         free       shared
Mem:        254936         8740       246196            0
-/+ buffers:               8740       246196
Swap:            0            0            0
```

df komutu ile, aşağıdaki gibi bağlı bölümler incelenebilir. Görüleceği gibi kök dosya sistemi açık olarak gözükmez. Sadece /dev/shm cihazı raporlanmaktdır. Bu cihaz

tmpfs üzerinde oturduğu için, kapasite olarak fiziksel belleğin yarısına sahiptir. Ama gerçekte kullanılan miktar 0 bayttır. Bütün tmpfs sistemleri gibi gerekli oldukça büyüyecektir.

```
root@UcanLinux:~ # df
Filesystem           1K-blocks      Used Available Use% Mounte
tmpfs                   127468         0    127468   0% /dev/s
```

df komutu, bağlı bölümleri listelelerken filtreleme yapar. Bütün bağlı bölümler, aşağıdaki gibi “df -a” veya “mount” komutu ile incelenebilir. Kolay okunması için araya boşluklar eklenmiştir.

```
root@UcanLinux:~ # df -a

Filesystem           1K-blocks      Used Available Use% Mounte
sysfs                        0         0         0   0% /sys
proc                         0         0         0   0% /proc
devpts                       0         0         0   0% /dev/p
tmpfs                   127468         0    127468   0% /dev/s

root@UcanLinux:~ # mount

rootfs on /          type rootfs (rw)
sysfs  on /sys       type sysfs  (rw,relatime)
proc   on /proc      type proc   (rw,relatime)
devpts on /dev/pts   type devpts (rw,relatime,mode=600)
tmpfs  on /dev/shm   type tmpfs  (rw,relatime)
```

mount çıkışından da görüleceği gibi, kök dosya sistemi rootfs olarak gözükmektedir. Diğer bütün dosya sistemleri, açılışta /etc/rcS betiği içinde bağlanılmaktadır. Genelde bağlama işi /etc/fstab dosyası yardımı ile yapılır. Ama mount işlemlerinin, /etc/rcS içinde, tek tek yazılarak yapılması gömülü sistemler için bizce daha uygundur. Çünkü yönetim tamamen bizdedir ve kullanılan dosya sayısı 1 adet azalacaktır.

Böyle bir sistem, ani kapanmaya karşı %100 dayanıklıdır. Çünkü her açılışta kök dosya sistemi yeniden kurulur. Bu sistemin en kötü taraflarından birisi, daha önce de bahsettiğimiz gibi, kök dosya sisteminin güncellenmesidir. Kök dosya sistemindeki en ufak bir değişiklikte, bütün kök dosya sistemi baştan kurulmalıdır.

6. Kök Dosya Sisteminin Güncellenmesi

Herhangi bir güncelleme, RootFS altında yapılır ve 3. başlıkta anlatıldığı gibi uInitrd kurulur. Eğer elimizde RootFS yok ve sadece uInitrd varsa ki, çalışma paketinde zaten böyledir, RootFS aşağıdaki gibi elde edilir ve güncellenir. Kolay okunması için çıkışlara

boşluk ve yeni satırlar eklenmiştir.

```
34 # file uInitrd
   uInitrd: u-boot legacy uImage,
            UcanLinux,
            Linux/ARM,
            RAMDisk Image (Not compressed),
            1528531 bytes,
            Sun Nov 25 21:07:53 2012,
            Load Address: 0x00000000,
            Entry Point: 0x00000000,
            Header CRC: 0x40CCEDDF,
            Data CRC: 0x4AF54359

35 # dd if=uInitrd of=Initrd.gz bs=64 skip=1
   23883+1 records in
   23883+1 records out
   1528531 bytes (1.5 MB) copied, 0.080854 s, 18.9 MB/s

36 # file Initrd.gz
   Initrd.gz: gzip compressed data,
              from Unix,
              last modified: Sun Nov 25 21:07:52 2012,
              max compression

37 # gunzip Initrd.gz

38 # file Initrd
   Initrd: ASCII cpio archive (SVR4 with no CRC)

39 # mkdir RootFS
40 # cd RootFS

41 # cpio -i < ../Initrd
   5858 blocks

42 # ls -l
   total 52
   drwxr-xr-x 2 root root 4096 Nov 25 21:16 bin/
   drwxr-xr-x 2 root root 4096 Nov 25 21:16 dev/
   drwxr-xr-x 2 root root 4096 Nov 25 21:16 etc/
   drwxr-xr-x 2 root root 4096 Nov 25 21:16 home/
   lrwxrwxrwx 1 root root   10 Nov 25 21:16 init -> /sbin/init
   drwxr-xr-x 3 root root 4096 Nov 25 21:16 lib/
   drwxr-xr-x 2 root root 4096 Nov 25 21:16 mnt/
   drwxr-xr-x 2 root root 4096 Nov 25 21:16 proc/
   drwxr-xr-x 2 root root 4096 Nov 25 21:16 root/
```

```
drwxr-xr-x 2 root root 4096 Nov 25 21:16 sbin/
drwxr-xr-x 2 root root 4096 Nov 25 21:16 sys/
drwxr-xr-x 2 root root 4096 Nov 25 21:16 tmp/
drwxr-xr-x 4 root root 4096 Nov 25 21:16 usr/
drwxr-xr-x 2 root root 4096 Nov 25 21:16 var/
```

file komutu dosyanın cinsini belirtir. Aynı zamanda eğer tespit edebilirse, dosya formatı hakkında özet bilgi verir. Yeri gelmişken hatırlatalım, file komutu her zaman doğru sonucu vermeyebilir.

34. satırda, Initrd dosyasının cinsi tespit edilir. file komutu dosyanın bir u-boot imajı olduğunu söyler. U-Boot imajlarını dışarıda kullanabilmek için başlıklarını atmak gerekir. Başlık boyu her zaman 64 bayttır. Eğer baştaki 64 baytı atarsak, geri kalan dosyayı kendi amaçlarımızı için kullanabiliriz.

35. satırdaki komut, başlıktaki 64 baytı atar ve nihai dosyaya Initrd.gz ismini verir. 36. satırdaki file komutu, elde edilen dosyanın gerçekten bir gzip dosyası olduğunu teyit eder.

37. satırdaki komut Initrd.gz dosyasını açar ve Initrd isimli arşiv dosyası elde eder. 38. satırdaki komut elde edilen dosyanın, bir cpio arşiv dosyası olduğunu teyit eder.

41. satırdaki cpio komutu, ilgili arşivi RootFS dizini altına açar. Bu komuttaki -i seçeneği “input” anlamındadır ve extract olarak düşünülmelidir. cpio komutu aynı anda ya input ya da output modunda işler. Linux’da hiç ısınamadığım, kullanımı kendinden menkul bir garip komuttur. Fakat initramfs sisteminin temelidir ve bu komut doğrudan çekirdek kodunun içine gömülüdür.

42. satırda, RootFS sisteminin içeriği listelenmiştir. Kök dosya sisteminde gerekli güncellemeler yapıldıktan sonra 3. başlıkta anlatıldığı gibi tekrar uInitrd elde edilebilir.

7. Diğer

Bu çalışmada elde edilen bazı dosyalar buradan indirilebilir.

Yazara **nazim** at **ucanlinux.com** adresinden ulaşılabilir. Ya da bu yazı doğrudan blog’dan okunuyursa blog’un 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.

26.Kasım.2012: ilk yayın

— 2. bölümün sonu —