Gömülü Linux Sistemleri

Login'e Kadar Linux

ARM Tabanlı Bir Sistemin Kuruluşu ve Emülatör ile Testi

Giriş

Bu yazıda ARM tababanlı bir sistem baştan sona kurulacak ve emülatör ortamında test edilecektir.

Gömülü sistemler kurulurken genelde MMC kart veya flash disk gibi ortamlara kurulur ve geliştirme bordları üzerinde test edilirler. Bu hem maliyetli hem de yavaş işleyen bir süreçtir. Öncelikle elimizde fiziksel olarak bir bordun bulunması gerekir. Elimizde bord yokken gömülü sistemin kuruluşu ve testi nasıl yapılabilir?

Bu sorunun çözümü için çeşitli emülatörler geliştirilmiştir. Bu yazımızda emülatör kullanılanarak Linux işletim sistemi baştan sona kurulacak ve test edilecektir. Yapılan işlemler gerçeğe o kadar yakın olacaktır ki, bir iki basit değişiklikle gerçek ortama geçilebilecektir.

1. Yazının Kaynağı

Uzun zamandır, gömülü sistemleri uzaktan yöneten bir yazılım üzerinde çalışmaktayız. Yazılımın testlerini yapabilmek için ağ ortamında çalışan onlarca ARM, PPC veya MIPS işlemcili makineye ihtiyaç duyduk. Pratik olarak böyle bir ortama sahip pek az labratuvar veya çalışma ortamı mevcuttur.

Böyle bir test ortamını gerçeğe en yakın olarak kurmanın en pratik yollarından birisi emülatörleri kullanmaktır. Her bir emülatör fiziksel olarak ayrı bir makine gibi çalışır. Bütün bu makinler sanal bir bridge ile ağ ortamında birleştirilebilir. Bu durumda

gerçeğe çok yakın bir test ortamı sıfır maliyet ile kurulabilir.

Örneğin tek bir notebook bilgisayar içinde aynı anda 20 adet emülatör işletilerek 20 farklı makine varmış gibi test yapılabilir. Bu yazıda, test için kurulan böyle bir ortama ait, tek bir sanal makinenin kuruluş ve testi gerçeklenecektir. Başka bir yazıda ise bu makinelerin sanal bir ağ ortamında nasıl birleştirileceğindan bahsedilecektir.

2. Yapılacakların Özeti

Öncelikle uygun bir emülatör seçilecektir. Linux altında çalışan pek çok emülatör mevcuttur. Bizler uzun zamandır kullandığımız ve inanılmaz derecede basit ve kullanışlı olan qemu emülatörünü kullanacağız. qemu, “quick emulator” kelimelerinden türetilmiştir.

Daha sonra, bu yazının yazıldığı sırada taze çıkmış Linux 3.8.2 çekirdeği kullanılarak, ARM için derleme yapılacaktır. Peşinden kök dosya sistemi sanal bir disk üzerinde, imaj olarak kurulacak ve bütün bu yapılanlar emülatör ortamında test edilecektir. Nihayetinde evsahibi sistem ile misafir sistem arasında ağ bağlantısı kurulacak ve test edilecektir.

Geliştirme yapılan sisteme evsahibi sistem denir. Örneğin benim x86 makinem evsahibi sistemdir. Geliştirme burada yapılmaktadır. Evsahibi sistem içinde emülatör yardımı ile çalışan sisteme de misafir sistem denir. Genelde misafir sistemler ARM veya PPC gibi farklı bir hedef mimariye sahiptir. Ama bu zorunlu değildir. Örneğin x86 mekine içinde yine x86 emülatörü kullanılabilir.

3. QEmu Sistemin Kurulması

En son qemu sistemi aşağıdaki gibi yüklenip, kurulabilir. Paket yönetimi kullanan dağıtımlar kendi paket yöneticileri ile doğrudan yükleme yapabiliriler. Bizler her zamanki gibi kendimize eziyet edip en son sürümü aşağıdaki gibi indirip derleyeceğiz.

```
$ git clone git://git.qemu-project.org/qemu.git
$ cd qemu/
$ ./configure --prefix=/opt/qemu
$ make -j2
$ make install
```

qemu pekçok işlemciye destek vermektedir. Aslında derleme işlemi sadece ARM için yapılabilir. Fakat bizce buna gerek yoktur. Belki bir gün gerekli olur diye bütün işlemcileri derlemeye katmaktayız. configure sırasında işlemci ismi verilmezse, desteklenen bütün işlemciler için emülatör programı üretilmektedir. Üretilen bütün sistem prefix ile verilen /opt/qemu altına yüklenmektedir. Okuyucu kendine uygun farklı bir dizin seçebilir. prefix’in mutlaka verilmesi tavsiye edilir. prefix verildiğinde,

örneğin bütün sistem /opt/qemu altına atılmaktadır. Eğer prefix verilmezse, bütün qemu sistemi son derece dağınık bir biçimde kurulmaktadır.

Bütün emülatörler $PREFIX/bin altında kurulur. Bizim örneğimizde emülatörler /opt/qemu/bin altındadır. Bu dizin PATH içine eklenmelidir ki her yerden çağırabilelim. bin dizininin içeriği aşağıda verilmiştir.

```
$ ls /opt/qemu/bin -l
total 123420
-rwxr-xr-x 1 root root  884776 Mar 4 23:28 qemu-alpha
-rwxr-xr-x 1 root root 1228792 Mar 4 23:28 qemu-arm
-rwxr-xr-x 1 root root 1232440 Mar 4 23:28 qemu-armeb
-rwxr-xr-x 1 root root  808680 Mar 4 23:28 qemu-cris
-rwxr-xr-x 1 root root  970694 Mar 4 23:27 qemu-ga
-rwxr-xr-x 1 root root 1232012 Mar 4 23:28 qemu-i386
-rwxr-xr-x 1 root root 2545092 Mar 4 23:27 qemu-img
-rwxr-xr-x 1 root root 2538288 Mar 4 23:27 qemu-io
-rwxr-xr-x 1 root root  897640 Mar 4 23:28 qemu-m68k
-rwxr-xr-x 1 root root  783904 Mar 4 23:28 qemu-microblaze
-rwxr-xr-x 1 root root  780384 Mar 4 23:28 qemu-microblazeel
-rwxr-xr-x 1 root root 1294500 Mar 4 23:28 qemu-mips
-rwxr-xr-x 1 root root 1292836 Mar 4 23:28 qemu-mipsel
-rwxr-xr-x 1 root root 2449069 Mar 4 23:27 qemu-nbd
-rwxr-xr-x 1 root root  737088 Mar 4 23:28 qemu-or32
-rwxr-xr-x 1 root root 1697176 Mar 4 23:28 qemu-ppc
-rwxr-xr-x 1 root root 2372524 Mar 4 23:28 qemu-ppc64
-rwxr-xr-x 1 root root 2346956 Mar 4 23:28 qemu-ppc64abi32
-rwxr-xr-x 1 root root 1001692 Mar 4 23:28 qemu-s390x
-rwxr-xr-x 1 root root  835844 Mar 4 23:28 qemu-sh4
-rwxr-xr-x 1 root root  838788 Mar 4 23:28 qemu-sh4eb
-rwxr-xr-x 1 root root  893600 Mar 4 23:28 qemu-sparc
-rwxr-xr-x 1 root root 1048156 Mar 4 23:28 qemu-sparc32plus
-rwxr-xr-x 1 root root 1087432 Mar 4 23:28 qemu-sparc64
-rwxr-xr-x 1 root root 3618064 Mar 4 23:27 qemu-system-alpha
-rwxr-xr-x 1 root root 4853104 Mar 4 23:27 qemu-system-arm
-rwxr-xr-x 1 root root 2485616 Mar 4 23:28 qemu-system-cris
-rwxr-xr-x 1 root root 4779312 Mar 4 23:27 qemu-system-i386
-rwxr-xr-x 1 root root 2441520 Mar 4 23:28 qemu-system-lm32
-rwxr-xr-x 1 root root 3376720 Mar 4 23:28 qemu-system-m68k
-rwxr-xr-x 1 root root 2445488 Mar 4 23:28 qemu-system-microbl
-rwxr-xr-x 1 root root 2445488 Mar 4 23:28 qemu-system-microbl
-rwxr-xr-x 1 root root 4672176 Mar 4 23:28 qemu-system-mips
-rwxr-xr-x 1 root root 5118640 Mar 4 23:28 qemu-system-mips64
-rwxr-xr-x 1 root root 5139952 Mar 4 23:28 qemu-system-mips64e
-rwxr-xr-x 1 root root 4672176 Mar 4 23:28 qemu-system-mipsel
-rwxr-xr-x 1 root root 2378544 Mar 4 23:28 qemu-system-or32
```

```
-rwxr-xr-x 1 root root 4994416 Mar 4 23:28 qemu-system-ppc
-rwxr-xr-x 1 root root 5697232 Mar 4 23:28 qemu-system-ppc64
-rwxr-xr-x 1 root root 4977072 Mar 4 23:28 qemu-system-ppcemb
-rwxr-xr-x 1 root root 2773168 Mar 4 23:28 qemu-system-s390x
-rwxr-xr-x 1 root root 3361872 Mar 4 23:28 qemu-system-sh4
-rwxr-xr-x 1 root root 3361872 Mar 4 23:28 qemu-system-sh4eb
-rwxr-xr-x 1 root root 2690352 Mar 4 23:28 qemu-system-sparc
-rwxr-xr-x 1 root root 3677520 Mar 4 23:28 qemu-system-sparc64
-rwxr-xr-x 1 root root 2365584 Mar 4 23:28 qemu-system-unicore
-rwxr-xr-x 1 root root 4996784 Mar 4 23:27 qemu-system-x86_64
-rwxr-xr-x 1 root root 2448144 Mar 4 23:28 qemu-system-xtensa
-rwxr-xr-x 1 root root 2444048 Mar 4 23:28 qemu-system-xtensae
-rwxr-xr-x 1 root root  736788 Mar 4 23:28 qemu-unicore32
-rwxr-xr-x 1 root root 1396444 Mar 4 23:28 qemu-x86_64
-rwxr-xr-x 1 root root   93281 Mar 4 23:27 virtfs-proxy-helper
-rwxr-xr-x 1 root root   50506 Mar 4 23:27 vscclient
```

Bu çıkışta “qemu-” ile başlayan her program ayrı bir emülatördür. Dikkat edilirse “qemu-” ile başlayan isimler iki farklı biçime sahiptirler.

qemu-arm örneğinde olduğu gibi, “qemu-” ile başlayan isimlerin önünde CPU ismi bulunur. Bu tür emülatörler işletim sistemine sahip olmayan, “bare metal” tabir edilen sistemleri test etmek içindir.

Örnek verecek olursa, ARM işlemcisi için bir C programı yazılmış olsun. Bu C programını test etmek için doğrudan qemu-arm isimli emülatör kullanılır. İşin içinde ne boot loader, ne disk imajı ne işletim sistemi vardır, sadece C programı bulunur. “Bare Metal” sistemler her zaman konumuz dışındadır. Bizim esas çıkış noktamız her zaman Linux çekirdeğidir.

“qemu-” ile başlayan diğer emülatörler ise işletim sistemi desteği veren emülatörlerdir. Bizler her zaman “qemu-system-” ile başlayan emülatörle ilgileneceğiz.

Şu an için esas ilgi alanımız ARM işlemcilerdir. Bundan dolayı emülatör olarak qemu-system-arm programı kullanılacaktır. Bu emülatörün desteklemiş olduğu işlemcilerin listesi aşağıdaki gibi elde edilebilir.

```
$ qemu-system-arm -cpu ?
Available CPUs:
 arm1026
 arm1136
 arm1136-r2
 arm1176
 arm11mpcore
```

```
arm926
arm946
cortex-a15
cortex-a8
cortex-a9
cortex-m3
pxa250
pxa255
pxa260
pxa261
pxa262
pxa270-a0
pxa270-a1
pxa270
pxa270-b0
pxa270-b1
pxa270-c0
pxa270-c5
sa1100
sa1110
ti925t
any
```

Aynı zamanda bu emülatörün destek verdiği bordlar aşağıdaki gibi listelenebilir.

```
$ qemu-system-arm -M ?
Supported machines are:
z2 Zipit Z2 (PXA27x)
xilinx-zynq-a9 Xilinx Zynq Platform Baseboard for Cortex-A9
vexpress-a9 ARM Versatile Express for Cortex-A9
vexpress-a15 ARM Versatile Express for Cortex-A15
versatilepb ARM Versatile/PB (ARM926EJ-S)
versatileab ARM Versatile/AB (ARM926EJ-S)
tosa Tosa PDA (PXA255)
lm3s811evb Stellaris LM3S811EVB
lm3s6965evb Stellaris LM3S6965EVB
akita Akita PDA (PXA270)
spitz Spitz PDA (PXA270)
borzoi Borzoi PDA (PXA270)
terrier Terrier PDA (PXA270)
realview-eb ARM RealView Emulation Baseboard (ARM926EJ-S)
realview-eb-mpcore ARM RealView Emulation Baseboard (ARM11MPCo
realview-pb-a8 ARM RealView Platform Baseboard for Cortex-A8
realview-pbx-a9 ARM RealView Platform Baseboard Explore for Co
cheetah Palm Tungsten|E aka. Cheetah PDA (OMAP310)
sx1 Siemens SX1 (OMAP310) V2
```

```
sx1-v1 Siemens SX1 (OMAP310) V1
n800 Nokia N800 tablet aka. RX-34 (OMAP2420)
n810 Nokia N810 tablet aka. RX-44 (OMAP2420)
musicpal Marvell 88w8618 / MusicPal (ARM926EJ-S)
mainstone Mainstone II (PXA27x)
kzm ARM KZM Emulation Baseboard (ARM1136)
integratorcp ARM Integrator/CP (ARM926EJ-S) (default)
highbank Calxeda Highbank (ECX-1000)
connex Gumstix Connex (PXA255)
verdex Gumstix Verdex (PXA270)
nuri Samsung NURI board (Exynos4210)
smdkc210 Samsung SMDKC210 board (Exynos4210)
collie Collie PDA (SA-1110)
none empty machine
```

Son derece geniş bir bord ailesine destek verilmektedir. Projemize ait bord burada yoksa, çok da önemli değildir. En yakın CPU veya en yakın bord seçilerek devam edilmelidir. Bizler bütün bu yazı boyunca ARM9 işlemcili Versatilepb bordunu seçtik. Bu seçimin nerede ise hiç bir öneminin olmadığı yazının ilerleyen bölümlerinde belli olacaktır. Yapılan tek farklı işlem çekirdek derlemesi sırasında uygun config dosyasının seçilmesidir.

$ qemu-system-arm –help girişi ile de seçenekler listesi elde edilebilir. Sadece bu seçenekler listesi bile qemu sisteminin ne kadar derya deniz olduğunu gösterir.

Artık emülatörümüz kurulmuş ve kullanıma hazırdır. İlk yapılacak iş çekirdeğin derlenmesi ve hemen emülatör ile test edilmesidir.

Bu yazı yazıldığı sırada, en son kararlı çekirdek 3.8.2 idi. Tekrar etmekte fayda görmekteyiz. Gömülü sistemlerde asla kararsız çekirdekler kullanılmamalıdır. Çekirdek aşağıdaki gibi indirilip derlenebilir.

```
$ wget https://www.kernel.org/pub/linux/kernel/v3.x/
                      linux-3.8.2.tar.bz2
$ tar jxvf linux-3.8.2.tar.bz2
$ ln -s linux-3.8.2 linux
$ ls -l
total 82648
lrwxrwxrwx 1  root root       11 Mar 5 12:59 linux -> linux-3.
drwxrwxr-x 23 root root     4096 Mar 4 00:04 linux-3.8.2
-rw-r--r-- 1  root root 84624148 Mar 4 00:35 linux-3.8.2.tar.b
```

```
$ CROSS=arm-none-linux-gnueabi-
$ cd linux
```

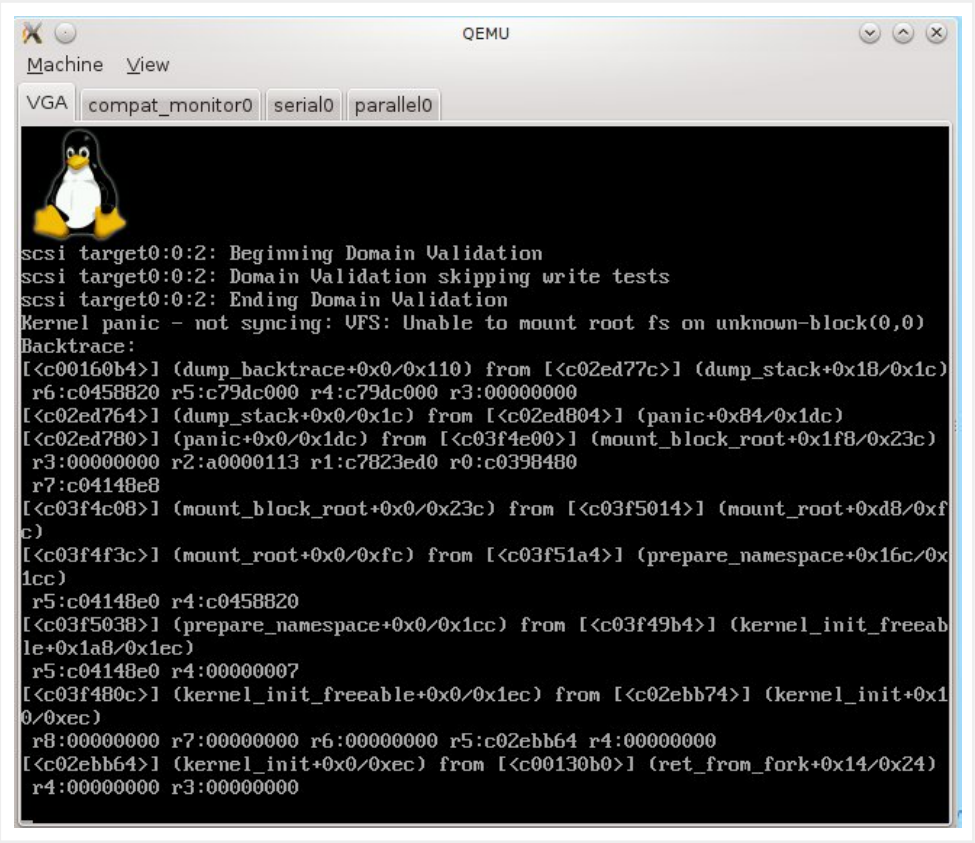
```
$ cp arch/arm/configs/integrator_defconfig .config
$ make ARCH=arm CROSS_COMPILE=$CROSS menuconfig
$ make ARCH=arm CROSS_COMPILE=$CROSS zImage -j2
$ cp arch/arm/boot/zImage /uygun/bir/yer
$ cp .config /uygun/bir/yer/kernel.config
```

Önceki yazılarımızda çok bahsettiğimiz için burada tool-chain ve cross derlemeden bahsedilmeyecektir. Çekirdeğe ait config dosyası çalışma paketi içinde bulunabilir. Okuyucu kendi derlemesini yaptıktan sonra config dosyasını mutlaka uygun bir yerde saklamalıdır ki sonra bütün işler baştan tekrar edilmesin.

Bizler çekirdeğe çok az dosya sistemi desteği verdik. Sadece ext2 dosya sistemi mevcuttur. Ayrıca prensibimiz gereği modül kullanmadık. Okuyucu amacına uygun değişiklikleri kolayca yapabilir.

Çekirdek derlendikten sonra, daha ileri gitmeden hemen test edilmelidir. Test sonucu aşağıda verilmiştir. Tabii ki henüz kök dosya sistemi kurulmadığından çekirdek panikleyecektir.

```
$ qemu-system-arm -M versatilepb -kernel zImage
```



— Çekirdeğin test edilmesi.

Bu tür hatalarda "… rootfs on unknown-block(0,0)" mesajı çok önemlidir. Çekirdek kök dosya sisteminin nerede olduğunu bilemez. Bir biçimde kök dosya sisteminin cihaz ismi verilmelidir. Bu cihaz ismini vermemin pek çok yolu vardır. qemu sisteminde kök dosya sistemi append seçeneği ile verilir. Bu seçenek ile kök dosya sisteminin oturduğu cihazına adı /dev/sda1 gibi verilir. Fakat çekirdek henüz bu isimlerin ne olduğunu bilemez. Çünkü /dev sistemi henüz bağlı değildir. Bundan dolayı çekirdek cihaz ismi yerine cihazın majör/minör numarasını kendi içinde saklar. İşte unknown-block(0,0) mesajında, parantez içinde verilen 0,0 sayıları, aslında çekirdeğin kök dosya sistemini beklediği cihazın majör/minör numaralarıdır.

Tabii ki bizim kök dosya sistemimiz disk üzerinde olacaktır. Fakat her işimiz sanal olduğu gibi diskimiz de sanal olacaktır.

4. Sanal Diskin Kuruluşu

Önce bir sanal disk kurulacak sonra bu sanal disk üzerinde aynen gerçek diskte olduğu gibi bölümlendirme yapılacaktır. İlk yapılacak iş sanal diskin boyunun tespitidir. Örnek kök dosya sistemimiz 6MB tutmaktadır. Düz hesap olması için 16MB'lık bir disk bizim işimizi fazlası ile görür. Aşağıda 16MB'lık diskin kuruluşu ve bölümlendirilmesi verilmiştir. Diskte tek bölüm vardır. Benzer işlemler daha önceki yazılarımızda yapıldığı

için ayrıca incelenmeyecektir.

```
$ dd if=/dev/zero of=disk.img bs=1M count=16
$ fdisk disk.img
```

5. Dosya Sisteminin Kuruluşu

Sanal diskimizin üzerinde tek bir bölüm vardır. Bu bölüme aşağıdaki gibi ext2 dosya sistemi kurulabilir.

```
1 $ losetup /dev/loop0 disk.img -o 1M

2 $ losetup -a
  /dev/loop0: [2050]:9454944 (/nk/blog/disk.img), offset 10485

3 $ mkfs.ext2 /dev/loop0

4 $ losetup -d /dev/loop0

5 $ losetup -a
```

ext2 dosya sistemi mkfs.ext2 komutu ile kurulabilir. Bu komut karşısında fiziksel bir cihaz bulunmasını ister. Fakat elimizde fiziksel bir cihaz yerine disk.img isimli sıradan bir dosya vardır. mkfs.ext2 komutunu bir biçimde kandırmak için loop cihazı kullanılır.

Loop cihazı sıradan dosyaları, karşı tarafa fiziksel bir cihaz gibi gösterir. Diğer bir deyişle onları kandırır. Hikaye çok basittir. mkfs.ext2 programı bir sektör talep ettiği zaman loop cihazı ilgili sektöre karşı gelen kaydı dosyadan okur veya yazar. Diğer bir deyişle fiziksel bir cihazdan talep edilebilecek her işi loop cihazı taklit eder. Okuma/yazma ortamı olarak da disk.img dosyasını kullanır. Böylece gerçek bir cihaz bekleyen programların dosya üzerinde de çalışmalarını sağlar. Tabii ki bu işlerin yapılabilmesi için çekirdekte loop cihaz desteğinin olması gerekir. Bütün dağıtımlarda bu özellik mutlaka mevcuttur.

Yukarıdaki satırlara dönecek olursak, 1 numaralı komut disk.img dosyası ile /dev/loop0 cihazını eşleştirir. Bu eşleştirmeden sonra artık bütün işler /dev/loop0 cihazı üzerinden yapılacaktır. -o seçeneği offset anlamındadır. -o 1M seçeneği ile disk.img dosyasının ilk 1MBaytlık kısmı atlanacak ve ilişkilendirme bu ofsetten sonra başlayacaktır.

Burada dikkat edilmesi gereken 2 önemli konu vadır. Başka bir program /dev/loop0 cihazını tutmuş olabillir. Bu durumda boşta bulunan bir cihaz numarası seçilmelidir. İlk boş cihaz numarası

```
$ losetup -f
```

komutu ile elde edilebilir. Ayrıca -o ile verilen offset değeri sistemden sisteme fark edebilir. Bir önceki yazımızda bu konudan bahsetmiştik. Disklerin ilk sektörleri boot yükleyicileri için ayrılmıştır. Şu anda kullandığımız fdisk programı boot yükleyicileri için 2048 sektör ayırır. fdisk'in eski sürümleri 63 sektör ayırır. Ya da kullanıcılar fdisk programının expert modundan girerek istediği kadar yer ayırabilirler.

Sonuçta -o ile verilen offset değeri için bir uylaşım yoktur. Bunu tam olarak öğrenmenin en basit yolu fdisk ile disk.img imajına girip, p ile bölümleri listelemektir. Birinci bölümün başlangıç adresi, sektör boyu ile çarpılarak -o ile verilecek olan offset değeri elde edilir. Sektör boyu genelde 512 olarak alınır.

2. komut ile yapılan işin doğruluğu kontrol edilir. -a seçeneği o anda kullanılan bütün loop cihazlarının isimlerini ve ilişkili dosyaları listeler. -o 1M ile verdiğimiz offset değeri açıkça listelenmiştir.

3. adımda /dev/loop0 üzerinde dosya sistemi kurulur. Loop cihazı sayesinde, ext2 dosya sistemi aslında disk.img dosyasının 1. bölümüne kurulur. Fakat mkfs.ext2 programı, loop0 cihazı sebebi ile kendini gerçek bir disk ile çalışıyor zanneder.

4. adımda loop0 cihazı ile dosya arasındaki ilişki koparılır. -d seçeneği "delete" anlamındadır. Yine çok yapılan hatalardan birisi cihaz ile dosyayı biribirinden ayırmamaktır. Bu işlem yapılmadan disk.img üzerinde çalışılırsa yapılan bütün işler boşa gidecektir.

5. adımda -a seçeneği ile kullanımdaki loop cihazlarının listesi alınır. /dev/loop0'ın kullanımda olmadığı açıkça görülmelidir.

Dosya sistemimiz artık kurulmuştur. Sonra dosya sistemi içine kök dosya sisteminde bulunması gereken dosyalar kopyalanmalıdır.

6. Kök Dosya Sisteminin Kuruluşu

Kök dosya sistemi busybox destekli kurulacaktır. Daha önce pek çok örneği yapıldığı için kuruluş tekniği üzerinde durulmayacaktır. Çalışma paketi içinden kök dosya sistemi ve busybox'ın config dosyası elde edilebilir.

Örnek kök dosya sisteminde busybox dinamik olarak derlenmiş ve gerekli kütüphaneler lib/ dizini altına kopyalanmıştır. Ayrıca programların testi için strace ve uzaktan güvenli bağlantı için ssh sunucusu veya diğer adı ile dropbear derlenmiş ve kök dosya sistemine eklenmiştir.

Kök dosya sisteminin RootFS dizini altında olduğunu kabul edelim. Bu durumda kök dosya sistemi sanal diskin 1. bölümüne aşağıdaki gibi kopyalanabilir.

```
 7 $ mount -o offset=1M disk.img /mnt/part1

 8 $ cp -a RootFS/* /mnt/part1

 9 $ df /mnt/part1
   Filesystem 1K-blocks Used Available Use% Mounted on
   /dev/loop1 14871 6450 7653 46% /mnt/part1

10 $ ls -l /mnt/part1
   total 27
   drwxr-xr-x 2 root root  2048 Mar  5 17:44 bin
   drwxr-xr-x 2 root root  1024 Nov 25 18:06 dev
   drwxr-xr-x 3 root root  1024 Mar 5 20:42 etc
   drwxr-xr-x 2 root root  1024 Aug 24 2012 home
   lrwxrwxrwx 1 root root    10 Nov 25 18:03 init -> /sbin/in
   drwxr-xr-x 2 root root  1024 Mar 5 20:11 lib
   drwx------ 2 root root 12288 Mar 6 11:30 lost+found
   drwxr-xr-x 2 root root  1024 Aug 24 2012 mnt
   drwxr-xr-x 2 root root  1024 Aug 24 2012 proc
   drwxr-xr-x 2 root root  1024 Aug 24 2012 root
   drwxr-xr-x 2 root root  2048 Mar 5 17:44 sbin
   drwxr-xr-x 2 root root  1024 Aug 25 2012 sys
   drwxr-xr-x 2 root root  1024 Aug 24 2012 tmp
   drwxr-xr-x 4 root root  1024 Aug 25 2012 usr
   drwxr-xr-x 2 root root  1024 Aug 24 2012 var

11 $ umount /mnt/part1
```

7. adımda sanal diskimizin 1. bölümü mount edilir. Aslında mount komutu disk.img'nin bir cihaz olmadığını anlar ve loop yardımı ile mount eder. -o seçeneği ile offset açıkça verilir. Daha önce loop0 ile disk.img'yi nasıl ilişkilendirdiysek, mount komutu da arka planda aynı ilşikiyi kurar. loop cihazına offset değerini gönderir. Fakat burada loop0 cihazı değil, loop1 cihazı kullanılmıştır. Muhtemelen başka bir yerde loop0 cihazı halen kullanımdadır.

8.satırda örnek kök dosya sistemi sanal diskin 1. bölümüne taşınır. -a seçeneği "arşiv" demektir ve cihazlar, sembolik linkler ve alt dizinler dahil her dosyayı taşır.

9. satırda mount komutunun açıkça loop1 cihazını kullandığı görülebilir.

11. satırda mutlaka umount işlemi yapılmalıdır. Eğer yapılmaz ve disk.img dosyası hemen kullanılmaya başlanırsa, dosya sistemi bozulacaktır. Ben bu hatayı inanılmaz

derecede çok sık yapıyorum. Terzi söküğünü dikemezmiş.

7. Açılış Betiği

Açılış betiği, sistemi kullanmaya hazır halen getiren betiktir. Bizler her zaman /etc/rcS dosyasını açılış betiği olarak kullanmaktayız.

Açılış betiğinin daha önce yazılanlardan bir farkı yoktur. telnet ve ssh sunucusu, telnetd ve dropbear satırlarında başlatılır. Bu satırlar silinirse her iki sunucu da devre dışı kalır.

telnet istenirse inetd yardımı ile başlatılabilir. Biz bağımsız başlattık.

Sistemde bir eternet kartı mevcuttur. Karta tamamen keyfi bir biçimde 10.0.2.15 IP değerini verdik. Her bir emülatör bağımsız bir makine olacağı için her türlü IP verilebilir. Kısıt yoktur.

Ayrıca dev dizini otomatik olarak çekirdek tarafından bağlanır. Eğer initramfs tabanlı kök dosya sistemi kullansaydık rcS içinde dev dizinini kendimiz bağlanmak zorunda kalacaktık. Çekirdeğin /dev dizinini bağlaması için gerekli seçenekler çekirdek derlemesine girilerek incelenebilir.

8. Emülatör ile Test

Artık elimizde çekirdek ve kök dosya sistemi mevcuttur. Çekirdeği ve u-boot gibi bir boot yükleyicisini doğrudan disk.img içine atıp kullanabiliriz. Ya da ağ üzerinden boot yapabiliriz. Ya da daha pek çok teknik ile boot işlemini yapabiliriz. Fakat işleri çok karıştıracağı için şimdilik boot yükleyicisi kullanmayacağız. Bundan dolayı çekirdek açıkça emulatöre tanımlanacaktır. Emülatör de “çekirdek nerede” diye kasmayacak ve boot yükleyicisine gerek kalmayacaktır. Örnek sistem aşağıdaki gibi test edilebilir. root şifresi root’tur.

```
$ qemu-system-arm -M versatilepb
                  -kernel zImage
                  -hda disk.img
                  -append "root=/dev/sda1 rw"
                  -net nic
                  -net user
                  -redir tcp:1234:10.0.2.15:23
                  -redir tcp:1235:10.0.2.15:22
```

Bütün komut tek satırda yazılmalıdır. Kolay okunması için alt alta yazılmıştır.

-M seçeneği ile ARM bordun markası verilir. -M, machine anlamındadır.

-kernel ile derlediğimiz çekirdek verilir. Eğer bu çekirdek disk.img içinde olsaydı, çekirdeğin bulunabilmesi için bir boot yükleyicisine gerek olacaktı. Ama açıkça -kernel seçeneği ile verdiğimizden, boot yükleyicisine gerek olmayacaktır.

-hda seçeneği ile sanal diskimizin adı verilmektedir. Eğer açılışta dosya sistemi ile ilgili bir uyarı gelirse, bilin ki umount veya losetup -d yapılmamıştır.

-append ile çekirdeğe parametre geçirilir. Eğer boot yükleyicisi kullanmış olsaydık bu satıra gerek olmayacaktı. Çünkü gerekli parametreler, örneğin u-boot içinden çekirdeğe geçirilecekti. Ama emülatörün kendisi de açıkça çekirdeğe parametre aktarabilmektedir.

append seçeneğinde bulunan root=/dev/sda1 tanımı ile kök dosya sisteminin, 1. bölümde oturduğu söylenir. Varsayılan değer olarak bütün diskler read/only bağlanır. rw seçeneği ile kök dosya sistemi read/write modunda bağlanır. Gerçek disklerde oturan kök dosya sistemleri asla read/write bağlanmamalıdır.

Eğer emülatör evsahibi sistemden ve ağ üzerinden ulaşmak istemiyorsak sonraki hiç bir seçeneğe gerek yoktur. Fakat bizler emülatördeki sisteme ağ üzerinden, dışarıdan erişeceğiz. Bunun için emülatörde mutlaka en az 2 adet -net seçeneği kullanılmalıdır.

Gerçek bir bilgisayar sisteminin ağa erişebilmesi için, örneğin eternet kartı gibi bir karta ihtiyacı vardır. Bu tür kartlara “network interface card” veya kısaca “nic” denir.

-net nic tanımı, zorlama bir benzetme ile, “bir eternet kartını alıp sanal cihaza takmak” gibi düşünülebilir. Yani bu tanımı yaptığınız anda sanal makineye eternet kartı takmış oluruz.

qemu sistemi çok az eternet kartına destek vermektedir. Destek verilen eternet kartının özellikleri çekirdek derlemesi sırasında incelenebilir. Bu konuda fazla bir seçenek yoktur. compat_monitor0 ekranından “(qemu) info network” girişi ile ilgili kart ve sanal ağ hakkında bilgi alınabilir. Sadece info girişi yapılırsa, help ekranı gelir.

Bu emülatör evsahibi makine ile nasıl haberleşecektir? qemu sisteminin içinde kendine has çok ilginç bir sanal ağ desteği mevcuttur. qemu sistemi dış dünya ile “user mode network stack” denilen bir kavram ile haberleşebilir. Bu tür haberleşme desteği için -net user girilir. Dış dünya ile başka bir haberleşme yöntemi daha vardır. Buna başka bir yazıda değineceğiz.

Bu tür haberleşme şeklinde qemu sistemine localhost yardımı ile erişilir. -redir seçeneği “redirection” demektir.

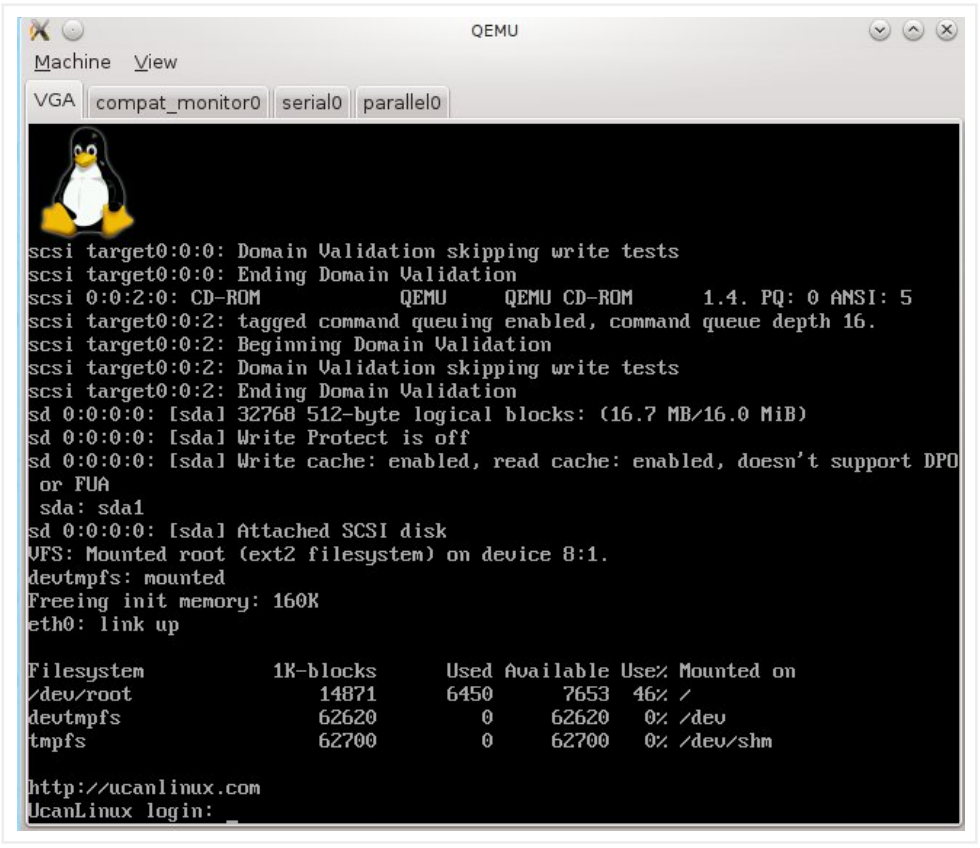
-redir tcp:1234:10.0.2.15:23 seçeneği ile localhost tarafından 1234 numaralı porta

gelen tcp paketleri, emülatör içinde 10.0.2.15 numaralı makinede 23 numaralı porta yönlendirilecektir. 23 numaralı port ise telnet sunucusuna aittir.

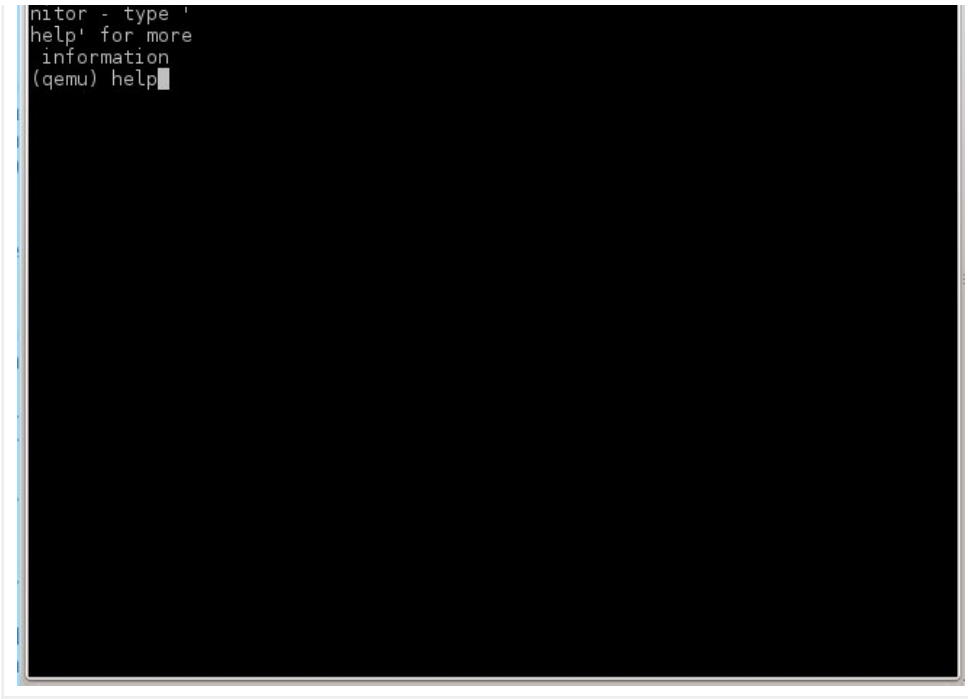
Bu sistemin çalışma mantığı çok basittir. -redir tcp:1234:…. girişi sayesinde, emülatör localhost üzerinden 1234 numaralı tcp portunu dinlemeye başlar ve gelen bütün paketleri yakalar. -redir …:10.0.2.15:23 sayesinde de, yakaladığı paketleri kendi içinde 10.0.2.15 numaralı makinenin 23 numaraları protuna aktarır. Böylece evsahibi makineden misafir makineye telnet bağlantısı kurulmuş olur.

Benzer şeklinde misafir makine ile ssh bağlantısı kurmak için -redir tcp:1235:10.0.2.15:22 seçeneği girilir. ssh portunun numarası 22'dir. Dışarıdan ulaşılması istenilen bütün port yönlendirmeleri -redir ile istenildiği kadar yazılabilir. 1234 veya 1235 gibi verilen port numaraları keyfidir. 1024'ten büyük olması tavsiye edilir.

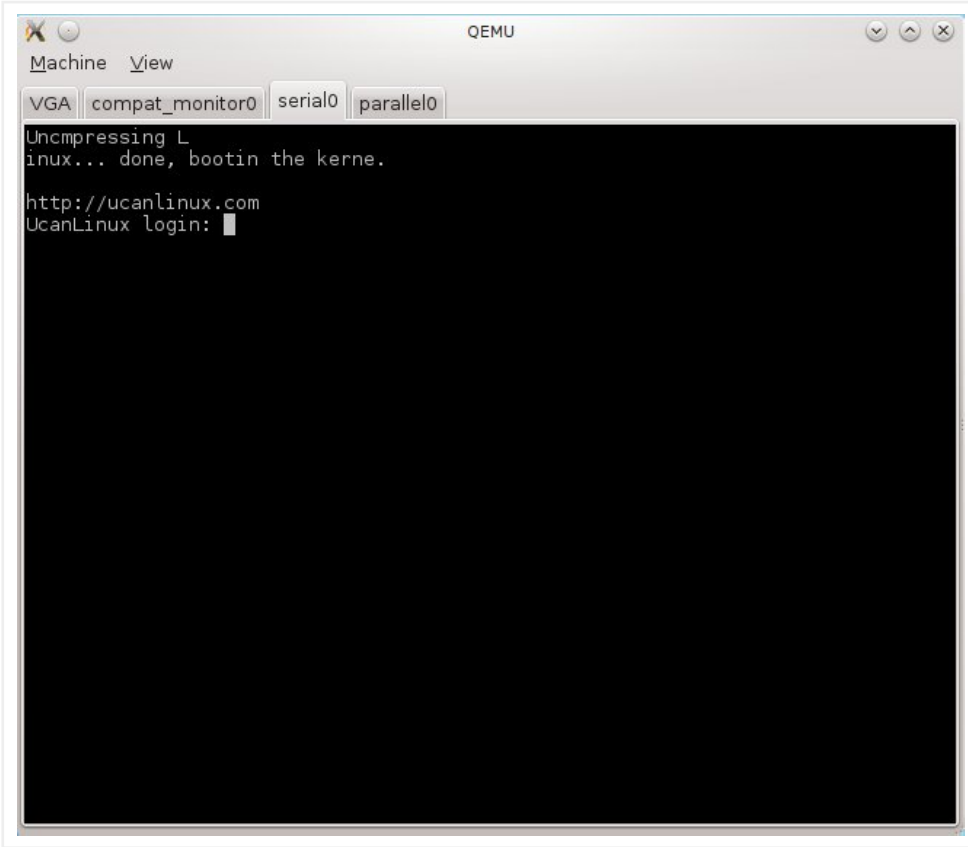
Emülatöre ait bazı ekran çıkışları aşağıda verilmiştir.



— login ekranı. root/root ile giriş yapılabilir.



— monitör ekranı. help veya info ile başlanabilir.



— seri konsol ekranı. root/root ile girilebilir.

Yukarıda 3 ekran çıktısı verilmiştir. Birinci ekran çıktısı esas login ekranımızdır.

root/root ile giriş yapılabilir. İkinci ekran montitör ekranıdır ve insanın moralini bozacak kadar çok seçeneğe sahiptir. help veya info ile komutları ile seçenekler öğrenilebilir. info network girişi ile nic ve ağ hakkında bilgi alınabilir.

Son ekran seri konsol ekranıdır. Sistemde hem VGA hem de seri konsol vardır. İkisi de kullanılabilir. Fakat seri konsol pek kararlı değildir.

Sistemde 2 adet VGA konsol tanımlıdır. Ana menüden "view –> grab input" tıklanırsa, input girişleri, yani klavye ve fare emülatöre geçer. Bu durumda Alt+F2'ye basılırsa ikinci VGA konsola geçilir. VGA konsol sayıları /etc/inittab içinde aşağıdaki satırlar ile aktif hale getirilir.

```
::respawn:/sbin/getty 115200 tty1
::respawn:/sbin/getty 115200 tty2
```

Bu satırlar tamamen silinirse VGA konsol açılmaz. Satır sayısı kadar VGA konsol açılabilir. Masaüstü sistemlerde genelde 6 adettir. AltF1..AltF6 arasındaki tuşlara basılarak konsollarda gezinilebilir. Ctrl+Alt+G'ye basılarak input'lar emülatörden geri alınabillir.

Masaüstünde GUI ortamında çalışırken, emülatörü tam ekran açıp input'u da emülatöre verdikten sonra sistem tıkanırsa, Ctrl + Alt + F1 ile kara ekrana geçip,

```
$ killall qemu-system-arm
```

komutu girilerek emülatör durudurlabilir. Emülatörün tam ekran kullanılması tavsiye edilmez.

Ayrıca sistemde bir de seri konsol özelliği vardır. Son ekranda seri konsolun resmi bulunmaktadır. Seri konsolu aşağıdaki inittab satırı aktif hale getirir.

```
::respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

Bu satır inittab içinden silinirse seri konsol gelmez. Seri konsolun cihaz ismi /dev/ttyAMA0 ile verilmiştir.

Emülatörün seri konsol veya VGA konsolundan kullanılması pek tavsiye edilmez. Çünkü rahat bir kullanım ortamı mevcut değildir. Bunun yerine emülatör ağ ortamından kullanılmalıdır.

9. Emülatöre Erişim

Şu anki örnek sistemimizde telnet ve ssh sunucuları mevcuttur. Emülatöre aşağıdaki gibi telnet veya ssh ile erişilebilir.

```
$ ssh root@127.0.0.1 -p 1235

The authenticity of host '[127.0.0.1]:1235 ([127.0.0.1]:1235)'
     can't be established.
RSA key fingerprint is 54:a2:36:10:56:b5:3b:40:df:68:b9:13:3b:
Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '[127.0.0.1]:1235' (RSA) to the
    list of known hosts.

root@127.0.0.1's password: root
Linux UcanLinux 3.8.2-UcanLinux #8 Tue Mar 5 20:28:16 EET 2013
    armv5tejl GNU/Linux

root@UcanLinux:~ # df

Filesystem 1K-blocks Used Available Use% Mounted on
/dev/root  14871 6451 7652 46% /
devtmpfs   62620    0 62620 0% /dev
tmpfs      62700    0 62700 0% /dev/shm

root@UcanLinux:~ # exit
Connection to 127.0.0.1 closed.

$ telnet -l root 127.0.0.1 1234

Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.

UcanLinux login: root
Password: root

Linux UcanLinux 3.8.2-UcanLinux #8 Tue Mar 5 20:28:16 EET 2013
     armv5tejl GNU/Linux

root@UcanLinux:~ # free
      total used free shared buffers
Mem: 125404 7904 117500 0 136
     -/+ buffers: 7768 117636
     Swap: 0 0 0

root@UcanLinux:~ # exit
Connection closed by foreign host.
```

ssh bağlantısında port numarası -p 1235 şeklinde verilir. telnet bağlanıtısında ise doğrudan IP adresinden sonra 1234 şeklinde yazılmıştır. Bu port numaralarının emülatörü çalıştırırken -redir ile vermiş olduğumuz port numaraları olduğunu tekrar hatırlatalım.

df çıktısından da görüleceği gibi, kök dosya sistemi, sanal diskin sadece yarısı kullanmaktadır. Ayrıca free çıktısından da görüleceği gibi 128MB'lık belleğin 80MB'lık kısmı kullanılmaktadır. qemu sistemi başlatılırken -m seçeneği ile bellek miktarı ayarlanabilir. Varsayılan değer 128MB'tır. -m 256 gibi bir girişle bellek miktarı açıkça verilebilir.

Son olarak, sistemi kapatırken halt veya poweroff kullanalım ki disk.img isimli imajımız bozulmasın. Aynı zamanda emülatör ekranının sol üst köşesinde bulunan "Machine -> reset" veya "Machine -> Power down" ile de sistem yeninden başlatılabilir veya kapatılabilir. Sistem kapatılsa bile emülatör halen ayaktadır. Emülatör yeniden başlatılmadan sistem tekrar tekrar test edilebilir.

Emülatör üzerindeki sistem NFS üzerinden evsahibi sistemin bir dizinini mount edebilir. Bu durumda özellikle uygulama programlarının testi çok ama çok basit olmaktadır. Başka bir yazıda NFS kullanımından bahsedilecektir.

10. Diğer

Çalışma paketi arm_qemu1.tgz

Sürümler

Linux Kernel 3.8.2
Busybox
1.20.2
Toolchain arm-none-linux-gnueabi-gcc (Sourcery CodeBench Lite 2012.03-57) 4.6.3

Güncellemeler

İlk yayın tarihi: 6.Mart.2013

Çekirdek derlemesindeki cd linux satırı yanlış yerdeymiş, düzeltildi. Bazı yazım hataları düzeltildi. 8.Mart.2013

Kullanılan programların sürümleri eklendi: 10.Mart.2013

Kullanım Hakları

— yazı sonu —