

Gömülü Sistemler İçin Linux Dağıtımı Geliştirme

Yazan: **Murat Demirten**

<murat (at) debian.org>

9 Ocak 2007

Özet

Bu belge gömülü sistemler için Linux kullanacak kişilere faydalı olması amacıyla hazırlanmaktadır.

Daha derli toplu bir içerik oluşturduktan sonra sayfaya koymayı düşünüyordum, ancak yoğun iş temposu nedeniyle bir süre yazıma ara vermek zorunda kalacağım. Bu nedenle şu anki haliyle birilerine yardımcı olabilir düşüncesiyle siteye koydum. Her türlü görüş, öneri ve istekleriniz için <murat (at) debian.org> adresine yazabilirsiniz.

İçindekiler

1. Gömülü (Embedded) Sistemler	4
1.1. Gömülü Sistemler ve Linux	4
1.2. Linux İşletim Sistemi	4
1.3. Tek Kartta Bilgisayar (SBC – Single Board Computer)	4
1.3.1. Advantech Half-Biscuit	5
1.3.2. Ampro EnCore	5
1.3.3. JUMPTec Adastra ETX	6
1.3.4. AMC Technologies NETdimm	6
1.3.5. FORTH–SYSTEME DIMM–520	7
1.3.6. Techsol Medallion	7
2. Kurulum Öncesi Hazırlık	9
2.1. Gerekli Bileşenler	9
2.2. Farklı Kurulum Seçenekleri	9
3. Sistemin Boot Edilmesi ve Diskin Hazırlanması	11
3.1. Disketten Açılış	11
3.2. Diskin Hazırlanması	11
4. Temel Ayarlamalar	15
4.1. Dizinlerin Oluşturulması	15
4.2. Dizinlere Erişimlerin Düzenlenmesi	15
4.3. /dev Dizini Altındaki Aygıtların Oluşturulması	15
4.4. Gerekli Kütüphanelerin Taşınması	16
4.5. Busybox Kullanımı	16
4.6. Tinylogin Kullanımı	18
4.7. Açılış Ayarları	19
4.7.1. Çekirdeğin Oluşturulması	19

4.7.2. Init Ayarları	20
4.7.3. LILO Ayarları	22
5. Sistem Kullanıcıları ve Gruplar	24
6. Ağ Ayarları	26
6.1. Inetd Kurulumu	27
6.2. FTP Sunucu Kurulumu	28
6.3. Telnet Sunucu Kurulumu	29
7. Seri Portların Kullanımı	32
7.1. Seri Porttan Dosya Aktarımı	32
7.2. Seri Konsol	33
8. Linux Açılış Süreci	35
9. Initial Ramdisk (Initrd)	39
10. X Window System Kurulumu	40
11. Türkçe Nasıl?	41
11.1. Yerel (Locale) Desteği	41
11.2. Konsolda Türkçe Desteği	42
11.3. X Window System Türkçe Desteği	44

Geçmiş

Versiyon 1.0.0
İlk sürüm

26 Mayıs 2003

murat

Yasal Uyarı

Bu belgeyi, Free Software Foundation tarafından yayınlanmış bulunan GNU Genel Kamu Lisansının 2 ya da daha sonraki sürümünün koşullarına bağlı kalarak kopyalayabilir, dağıtabilir ve/veya değiştirebilirsiniz. Bu lisansın bir kopyasını <http://www.gnu.org/copyleft/gpl.html> adresinde bulabilirsiniz.

Bu belgedeki bilgilerin kullanımından doğacak sorumluluklar ve olası zararlardan belge yazarı sorumlu tutulmaz. Bu belgedeki bilgileri uygulama sorumluluğu uygulayana aittir.

Tüm telif hakları aksi özellikle belirtilmediği sürece sahibine aittir. Belge içinde geçen herhangi bir terim bir ticarî isim yada kuruma itibar kazandırma olarak algılanmamalıdır. Bir ürün ya da markanın kullanılmış olması ona onay verildiği anlamında görülmemelidir.

1. Gömülü (Embedded) Sistemler

Gömülü sistemler, özel amaçlar için kullanılmak üzere tasarlanmış, sadece beklenen görevleri yerine getirmeye programlanmış aygıtlardır. Buradaki amaç belirli bir görevi mümkün olduğu kadar iyi bir şekilde yerine getirmektir.

Gömülü sistemler hiçbir zaman kişisel bilgisayarlar gibi genel amaçlı kullanımlar için uygun değildir. Kişisel bir bilgisayar ile gömülü sistemlerle yapılabilecek işler de gerçekleştirilebilecek olmasına rağmen böyle bir kullanım çoğu durumda kaynakların verimsiz kullanılmasına yol açacaktır.

1.1. Gömülü Sistemler ve Linux

Özellikle donanım teknolojisindeki hızlı gelişmelerle birlikte akıllı cihazlarla sıklıkla karşılaşır olduk. Gömülü aygıtlar (Embedded Devices) olarak adlandırılan bu sistemler yaptıkları işin kapsamına göre özelleştirilmiş işletim sistemleri kullanırlar. Bu türden aygıtlar için geliştirilen özel işletim sistemleri pazarı şimdilerde büyük bir tehlikeyle karşı karşıya. Tehlikenin adı Linux! :)

Gömülü sistemlerde kullanılmak üzere geliştirilmiş pek çok ticari ve ticari olmayan işletim sistemi bulunmakta birlikte, bu belgede sadece Linux işletim sisteminin bu anlamda kullanılabilmesinden bahsedilecektir. Standartları oturmuş, halihazırda on binlerce uygulamayı barındıran komple bir işletim sistemi olarak Linux'un yakında gelecekte bu pazarda çok yoğun bir şekilde kullanılacağını düşünüyoruz.

1.2. Linux İşletim Sistemi

Linux işletim sistemi Linus Torvalds tarafından 1991 yılında ilk duyurulduğunda ancak bir işletim sisteminden beklenen en temel özellikleri yerine getirebiliyor ve sadece Intel-x86 mimarisinde çalışabiliyordu. 2003 yılına gelindiğinde ise Linux çok sayıda mimariye taşınmış ve kaynak kodunun büyüklüğü devasa boyutlara ulaşmıştır. Modüler ve güçlü yapısı, özelleştirilmeye uygunluğu ve kaynak kodunun açık olması gibi özellikleriyle Linux, gömülü işletim sistemleri pazarında da giderek artan bir yoğunlukla kullanılmaya başlanmıştır. Yapılan tahminler ve piyasa araştırmaları sonuçları çok yakın gelecekte gömülü işletim sistemleri pazarına hitap eden ürünlerin büyük oranda Linux tabanlı olacağı yönündedir.

Gömülü sistemler için özel bir işletim sistemi yazma çalışması pek çok zorluğu barındırmaktadır. Yazılan kodların çok sayıda kişi veya grup tarafından test edilmesi gereklidir, ancak özel bir işletim sistemi çekirdeği kullanıldığından test edebilecek kişi sayısı olması gereken sayının çok altında bir değer olacaktır. Ayrıca benzer işlemler için, sıfırdan bir işletim sisteminin geliştirilmesi zaman ve para kaybına yol açmaktadır. Basit bir örnek verecek olursak, eğer sisteminizde TCP/IP yığıtı kullanılacak ise, yeni geliştirilen, çok sayıda kişi tarafından test edilme fırsatı bulamayan bir işletim sistemi, mutlaka ciddi hataları kendi TCP/IP yığıtı içerisinde barındıracaktır. Oysaki aynı işlem Linux ile gerçekleştirilmeye çalışıldığında, TCP/IP yığıtından kaynaklanan bir hata çıkma olasılığının ne kadar düşük olacağı ortadadır.

Gömülü sistemler için Linux kullanımının, sadece alt katmandaki çekirdeğin sağlamlığı açısından değil, üzerinde halihazırda çalışan ve yeni geliştirilmekte olan uygulamaların çokluğu yönünden de ciddi getirileri mevcuttur. Kaynak kodu açık ve bu nedenle özelleştirmelere çok müsait binlerce programa her geçen gün artan bir hızla yenileri de eklenmektedir. Üstelik bu programlar da tüm dünyada çok sayıda kişi tarafından kullanıldığından, karşınıza çıkabilecek hata sayısı az olacaktır ve yeni bir hata bulunsa dahi hatanın düzeltilebilmesi anlamında çoğu durumda bizzat sizin bir şeyler yapmanıza gerek kalmayacak, birileri bunu zaten yapacaktır.

1.3. Tek Kartta Bilgisayar (SBC – Single Board Computer)

Bu belgede gömülü sistemler için Linux dağıtımı geliştiriminden bahsederken aslında kullandığımız donanım bir Tek Kartta Bilgisayar olacaktır. SBC, temel olarak bir masaüstü bilgisayarından çok farklı olmamakla birlikte, üzerinde kullanılan bazı bileşenlerde farklılıklar gösterir. Bu farklılık işlemciden başlar. SBC üzerlerindeki

işlemcilerin çoğunluğu Intel x86 uyumlu olsa da standart bir Intel işlemci değildir. Farklı firmalarca, Intel uyumluluğu göz önünde bulundurularak daha az güç tüketecek, daha az yer kaplayacak ve elbette daha düşük maliyetlerle üretilebilecek şekilde tasarlanmışlardır.

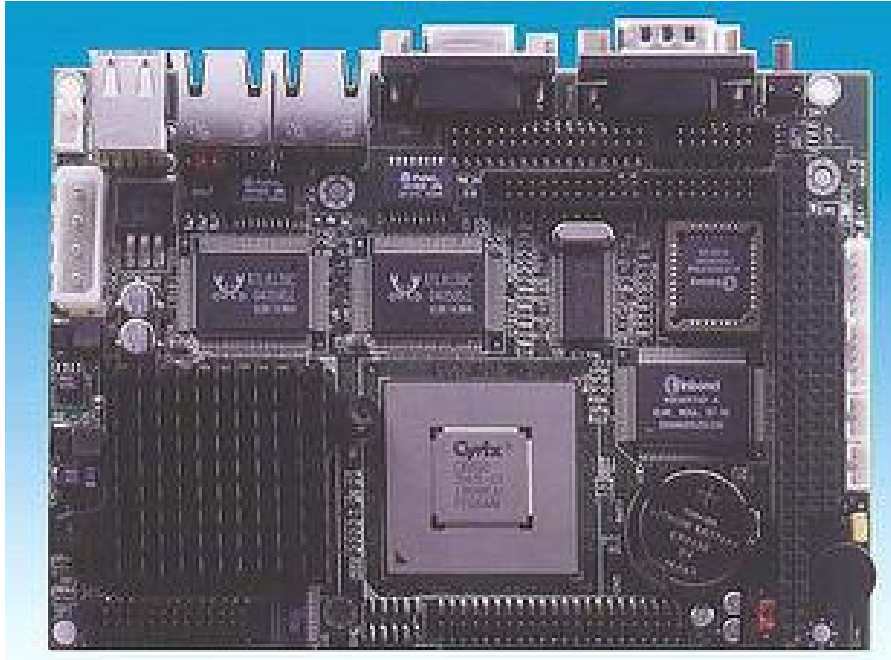
Sistemin diğer bileşenleri için de aynı durum söz konusudur. Bellek, ekran kartı, ağ kartı, ses kartı gibi bileşenler genelde tümleşik olarak (on-board) gelmektedir. Depolama amacıyla Flash Disk teknolojileri kullanılır: Disk On Module, Disk On Chip veya Compact Flash.

Piyasada farklı büyüklükte ve çok farklı amaçlar için üretilmiş SBC'ler mevcuttur. Genel olarak baktığımızda iki tür SBC olduğunu görmekteyiz. Bunlardan birincisi, gerekli tüm bileşenleri içeren (All-in-one) tümleşik yapıdaki SBC'lerdir. Bunlar hemen her zaman tek başlarına kullanılır. İkinci tür ise gömülü sistemler için temel işlevleri içeren ancak daha büyük sistemler veya kartlara takılarak bütünleşik olarak çalışan, diğer türe göre özellikleri daha kısıtlı ve daha kendine özgü SBC'lerdir. Fikir vermesi açısından piyasadaki ürünlerin bazılarını inceleyelim. Buradaki veriler 2002 Ağustos ayı için geçerli olduğundan arada geçen süre zarfındaki olası gelişmeleri de hesaba katınız. Ayrıntılı bilgileri üretici firmaların web adreslerinden elde edebilirsiniz.

1.3.1. Advantech Half-Biscuit

Standart 3,5" büyüklüğünde bir SBC. Üzerinde masaüstü bilgisayarınızda bulunan hemen her bileşen mevcut. National Geode, Transmeta Crusoe ve Cyrix işlemci seçenekleri var. 486-100 ile P3-600 işlemcileri arasındakilerle eşdeğer hızlarda çalışabilen modellerini bulabilirsiniz. Üretici firma adresi: <http://www.advantech.com>

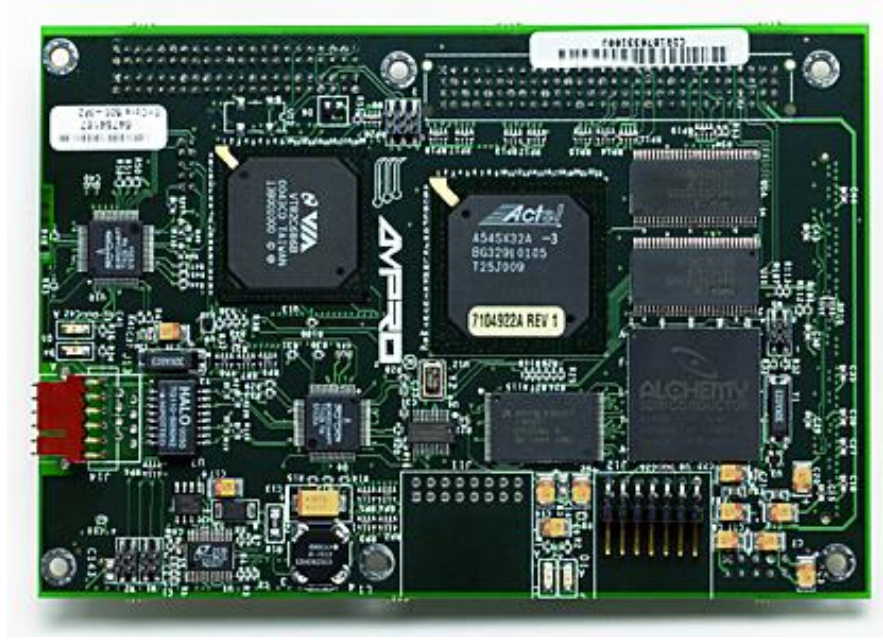
Şekil 1. Advantech's Half-Biscuit



1.3.2. Ampro EnCore

Her birini 3.9" x 5.7" büyüklüğündeki modüller halinde bulup, bunları birleştirebileceğiniz türden bir SBC. İşlemci, Flash depolama birimi, IDE, disket sürücü, ethernet, seri, paralel, ses çıkışları, LCD ve CRT ekran çıkışları modül üzerinde yer almaktadır. İşlemci seçenekleri 486, Pentium, PentiumIII, MIPS ve PowerPC'dir. Üretici firmanın adresi: <http://www.ampro.com>

Şekil 2. Ampro EnCore



1.3.3. JUMPtec Aadastra ETX

3.7" x 4.4" boyutlarında üretilen SBC, modüler yapısı sayesinde farklı sistemlerle bütünleşebiliyor. Her ETX modülü işlemci, bellek, IO birimleri, USB, ses ve ethernet desteğini içermektedir. National Geode, Intel Pentium II ve Intel Pentium III işlemcilerini desteklemektedir. Üretici firma adresi: <http://www.aadastra.com>

Şekil 3. JUMPtec Aadastra ETX

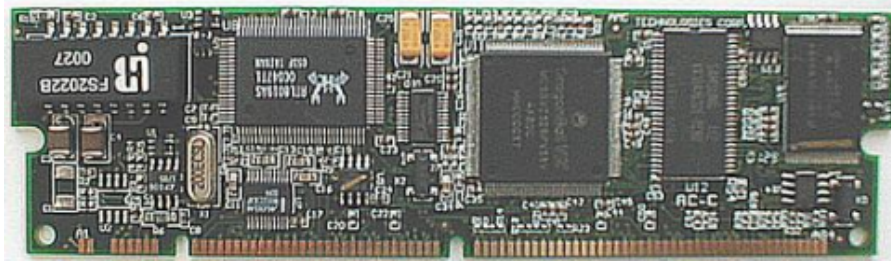


1.3.4. AMC Technologies NETdimm

5.25" x 1.5" boyutlarında dimmPCI normunda olan bu küçük SBC 32 MB SDRAM bellek, 8 MB Flash disk

dahili ethernet, LCD denetleyici ve iki seri portla birlikte gelmektedir. Üretici firmanın adresi: <http://www.amctechcorp.com>

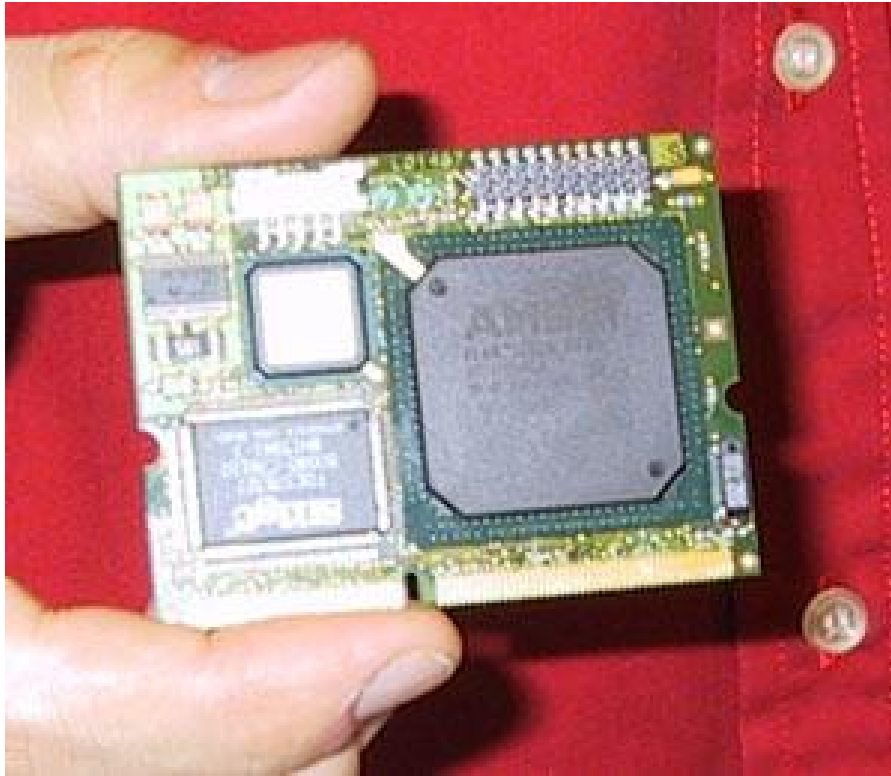
Şekil 4. AMC Technologies NETdimm



1.3.5. FORTH-SYSTEME DIMM-520

Oldukça küçük boyutlardaki (2.7" x 2") bu SBC üzerinde 32 bit 133 Mhz AMD ElanSC520 x86 tabanlı bir işlemci mevcut. Ayrıca 64 MB SDRAM bellek, 16 MB Flash disk, PCI veriyolu, 2 seri 1 paralel çıkış ve dahili 100 Mbit ethernet desteği de bu küçük cihaz üzerine sığdırılmış durumda. Üretici firma adresi: <http://www.fsforth.de>

Şekil 5. FORTH-SYSTEME DIMM-520



1.3.6. Techsol Medallion

En küçük SBC'lerden biri, sadece 10 cm₂ alan kaplıyor. 60MHz ARM-720T RISC işlemci, 32 MB SDRAM, 32 MB Flash disk, 2 seri port, IrDA ve USB desteğiyle birlikte gelmektedir. Üretici firmanın adresi: <http://www.techsol.ca>

Şekil 6. Techsol Medallion



2. Kurulum Öncesi Hazırlık

Bir Tek Kartta Bilgisayar üzerine Linux işletim sisteminin kurulması temelde bir Linux dağıtımı geliştiriminden farksızdır. Yani Red Hat, Debian vb. gibi Linux dağıtımlarının geliştirimi aşamasında yapılanlara benzer işlemlerin yapılması gereklidir. Fakat gömülü bir sistem geliştirirken çok kapsamlı bir kurulum programı hazırlamak gereksizdir, hemen her durumda ana sistem bir defa kurulur ve sonra bire bir kopyalama ve güncellemeler yoluyla yeni sistemler oluşturulur. Bütün mesele sistemi ilk başta kurabilmek ve sisteme hakim olabilmektir.

Linux üzerinde verimli çalışabilmek için sistemin işleyişi hakkında bilgi sahibi olunması gerekir. Sorunların büyük çoğunluğu sistem bileşenlerin seçimi ve kullanımından kaynaklanır. Sistem belirli bir kararlılığa ulaştıktan sonra sürekli sisteme müdahale gerekmeyecektir, ancak o aşamaya gelene kadar mutlaka ciddi çaba sarfedilmelidir.

Sistemin geliştirimine başlamadan önce kullanılan donanımın karakteristik özelliklerinin belirlenmesi ve buna göre sistemin tasarlanması gerekir. Küçük disk ve bellek hacmi bunların başında gelir. Ancak sistemin temel olarak yapacağı işlemlerin özellikleri ve sık kullanılan donanım bileşenlerinin kısıtlamaları birlikte düşünüldüğünde ek tasarım prensipleri de ortaya çıkacaktır.

2.1. Gerekli Bileşenler

Kurulum çalışmasına başlayabilmemiz için ihtiyaç duyduğumuz bileşenler aşağıdaki gibidir:

- Elbette bir Tek Kartta Bilgisayar :)
- Linux ve geliştirme uygulamalarının kurulu olduğu bir kişisel bilgisayar
- SBC'den Linux'u açabilmek amacıyla bir disket veya CD-ROM sürücü
- Şart olmamakla birlikte seri port üzerinden dosya aktarımı yapılacaksa bir "seri iletişim kablosu"

Şekil 7. Kullandığımız Tek Kartta Bilgisayar – Arbor



2.2. Farklı Kurulum Seçenekleri

GNU/Linux sistemlerinde bir işi yapmanın daima birden fazla yolu mevcuttur. Aynı kural bizim için de geçerli. Amacımız elimizdeki SBC üzerinde çalışan kendi GNU/Linux dağıtımımızı oluşturmak, fakat bu işi yaparken nasıl bir yol izlemeliyiz?

Geliştirme yaparken en fazla kolaylık sağlayacak seçenek, SBC üzerindeki diskin, geliştirme amaçlı kullandığınız Linux bilgisayara takılıp, tüm geliştirmenin burada yapılması; disk hazır olduğunda SBC üzerine takılarak sistemin çalıştırılmasıdır. Bu yöntemin bize sağlayacağı en büyük getiri, geliştirme sırasında sürekli disket veya CD-ROM ile iki sistem arasında dosya taşıma işlemine gerek kalmamasıdır.

Örnek olarak, piyasada yaygın olarak kullanılan 40 pinlik IDE bağlantısı ile anakarta takılan bir DiskOnModule kullanacaksanız, aynı anda hem DiskOnModule'u hem de diskinizi bilgisayarınıza takabilmek için, uçlarından biri 40 pin, diğeri standart 44 pin olan IDE kablosu bulmalısınız. Buna benzer bileşenler şanslı iseniz bazen SBC ile birlikte gelmektedir. Diğer durumda kendi başınıza yapmalı veya yapabilen birilerini bulmalısınız.

Buradaki örneklerimizde hep zor olan yöntemleri kullanmak zorunda kaldığınızı varsayarak bu şekilde bir kurulum anlatılacaktır. Örneklerdeki işlem adımlarını kısaltmak sizin elinizdedir. Eğer Linux konusunda henüz yeterli birikime sahip değilseniz temel mantığı anlamak adına bir defa tamamen burada anlatıldığı gibi bir kurulum yapmanız yararınıza olabilir.



Faydalı Belgeler

<http://www.linuxfromscratch.org> veya <http://lfs.geleceklinux.org> (Türkçe) adresinde kendi Linux dağıtımınızı oluşturmanız için yapmanız gereken adımlar anlatılmaktadır. Bu belge hem anlatılan yöntemler hem de geliştirilmek istenen hedef sistem açısından LFS'den çok farklı olmasına rağmen zaman ayırıp LFS belgesini de incelemeniz oldukça faydalı olacaktır.

3. Sistemin Boot Edilmesi ve Diskin Hazırlanması

Bundan sonra "Sistem"den kastımız, geliştirmekte olduğumuz, SBC üzerinde çalışacak olan sistemdir. Gerekli donanım hazır durumdaysa, SBC'yi VGA çıkışına bir monitor bağlayarak geliştirmeye başlayabiliriz. İlk olarak yapmamız gereken SBC üzerindeki diski bölümlendirmek olacaktır. Bunun için çalışan bir Linux sistemine ihtiyaç olduğundan dolayı sistemi herhangi bir dağıtımın kurulum için kullandığı açılış disketleriyle veya varsa kurulum CD'si ile açmalıyız. Ardından bellekte oluşan geçici Linux sisteminde yer alan uygulamalar ile diskimizi bölüm- lenebilir ve biçimlendirebiliriz.

3.1. Disketten Açılış

Eğer SBC sisteminize bir CD-ROM sürücü bağlı ise bu adımı atlayabilirsiniz. Ancak düşük kapasiteli SBC'ler genelde CD-ROM desteği ile gelmezler ya da CD-ROM bağlayabilmek için gerekli olanağı size sunmazlar. Bu durumda yapmanız gereken Linux açılış disketleri oluşturarak sisteminizi açmak olacaktır.

Örnek olarak Debian GNU/Linux dağıtımının kurulum için kullandığı açılış disketlerinin bir kopyasını aşağıdaki adreslerden bilgisayarınıza indirip diskete yazabilirsiniz.

- <http://ftp.debian.org/debian/dists/woody/main/disks-«i386/3.0.23-«2002-«05-«21/images-«1.44/rescue.bin>
- <http://ftp.debian.org/debian/dists/woody/main/disks-«i386/3.0.23-«2002-«05-«21/images-«1.44/root.bin>

Dosyaları bilgisayarınıza indirdikten sonra bilgisayarınıza boş bir disket takıp aşağıdaki komutla indirmiş olduğunuz disket görüntüsünü diskete doğrudan yazabilirsiniz:

```
dd if=rescue.bin of=/dev/fd0
dd if=root.bin of=/dev/fd0
```

Disketler hazırlandıktan sonra SBC *rescue* ve *root* disketleri sırasıyla takılıp sistem açılmalıdır. Açılış işlemi sonrasında, yukarıdaki adreste yer alan disketleri kullandı iseniz Debian GNU/Linux kurulum programı açılacaktır. Bu esnada **Alt - F2** tuşlarına basarak 2. konsol ekranına geçebilirsiniz. Burada temel Linux uygulamalarını kullanabilirsiniz.

3.2. Diskin Hazırlanması

Sistem boot edildikten sonra diskimizi istediğimiz gibi bölümlendirebilir ve bölümleri istediğimiz dosya sistemine göre biçimlendirebiliriz. Burada kullanılan disk teknolojisi nedeniyle sistemde bazı kısıtlamalara gitmemiz gerekebilir. Flash disk teknolojilerine kısaca değinelim.

Flash bellek tabanlı diskler gömülü sistemler için düşük fiyatına karşın sağladığı olanaklar sayesinde her geçen gün daha fazla kullanılır olmuştur. Flash bellekler elektrik ile silinebilir ve tekrar yazılabilirler (Electrically Erasable Read Only Memory, EEPROM). Kendi içerisinde farklı tipleri olsa da günümüzde yaygın olarak NOR Flash bellekler kullanılmaktadır. NOR Flash bellekler 128 kB büyüklüğündeki bloklar halinde üretilir. Dolayısıyla bu tip belleklerde en küçük elemanın aslında 128 kB olduğunu söyleyebiliriz. Diskten 1 bayt dahi silmek isterseniz, bu işlem silmek istediğiniz baytın bulunduğu 128 kB'lık blokun tamamen silinmesini gerektirir. Flash belleklerin ömrünü de işte bu yazma işlemlerin sayısı belirler. Tipik bir flash bellek ömrü blok başına ortalama 100.000 yazma işlemini gerçekleştirebilecek kadardır. İlk başta 100.000 ulaşılması zor bir rakam gibi gözükabilir, ancak 1 bayt için dahi bloklar halinde bu işlemin gerçekleştirileceği hesaba katılacak olunursa aslında bu değere hiç de uzun olmayacak bir zaman diliminde erişilmesi olasılığının bulunduğu anlaşılacaktır.

Kısıtlamaların aşılabilmesi için sürekli yeni yöntemler geliştirilmeye çalışılmaktadır. Örneğin "wear levelling" adı verilen bir teknik ile herhangi bir blokun diskteki diğer bloklardan çok daha önce bu 100.000 değerine ulaşması

önlenebilmektedir. Bu sayede diskte yoğun olarak kullanılan bir blok yüzünden kısa zamanda diskin tamamının kullanılamaz hale gelmesinin önüne geçilebilmektedir.

Şekil 8. Disk On Module



Yazma ömrü konusundaki kısıtlamalar nedeniyle Flash bellek teknolojilerini kullanan diskler üzerinde takas alanı kullanımından kesinlikle kaçınılmalıdır. Gömülü sistemler zaten önceden belirlenmiş amaçlara hizmet edeceklerinden, bellek kullanımı belirli değerler arasında sabit kalacaktır. Dolayısıyla sistem iyi tasarlandığı müddetçe takas alanı kullanımına zaten ihtiyaç olmayacaktır.

Gerçekleştirilecek sistemde eğer yazma işlemleri sayıca yüksek olursa bu durumda salt-okunur bir sistem tasarlanması mantıklı olabilir. Sistem açılışında belleğin bir bölümünün disk bölümü olarak kullanılması, gerçek diskin ise tamamen salt-okunur kipe geçirilmesiyle yazma ömrü ciddi oranda artırılabilir. Ancak bu durumda veri kayıplarını engellemek için ramdisk üzerindeki bilgilerin güvenliğini sağlayıcı önlemler getirilmelidir (Belirli aralıklarla disk üzerindeki sistemi salt-okunur kipten çıkarıp veriler ramdisk'den diske yazıldıktan sonra tekrar salt-okunur kipe geçirmek gibi).

Disk üzerinde kullanılacak dosya sisteminin seçimi de oldukça önemlidir. Çoğu gömülü sistem için ext2 kadar karmaşık bir dosya sistemine ihtiyaç duyulmayacaktır. İhtiyaç olmadığı halde ext2 dosya sistemi kullanıldığı takdirde gereksiz yere dosya sistemi tarafından bir yük oluşturulacaktır. İleride dosya sistemleri konusuna ayrıntılı olarak değineceğiz. Biz geliştireceğimiz gömülü sistem için **minix** dosya sistemini kullanalım. Minix dosya sisteminde bir disk bölümü maksimum 64 MB, dosya adları maksimum 31 karakter olabilir. Kullanılacak dosya sistemi seçilmeden önce bu gibi özelliklerinin araştırılıp, geliştirilecek olan sistemin yapısını etkileyip etkilemeyeceği önceden belirlenmelidir.

Son olarak SBC sistemimizi disketler ile açmış ve **Alt - F2** tuşları ile 2. sanal konsola geçiş yapmıştık. Diskimizin Linux çekirdeği tarafından nasıl tanındığını öğrenelim:

```
# dmesg | grep -i disk
RAM disk driver initialized: 16 RAM disks of 16384K size
hda: 64MB ATA Flash Disk, ATA DISK drive
...
```

Diskimizin `/dev/hda` olarak Linux tarafından tanındığını gördük. Şimdi **fdisk** programı ile diski bölümlendirelim.

```
# fdisk /dev/hda
```

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
P
Partition number (1-4): 1
First cylinder (1-489, default 1): 1
Last cylinder or +size or +sizeM or +sizeK (1-489, default 489): 489

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

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
#
```

Yukarıda örnekte görüldüğü gibi disk üzerinde 64 MB'lık bir Linux disk bölümü yarattık. Debian açılış disketlerinde (ve büyük olasılıkla diğer dağıtımların kullandığı disketlerde de) minix dosya sistemi yaratmak için gereken **mkfs.minix** programı bulunmamaktadır. Dolayısıyla bu programı çalışan bir Linux sisteminden disket ile SBC üzerine taşımamız gerekir. Bunun için bir Linux bilgisayardan gerekli dosya diskete kopyalanır. Ardından disket SBC'ye bağlı disket sürücüyü takılır (Açılış tamamlandıktan sonra, açılış için kullanılan disketlerin sürücüde kalmasına gerek yoktur). Disketi SBC'ye taktıktan sonra,

```
# mount /dev/fd0 /floppy
# ls /floppy
mkfs~1.min
#
```

Dosyayı taşımak için kullandığım disket üzerindeki dosya sistemi `vfat` idi. Ancak SBC'nin açılışında kullandığım sistemde `vfat` desteği yok, bu nedenle disket `msdos` dosya sistemi ile bağlandı. `ls` komutunun çıktısında görülen `mkfs~1.min` bundan kaynaklanmaktadır. Şimdi dosyayı disketten `/tmp` dizini altına olması gerektiği ad ile kopyalayıp disketi çıkartalım:

```
# cp /floppy/mkfs~1.min /tmp/mkfs.minix
# umount /floppy
#
```

Şimdi artık **mkfs.minix** ile hazırlamış olduğumuz disk bölümü üzerinde `minix` dosya sistemi oluşturabiliriz:

```
# /tmp/mkfs.minix /dev/hda1
20864 inodes
62576 blocks
Firstdatazone=665 (665)
Zonesize=1024
Maxsize=268966912

#
```

Şimdi dosya sistemi de oluştuğuna göre diski bağlayabiliriz. Bunun için `/hedef` ismiyle bir dizin açıp disk bölümünü bu dizine bağladıktan sonra, geliştirmelerimizi burada yapalım.

```
# mkdir /hedef
```

```
# mount /dev/hda1 /hedef
# mount
/dev/ram0 on / type ext2 (rw)
/proc on /proc type proc (rw)
/dev/hda1 on /hedef type minix (rw)
#
```

Artık diskimiz üzerindeki bölüm ve dosya sistemi /hedef dizini altında kullanımımız için hazır bulunuyor.

4. Temel Ayarlamalar

Her Linux sisteminde mutlaka bulunması gereken bazı dizinler vardır (`/etc`, `/sbin` vb.) ve bu dizin yapılarının oluşturulması gereklidir. Aynı şekilde bazı dizinlerin erişim hakları diğerlerinden farklılık gösterebilir (`/tmp` gibi), bazı dizinlerin başka yerlere sembolik olarak bağlanması gerekebilir. Sistemin çalışabilmesi için `/etc` dizini altında birtakım ayarlamaların yapılması gereklidir.

4.1. Dizinlerin Oluşturulması

Aşağıdaki komutla temel dizinleri oluşturabiliriz. Unutulan bir dizin olursa herhangi bir anda oluşturulabilir.

```
# cd /hedef
# mkdir -p bin boot etc floppy home root lib proc sbin tmp usr var \
usr/bin usr/lib usr/sbin var/lib var/lock var/log var/run var/tmp
#
```

4.2. Dizinlere Erişimlerin Düzenlenmesi

Linux sistemlerde `/tmp` dizini geçici işler için sistemdeki tüm kullanıcılar tarafından kullanılabilir. Geliştirmekte olduğunuz sisteme sadece bir kişi giriş yapacak olacak olsa dahi bu ayarlamalar yapılmalıdır. Çünkü sisteminizdeki bazı servisler kendilerine özel kullanıcı haklarıyla çalışacak ve `/tmp` dizini altına yazılabilir olduğunu varsayacaklardır. Bu nedenle `/tmp` dizini mutlaka kullanıcılar tarafından yazılabilir bir alan olmalı ve yapışkan biti ile verilmelidir.

```
# chmod -R 1777 tmp
# ls -lad tmp
drwxrwxrwt      2 root  root           64 May 25 19:53 tmp
#
```

Geliştirdiğimiz sistem için çok gerekli olmamakla birlikte `/root` dizinine sadece `root` kullanıcısının erişebilmesi için erişim hakkını düzenleyelim:

```
# chmod 700 root
#
```

4.3. `/dev` Dizini Altındaki Aygıtların Oluşturulması

Linux sistemlerde aygıtlara erişmek için kullanılan özel dosyalar `/dev` dizini altında bulunur. Gömülü bir sistemde kullanılacak olan aygıtlar bellidir ve çok ender olarak değişir. Çok geçerli bir neden olmadıkça gömülü sistemlerde SCSI aygıtlar kullanmazsınız veya sisteminize yüzlerce kişi sisteme giriş yapmaz. Bu nedenle normal bir Linux sistemindeki `/dev` dizini altında yer alan aygıt dosyalarının çoğu bizim için geçerli değildir. Burada önerebileceğim yol, Linux'a hakimseniz, sadece kullanacak olduğunuz aygıtlarla ilgili dosyaları `mknode` ile ana ve alt düğüm numaraları, karakter veya blok veri transferi özelliklerine göre oluşturmanızdır. Ancak bu yol epey zahmetli olacağından hazır bir `/dev` dizininden yararlanmak da mantıklı olabilir. Sistemimizi açtığımız disketler de bellekte küçük bir Linux sistemi oluşturduğundan buradaki `/dev` dizinini aynen kopyalayabiliriz:

```
# cp -a /dev .
#
```

Varolan bir sistemdeki `/dev` dizini kopyalandıktan sonra gereksiz gördüğünüz aygıt dosyalarını sonradan da silebilirsiniz.

4.4. Gerekli Kütüphanelerin Taşınması

Geliştirdiğimiz Linux sistemimizde kullanacak olduğumuz uygulamalara göre gerekli kütüphanelerin sisteme kurulması gereklidir. Bunun için çalışan bir Linux sistemden ilgili dosyalar, uygun yerlere kopyalanmalıdır. Dosyaları taşıyabilmek için bu aşamada tek çözüm disket kullanmak olacaktır.

Aşağıdaki temel paylaşımlı kütüphaneler hemen her sistem için gerekli olduğundan, bunları çalışan Linux sisteminizin `/lib` dizininden bir disket üzerine kopyalamalısınız:

- `libc.so.6`
- `ld-linux.so.2`
- `libdl.so.2`
- `libnsl.so.1`
- `libm.so.6`
- `libcrypt.so.1`

Daha sonra disketi SBC üzerine takıp `mount /dev/fd0 /floppy` komutu ile sisteme bağlayın. Disketin üzerindeki dosya sistemi FAT ise dosyalarda 8.3 karakter problemi yaşayabilirsiniz. Bu durumda dosyaları yeniden adlandırmakla uğraşmak istemiyorsanız `mkfs.minix /dev/fd0` veya `mkfs.ext2 /dev/fd0` komutuyla çalışan Linux sisteminizde disketi biçimlendirip dosya taşıma işlemini yeniden yapınız. Tüm bu adımları geçtiğinizi varsayarsak disket SBC'ye takılı ve bağlı durumda iken

```
# cp /floppy/* /hedef/lib
```

komutuyla dosyaları olması gereken yere kopyalayın.

Normalde kütüphane dosyaları doğrudan çalıştırılmadığı için erişim kipinin sadece okumaya izin vermesi yeterlidir. Ancak `libc.so.6` ve `ld-linux.so.2` kütüphanelerine çalıştırılabilir dosya erişimi de vermeniz gereklidir:

```
# chmod 755 /hedef/lib/libc.so.6
# chmod 755 /hedef/lib/ld-linux.so.2
```

Artık sistemimiz için gerekli olan temel kütüphaneler hazır. Şimdi gene çok gerekli olan uygulamaları kuralım ama nasıl?

4.5. Busybox Kullanımı

Busybox projesi Debian GNU/Linux dağıtımı kurulum programının çalışabilmesi için oluşturulan 2 disketlik Linux sisteminde kullanılmak üzere geliştirilmiştir. Burada amaç, kurulum için gerekli ufak fakat çok sık kullanılan uygulamaları (`cp`, `mv`, `tar`, `ls`, `sh`, `find`, `init` vb.) yeniden yazarak tek bir uygulama üretmek ve böylece yerden kazanmaktır. Busybox bu yer kazancını, uygulamaların sık kullanılmayan özelliklerini desteklemeyen sürümlerinin yeniden yazılmasıyla gerçekleştiriyordu. Örnek olarak `find` komutu normalde onlarca parametre ile çalıştırılabilirken Busybox içerisinden çıkan `find`, sadece en sık kullanılan beş parametresini desteklemektedir. Bu koşul gözönünde bulundurularak en temel uygulamalar yeniden yazılmış ve sonuçta 300–400 kB büyüklüğündeki bir dosya ile, her biri ayrı ayrı kullanıldığında normalde 4–5 MB toplam yer kaplayacak olan uygulamalar ile aynı işlevsellik büyük ölçüde sağlanabilmiştir.

Sistemimizde **busybox** kullanımı disk alanından kazanç sağlayacaktır. Busybox'ın bir diğer önemli özelliği modüller yapısı sayesinde istediğimiz parçaları ekleyip çıkarabilmemizdir. Busybox'ı derlemeden önce `Config.h` dosyasını bir metin düzenleyici ile istediğiniz gibi düzenleyebilirsiniz. Bu dosyanın biçimi aşağıdaki gibidir:

```
//#define BB_LOGNAME
//#define BB_LOSETUP
```

```
#define BB_LS
#define BB_LSMOD
//#define BB_MAKEDEV
//#define BB_MD5SUM
#define BB_MKDIR
//#define BB_MKFIFO
//#define BB_MKFS_MINIX
#define BB_MKNOD
#define BB_MKSWAP
//#define BB_MKTEMP
```

Standart bir C başlık dosyası olan `Config.h` içerisinde istediğiniz özellikleri yukarıdaki örnekte olduğu gibi ekleyebilir ve çıkarabilirsiniz. Böylelikle sadece size gerekli olan bileşenleri içeren bir **busybox** derleyebilirsiniz.

busybox derlendikten sonra sistemde sadece **busybox** programının olması, **ls**, **cp** vb. gibi desteklenen uygulamaların ise **busybox**'a sembolik bağ olarak bağlı olması yeterlidir. **busybox** kendi başına çalıştırıldığında ise içerdiği uygulamaları listeler:

```
laptop:/usr/src/busybox-0.60.5$ ./busybox
BusyBox v0.60.5 (2003.04.22-14:09+0000) multi-call binary

Usage: busybox [function] [arguments]...
or: [function] [arguments]...

BusyBox is a multi-call binary that combines many common Unix
utilities into a single executable. Most people will create a
link to busybox for each function they wish to use, and BusyBox
will act like whatever it was invoked as.

Currently defined functions:
[, ash, basename, busybox, cat, chgrp, chmod, chown, chroot, chvt,
clear, cp, cut, date, dd, df, dirname, dmesg, du, echo, env, false,
find, free, grep, gunzip, gzip, halt, head, id, init, kill, killall,
klogd, linuxrc, ln, logger, ls, lsmmod, mkdir, mknod, mkswap, modprobe,
more, mount, mv, pidof, poweroff, ps, pwd, reboot, reset, rm,
rmdir, sed, sh, sleep, sort, swapoff, swapon, sync, syslogd, tail,
tar, test, touch, true, tty, umount, uname, uniq, uptime, wc,
which, whoami, xargs, yes, zcat
```

busybox kaynak kodlarını <http://www.busybox.net/downloads/> adresinden indirebilirsiniz. tar.gz arşivi indirip açtıktan sonra bu dizine geçip, `Config.h` dosyasını düzenlemelisiniz. Gene aynı dizindeki `Makefile` dosyasını inceleyerek programın derlenme aşamasındaki seçenekleri değiştirebilirsiniz. Bu adımları tamamladıktan sonra **make** ve **make install** komutlarıyla programın derlenmesini ve kurulumu hazır hale gelmesini sağlıyoruz. **make** komutunun çalışması bittiğinde elimizde bir **busybox** uygulaması oluyor. **make install** ise bulunduğunuz dizinde `_install` adında bir alt dizin açıyor ve **busybox** uygulamasını olması gereken sembolik bağlar ile birlikte buraya kopyalıyor:

```
laptop:/usr/src/busybox-0.60.5$ ls -l _install/usr/sbin
total 0
lrwxrwxrwx 1 demirten demirten 17 May 30 00:05 chroot -> ../../bin/busybox
lrwxrwxrwx 1 demirten demirten 17 May 30 00:05 fbset -> ../../bin/busybox
```

Burada `bin`, `sbin`, `usr/bin` ve `usr/sbin` dizinleri oluşturulmuş, `bin` dizini altına **busybox**'ın kendisi atılmış, ilgili tüm sembolik bağlar oluşturulmuştur. Bu dizini `tgz` arşivi haline getirerek diskette SBC üzerine taşıyıp orada açalım:

```
laptop:/usr/src/busybox-0.60.5$ cd _install
laptop:/usr/src/busybox-0.60.5/_install$ tar cvfz /tmp/busy-bin.tgz .
...
```

Oluşan `busy-bin.tgz` arşivinin boyu 180 kB gibi çok küçük bir değer olduğu için disket ile SBC üzerine rahatlıkla taşınabilir. Daha önceden yaptığımız gibi disketi sisteme bağlayıp `busy-bin.tgz` dosyasını kopyalıyoruz. Ardından bu disketi SBC sistemine takıp bağladıktan sonra aşağıdaki komutlarla arşivi açıyoruz:

```
# cd /hedef
# cp /floppy/busy-bin.tgz .
# tar zxvf busy-bin.tgz
...
```

Ardık SBC sistemimiz üzerinde onlarca kullanıma hazır temel uygulama var.

4.6. Tinylogin Kullanımı

busybox kurulumu sonrasında ihtiyaç duyduğumuz temel uygulamaların neredeyse tamamı SBC sistemimize yüklenmiş oldu. Ancak **busybox** içerisinden sisteme girişler, erişim denetimi, kullanıcılar üzerinde işlemler vb. yapabilecek uygulamalar mevcut değildir. Bu uygulamalar **busybox** projesinin tamamlayıcısı niteliğinde olan **tinylogin** içerisinden çıkmaktadır. **busybox** için söylediklerimizin hepsi **tinylogin** için de geçerlidir ve aradaki tek fark, her iki uygulamanın içerdiği işlevlerdedir. Yakın bir gelecekte **tinylogin**'in tamamen **busybox** içerisine dahil edildiğini görürseniz şaşırmayın.

tinylogin uygulamasını hazırlamak için önce <http://www.busybox.net/downloads/> adresinden kaynak kod arşivini indiriyoruz. Arşivi açtıktan sonra gene **busybox**'ta olduğu gibi **tinylogin** içerisine dahil etmek istediğimiz özellikleri belirleyebiliyoruz:

```
#define CONFIG_PASSWD
//#define CONFIG_SU
// Enable using shadow passwords
//#define CONFIG_FEATURE_SHADOWPASSWDS
// Enable using sha passwords
//#define CONFIG_FEATURE_SHA1_PASSWORDS
```

Burada istediğiniz özellikleri seçtikten sonra dosyayı kaydedin **make && make install** komutunu verin. **busybox**'da olduğu gibi gene `_install` adında bir alt dizin oluşacak, burada **tinylogin** uygulaması ve olması gereken sembolik bağlar oluşturulacaktır. Bu dizin yapısını SBC üzerine taşımak için izlediğimiz yol öncekiyle aynıdır:

```
laptop:/usr/src/tinylogin-1.4$ cd _install
laptop:/usr/src/tinylogin-1.4$ tar cvfz /tmp/tiny-bin.tgz .
...
```

Ardından oluşan `tiny-bin.tgz` dosyasını disket ile SBC sistemi üzerine taşıyıp disketi bağladıktan sonra:

```
# cd /hedef
# cp /floppy/tiny-bin.tgz .
# tar zxvf tiny-bin.tgz
...
```

Ardık sistemimize **tinylogin** uygulaması kurulmuş oldu. Derlediğimiz **tinylogin** içerisinden nelere destek verildiğini görelim:

```
# /hedef/bin/tinylogin
Tinylogin v1.4 (2003.05.29-21:22+0000) multi-call binary

Usage: tinylogin [function] [arguments]...
       or: [function] [arguments]...

TinyLogin is a multi-call binary that combines several tiny Unix
utilities for handling logins, user authentication, changing passwords,
and otherwise maintaining users and groups on an embedded system. Most
people will create a link to TinyLogin for each function they wish to
use, and TinyLogin will act like whatever it was invoked as.

Currently defined functions:
  addgroup, adduser, delgroup, deluser, getty, passwd, tinylogin,
  vlock
```

busybox ve **tinylogin** kurulumu tamamlandığında elimizde çalışabilir bir sistem olmaktadır. Şimdi yapmamız gereken bu sistemin kendi başına açılabilmesini sağlamak amacıyla gerekli düzenlemelerdir.

4.7. Açılış Ayarları

Sistemi SBC üzerindeki sistemden başlatabilmek için yapmamız gereken bir takım ayarlamalar mevcut. Bunlar, SBC'ye özgün bir çekirdek oluşturma, açılışta **init** tarafından çalıştırılacak ilk programı hazırlama, sisteme girişleri düzenleme ve çekirdeği yükleyebilmek için bir önyükleyici ayarlamak olacaktır.

4.7.1. Çekirdeğin Oluşturulması

Sistemimizi disketlerin yardımı olmadan açabilmek için öncelikle sistemi düzgün bir şekilde başlatabilecek, istediğimiz özelliklere sahip bir çekirdeğin olması gereklidir. SBC sistemi üzerinde çalışacak olan Linux çekirdeğini derlemek için bir Linux makine kullanacağız. Derleme işlemini yapacağınız zamandaki kararlı son sürüm Linux çekirdeğini <http://www.kernel.org> adresinden indirip, SBC üzerindeki donanımlar ve SBC üzerinde kullanmak istediğiniz yapılar için gerekli olan çekirdek desteklerine sahip olacak şekilde bir derleme yapmalısınız. Linux çekirdeğinin derlenmesi hakkında örneğin [Çekirdeğe Yama Uygulanması ve Çekirdeğin Derlenmesi NASIL¹](#) belgesinden yararlanabilirsiniz.

Bu aşamada SBC üzerinde çalışacak olan Linux çekirdeğini derlediğinizi ve bir disket yardımı ile SBC üzerindeki /hedef/boot dizini altına linux adıyla kopyaladığınızı varsayıyoruz.



Bilgi

Çekirdeğin derlenmesi sırasında en sık yapılan hata, SBC sisteminizde kullandığınızdan farklı bir işlemci ailesinin seçilmesi ve bu yüzden sistemin yüklenememesidir. SBC üzerindeki işlemciler aynı hızlardaki ağabeylerinin birebir aynısı değillerdir. 133 MHz hızındaki bir işlemci için 486 kipinde çekirdek derlemeniz gerekebilir, 586 ve üstü çalışmayabilir. Böyle bir durumla karşılaşırsanız çekirdeği yeniden derleyip, SBC'yi daha önce kullandığımız iki disket ile açıp kök bölümünü gene /hedef dizini altına bağladıktan sonra, yeni çekirdeği disket ile /hedef/boot dizini altına taşımalı ve ardından burada anlatılan **lilo** ayarlamalarının ilgili kısımlarını yeniden yapmalısınız.



Uyarı

Buradaki /hedef dizini, SBC'yi açarken kullandığımız iki disketin bellekte oluşturduğu geçici Linux sistemi altında yer almaktadır. DiskOnModule üzerinde /hedef adında bir dizin yoktur.

4.7.2. Init Ayarları

Linux çekirdeği bir önyükleyici programı tarafından açılış esnasında belleğe yüklenip çalışmaya başlar. Çekirdek donanımları tarar, sisteme yerleşir ve tüm yapması gereken işlemleri bitirdiğinde artık sistemi bize devretmek ister. İşte bu noktada çekirdeğin yaptığı iş `/sbin/init` programını çalıştırmaktır.

Çekirdek tarafından ilk olarak çalıştırılan `init` programının bu nedenle süreç numarası (PID) daima 1'dir. Geri kalan tüm süreçler için `init` ana süreç (parent) konumundadır.

Peki `init` neler yapar ve biz nerede devreye gireceğiz? `init` programı, çalıştığında öncelikle `/etc/inittab` ayar dosyasını okur. Buradan aldığı verilerle sistemin hangi çalışma seviyesinde (runlevel) açılacağına karar verir ve ilgili programı çalıştırır. Ayrıca sanal konsollardan sisteme girişleri sağlayabilmek için `getty` programını bu konsollarla ilişkilendirir.

`init` ile ilgili dikkat etmemiz gereken nokta, `busyBox` içerisinden çıkan yeniden yazılmış `init` sürümünün çalışma seviyelerini (runlevels) desteklemediğidir. Bu nedenle normal Linux bilgisayarınızdaki `/etc/inittab` dosyasını örnek olarak kullanamazsınız. Bunun yerine `busyBox` içerisindeki `init`'in kullandığı dosya biçimini kullanmalısınız.



Bilgi

Çoğu durumda gerekli olmamakla birlikte herhangi bir nedenden ötürü çalışma seviyelerine ihtiyaç duyan bir sistem geliştiriyorsanız `busyBox` ile gelen `init`'i değil, `sysvinit`'i kullanmalısınız.



Bilgi

Eğer bir `inittab` dosyası ayarlamaz iseniz `busyBox` aşağıdaki gibi bir `inittab` içeriğini girmiş olduğunuzu varsayarak, bu içeriğe göre çalışır:

```
::sysinit:/etc/init.d/rcS
::askfirst:/bin/sh
::ctrlaltdel:/sbin/reboot
::shutdown:/sbin/swapoff -a
::shutdown:/bin/umount -a -r
::restart:/sbin/init
```

Ayrıca `/dev/console` aygıtının erişilebilir olduğunu görürse aşağıdaki satırları da ekleyecektir:

```
tty2::askfirst:/bin/sh
tty3::askfirst:/bin/sh
tty4::askfirst:/bin/sh
```

Şimdi örnek bir `inittab` dosyası hazırlayalım. İlk iki konsoldan doğrudan sisteme giriş yapılabilsin, üçüncü konsoldan ise `/etc/passwd` dosyasından kimlik doğrulama yapılarak sisteme girilebilsin. Bunun için aşağıdaki gibi bir `inittab` dosyasını `vi /hedef/etc/inittab` komutu ile oluşturalım:

Örnek 1. /etc/inittab içeriği

```
::sysinit:/etc/init.d/rcS
::askfirst:/bin/sh
::ctrlaltdel:/sbin/reboot
::shutdown:/bin/umount -a -r
::restart:/sbin/init

#tty1::askfirst:/bin/sh
tty2::askfirst:/bin/sh
```

```
tty3::respawn:/sbin/getty 38400 tty3
```

Dosyayı bu şekilde oluşturup kaydettikten sonra şimdi dosyadaki satırları açıklayalım. İlk satırda **init** programının sistemin açılış işlemini devredeceği program belirtmektedir. İşte burası bizim sistem açılışında doğrudan müdahale etmemiz gereken yerdir. Örneğimizde bu değer `/etc/init.d/rcS`'dir. **init** kendi işini bitirince bu programı çalıştıracaktır. Bu programı birazdan hazırlayacağız.

İkinci satırda **askfirst** özel belirteci `/bin/sh` ile ilişkilendirilmiştir. **askfirst** ile ilişkilendirilmiş bir sanal konsola geçildiğinde <ENTER> tuşuna basmak suretiyle kabuğa düşülür (`/bin/sh`). Birinci konsol `tty1` için **askfirst** elemanı öntanımlı olarak ayarlandığı için dosyanın 7. satırındaki ayar, önüne `#` getirilerek iptal edilmiştir. Etkinleştirseniz de bir problem olmaz, sadece birinci konsolda 2 defa giriş mesajını alırsınız:

```
Please press Enter to activate this console.  
Please press Enter to activate this console.
```

3, 4 ve 5. satırda yapılan işlemler isimlerinden de anlaşılacağı üzere sistemdeki belirli olaylar karşısında çalıştırılacak olan uygulamaların seçilmesini sağlamaktadır. 8. satırda 2. sanal konsol da **askfirst** ile ilişkilendirilmekte, 9. satırda ise 3. sanal konsol **getty** programıyla ilişkilendirilmekte ve *respawn* yapılarak süreç öldüğü zaman **getty**'nin yeniden çalıştırılması gerektiği belirtilmektedir.

Şimdi `/etc/init.d/rcS` dosyamızı oluşturalım. Basit bir kabuk programı ile açılışta yapılmasını istediğimiz işlemleri burada gerçekleştireceğiz. Öncelikle sistemde `init.d` dizinini oluşturalım.

```
# mkdir /hedef/etc/init.d
```

Ardından **vi /hedef/etc/init.d/rcS** komutu ile dosyamızı yazmaya başlayalım. Bu dosyada neler yapmamız gerekiyor? `rcS` içerisinde teorik olarak sistemin açılırken yapmasını istediğimiz her türlü işlemi kendi keyfimize göre yaptırabiliriz (Linux'te bir işi yapmanın her zaman birden fazla yolu olduğuna göre keyif burada doğru bir tanımlama :). Ancak bu keyfi işlem adımlarının yanında bir de bazı özel durumlar haricinde yapmamız gereken bir işlem var: kök dosya sistemini oku-yaz kipinde yazılabilir olarak sisteme bağlamak. Linux çekirdeği sistemi açarken ve **init** programını çalıştırdığında kök dosya sistemi salt-okunur olarak sisteme bağlı durumdadır (Elbette bunu da değiştirmenin yolları var ille de ben bildiğimi yaparım diyorsanız). Bu yüzden `rcS` içerisinde kök dosya sistemi oku-yaz kipine geçirilmelidir. Ayrıca `/proc` sanal dosya sistemi de burada sisteme bağlanmalıdır. Bu işlemleri yapan dosya içeriği aşağıdaki gibi olup şu aşamada sistemin açılabilmesi için bize yeterlidir. İleride açılışta yapılacak yeni işlemler eklendiğinde (ağ ayarlarının yapılması gibi) bu dosyanın içeriğinde değişiklikler yapacağız.

Örnek 2. /etc/init.d/rcS

```
#!/bin/sh  
  
## Sistemi read-write moduna geçirelim  
mount -n -o remount,rw /  
  
## Proc'u bağlayalım  
mount /proc
```

Bu dosya mutlaka çalıştırılabilir olmalıdır, **chmod 755 /hedef/etc/init.d/rcS** komutuyla dosyaya çalışma hakları verilmelidir.

`rcS` içerisinde **mount** komutunu kullanma şeklimize bakacak olursanız dosya sistemi türü, aygıt ismi gibi ayrıntıların verilmemesini görmekteyiz. Bu da doğru ayarlanmış bir `/etc/fstab` dosyası sayesinde olabilmektedir. Şimdi bu dosyayı da gene **vi /hedef/etc/fstab** komutu ile aşağıdaki gibi oluşturalım:

Örnek 3. /etc/fstab

#	Device	Point	Type	Options	Dump	Pass
	/dev/hda1	/	minix	defaults,rw	1	0
	proc	/proc	proc	defaults	0	0

Artık `rcS` dosyamız çalışmaya hazır. Şimdi tek yapmamız gereken önyükleyici programımızı ayarlamak ve sistemi yeniden başlatmak olacak.

4.7.3. LILO Ayarları

Sistemimiz neredeyse kendi başına açılabilir hale geldi. Şimdi çekirdeği yüklememizi sağlayacak bir önyükleyici kurmamız ve ayarlamamız gerekiyor. Ben burada önyükleyici olarak LILO kullandım, siz alternatif olarak aynı işlemleri GRUB ile de yapabilirsiniz.

Öncelikle `lilo` uygulaması bir disket ile normal Linux bilgisayarınızdan SBC üzerine taşınmalıdır. Bazı sistemlerde `/sbin/lilo` dosyası bir yönlendirici olup uygulamanın kendisi `/sbin/lilo.real` olarak kurulu olabilir. Benzeri bir durum sizin için de geçerli ise uygulamanın aslını (ismi farklı olsa da) diskete `lilo` adıyla kopyalayın. Disketi SBC'ye takıp bağlayın ve `lilo`'yu buradaki `/hedef/sbin` dizini altına kopyalayın.

LILO uygulamasının çalışabilmesi için gereken bir kaç dosya daha var. Bu dosyaları da her zaman olduğu gibi Linux yüklü bilgisayarımızdan disket yardımı ile SBC üzerine taşımamız gerekir. Aşağıdaki dosyaları Linux sisteminizden bir diskete kopyalayın:

- `/boot/boot-menu.b`
- `/boot/boot.b`
- `/boot/chain.b`
- `/boot/map`

Ardından bu disketi SBC üzerine takarak dosyaları `/hedef/boot` dizini altına kopyalayın.



Bilgi

Hazırlamış olduğumuz Linux çekirdeğinin de bu aşamaya gelinene kadar `/hedef/boot` dizini altına `kernel` adıyla atılmış olduğunu varsayıyoruz.

Gerekli dosyaları bu şekilde SBC üzerine attıktan sonra şimdi bu ana kadar `/hedef` dizini altında oluşturduğumuz SBC sistemine `chroot` edelim (Bu işi `busybox` kurulumu bittikten sonra herhangi bir adımda yapabildik:).

```
# chroot /hedef

BusyBox v0.60.5 (2003.05.29-21:02+0000) Built-in shell (ash)
Enter 'help' for a list of built-in commands.

/ #
```

Artık kök dizinimiz `/`, `/hedef` dizininin altına kaydırılmış durumdadır. Şimdi burada `vi /etc/lilo.conf` komutuyla aşağıdaki gibi bir ayar dosyası oluşturalım.



Dikkat

chroot> işleminden sonra zaten /hedef dizin yapısı içerisinde olduğumuzdan artık /hedef/etc/lilo.conf demek yerine doğrudan /etc/lilo.conf demeliyiz, çünkü bu noktada /hedef dizini bizim için erişilemez durumdadır (zaten biz oyuz, matrix :p) Eski sisteme dönmek için **exit** komutu kullanılmalıdır. Ayrıca burada kullandığımız **vi** programı öncekilerden farklı olarak gerçekte bizim derlediğimiz **busybox** içerisinden çıkan ve ilk açılışta kullandığımız sisteme göre /hedef/bin/vi yani /hedef/bin/busybox olan uygulamadır.

Örnek 4. /etc/lilo.conf

```
lba32
boot=/dev/hda
install=/boot/boot-menu.b
map=/boot/map
prompt
timeout=150
vga=normal
default=Linux

image=/boot/kernel
    root=/dev/hda1
    label=Linux
    read-only
```

Lilo formatıyla ilgili bilgi almak için Linux yüklü bilgisayarınızda **man 5 lilo.conf** komutunu verebilirsiniz. Şimdi artık son adım olarak **lilo** uygulamasını çalıştırıp diskimizin Ana Önyükleme Kaydının (MBR –Master Boot Record) açılışa uygun hale getirilmesini sağlayalım:

```
/ # /sbin/lilo -v
LILO version 22.2, Copyright (C) 1992-1998 Werner Almesberger
Development beyond version 21 Copyright (C) 1999-2001 John Coffman
Released 05-Feb-2002 and compiled at 20:57:26 on Apr 13 2002
MAX_IMAGES=27

Reading boot sector from /dev/hda
Merging with /boot/boot-menu.b
Boot image: /boot/kernel
Added Linux *

Backup copy of boot sector in /boot/boot.0300
Writing boot sector.
/ #
```

Artık sistemin tek başına çalışabilmesi için her şey hazır. Şimdi sistemi yeniden başlatalım.

```
/ # exit
# reboot
```

Disket sürücüde disket olmadığından emin olunuz :) Sisteminiz açılmazsa karşılaştığınız sorunları <[murat \(at\) debian.org](mailto:murat@debian.org)> eposta adresine gönderirseniz bu belgenin sonuna "Problemler ve Çözümler" bölümü ekleyebiliriz.

5. Sistem Kullanıcıları ve Gruplar

Bu aşamada artık tek başına açılabilir bir sistemimiz olduğuna göre, elimizdeki sistemi daha ayrıntılı bir şekilde yapılandırmak için ayarlamalara başlayabiliriz. İşe sistemdeki kullanıcı ve grupların yapılandırılması, sisteme girişlerin düzenlenmesinden başlayalım.

Sistemdeki ana kullanıcı olan `root` için gerekli ayarları elle yapmak durumundayız. Bunun için `vi /etc/passwd` komutuyla açtığımız dosyaya aşağıdaki satırı girelim:

```
root::0:0:root:/root:/bin/sh
```

Dosyayı kaydedip çıktıktan sonra `vi /etc/group` komutu ile `root` kullanıcısının gurubunu aşağıdaki gibi oluşturalım:

```
root:x:0:
```

Artık `root` kullanıcısı sisteme eklenmiş durumdadır. Ancak programların kullanıcı kimliklerinden (`uid`) kullanıcı isimlerine dönüşüm yapabilmeleri için sistemde `libnss_files` kütüphanesi kurulu olmalıdır. Bunun için Linux yüklü bilgisayarınızdan `/lib/libnss_files.so.2` dosyasını disket ile SBC üzerindeki `/lib` dizini altına taşımalsınız. Ardından `vi /etc/nsswitch.conf` komutuyla örnekteki gibi bir ayar dosyası oluşturmanız gereklidir:

Örnek 5. /etc/nsswitch.conf

```
passwd: files
group: files
shadow: files

publickey: files

hosts: files dns
networks: files

protocols: db files
services: db files
ethers: db files
rpc: db files

netgroup: db files
```

Sisteme kullanıcı veya grup ekleme çıkarma işlemlerini `adduser`, `deluser`, `addgroup`, `delgroup` uygulamaları ile yapabilirsiniz (!). Ancak bu yazının hazırlandığı zamanki son `tinylogin` sürümü olan 1.4'te bahsedilen uygulamaların davranışı sizin onlardan beklediğiniz yönde değil. `tinylogin` bu anlamda ağabeyi `busybox` kadar yetenekli değil. Bu yüzden kullanıcı ve grupları `/etc/group` ve `/etc/passwd` dosyalarını düzenleyerek eklemenizi öneriyorum. Örnek olarak `ali` kullanıcıını `users` gurubuna dahil olacak şekilde eklemek için öncelikle `/etc/group` dosyasına `users` grubuna ait bir kayıt ekleyelim, `echo "users:x:100:"` >> `/etc/group` ve ardından `/etc/passwd` dosyasına aşağıdaki kaydı girelim, `vi /etc/passwd`

```
ali::1000:100:Ali Hemenkavrar:/home/ali:/bin/sh
```

Şimdi `ali` kullanıcısının parolasını değiştirelim:

```
/etc # passwd ali
Changing password for ali
Enter the new password (minimum of 5, maximum of 8 characters)
Please use a combination of upper and lower case letters and numbers
```

```
Enter new password:  
Re-enter new password:  
Password changed.  
/etc #
```

Şimdi isterseniz daha önce **getty** ile ilişkilendirdiğiniz 3. sanal konsoldan bu kullanıcı ile sisteme giriş yapabilirsiniz. Bunun için **Alt - F3** ile 3. konsola geçelim:

```
(none) login: ali  
Password:  
  
BusyBox v0.60.5 (2003.05.29-21:02+0000) Built-in shell (ash)  
Enter 'help' for a list of built-in commands.  
  
~ $ id  
uid=1000(ali) gid=100(users)  
~ $
```

Burada dikkatinizi çektii ise, sistem ismi (*none*) olarak gözükmetedir. Sisteme verdiđiniz ismin burada gözükmesi için neler yapılması gerektiđine [Ađ Ayarları](#) (sayfa: 26) bölümünden bakabilirsiniz.

6. Ağ Ayarları

Kullandığınız SBC üzerinde bir ethernet kartı varsa bu hayatınızı ciddi orada kolaylaştıracaktır. Dosya aktarımları, sisteme erişim vb. gibi konularda getirileri oldukça fazladır. Ayrıca SBC'yi bir ağa dahil etme, onunla yapabileceklerinizi de elbette artıracaktır.

Belgenin yazımında kullandığımız SBC üzerinde bir Realtek 8139 ethernet arayüzü mevcut idi. Bu nedenle çekirdeği derlerken RTL8139 desteğini de seçtik ve ethernet kartımızın Linux çekirdeği tarafından tanınabilmesini sağladık. Benzer işlemleri kendi ethernet kartınıza göre sizin de yapmanız gerekiyor.

```
# cat /proc/pci | grep -i eth
Ethernet controller: Realtek Semiconductor Co., Ltd. RTL-8139/8139C/8139C+ ↵
(rev 16).
# dmesg | grep eth
eth0: RealTek RTL8139 Fast Ethernet at 0xc2800000, 00:05:b7:01:20:16, IRQ 10
eth0: Identified 8139 chip type 'RTL-8139C'
#
```

Kartımız sistem tarafından tanındığına göre ayarlamaları yapabiliriz. Bunun için değişebilir nitelikte olan bilgileri statik olarak açılışta çalışacak kabuk programına yazmak yerine `/etc` dizini altında bir genel ayar dosyası oluşturup buraya yazma yöntemini seçelim. Dosyamızın adı `/etc/config` olsun. Şimdi `vi /etc/config` ile bu dosyanın içeriğini aşağıdaki gibi oluşturalım:

```
HOSTNAME=embed
IP=192.168.0.155
NETMASK=255.255.255.0
GATEWAY=192.168.0.1
```

Gerekli parametrelerimizi burada ayarladık. Şimdi bu parametrelerle sistemin açılışı esnasında `/etc/hosts` dosyasını düzenleyecek ve `HOSTNAME` değerini atayacak, ardından ağ kartına ilişkin IP ve yönlendirme ayarlarını yapacak program satırlarını açılışta çalışan betiğimiz olan `/etc/init.d/rcS` dosyasına ekleyelim.

```
## Ayar dosyasını oku
. /etc/config

## Ağ ayarlamaları
if [ -n "$HOSTNAME" ]; then
    hostname $HOSTNAME
    echo "127.0.0.1    localhost" > /etc/hosts
    echo "$IP      $HOSTNAME" >> /etc/hosts
fi

ifconfig lo 127.0.0.1 netmask 255.0.0.0 up

ifconfig eth0 $IP netmask $NETMASK up

if [ -n "$GATEWAY" ]; then
    route add default gw $GATEWAY
fi
```

Bu değişiklikleri yaptıktan sonra sistemi yeniden başlatmaya gerek olmaksızın `/etc/init.d/rcS` betiğini çalıştırıp istenilen sonucu verip vermediğini görebiliriz:

```
# /etc/init.d/rcS
mount: Mounting proc on /proc failed: Device or resource busy
eth0: Setting 100mbps full-duplex based on auto-negotiated partner ability 45e1
```

```
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:05:B7:01:20:16
          inet addr:192.168.0.155  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:1234 (1.2 kiB)  TX bytes:610 (610.0 iB)
          Interrupt:10

# ping 212.156.4.1
PING 212.156.4.1 (212.156.4.1): 56 data bytes
64 bytes from 212.156.4.1: icmp_seq=0 ttl=57 time=23.2 ms
64 bytes from 212.156.4.1: icmp_seq=1 ttl=57 time=22.7 ms
64 bytes from 212.156.4.1: icmp_seq=2 ttl=57 time=21.8 ms
```

SBC üzerinden ağ bağlantısını artık gerçekleştirdiğimize göre sisteme **FTP** ve **telnet** sunucularını kurabiliriz. Bunun için öncelikle **inetd** servisinin kurulması gereklidir.

6.1. Inetd Kurulumu

Sisteme TCP/IP tabanlı bir ağ üzerinden dosya göndermek veya sisteme bağlanmak için FTP ve telnet servislerini kurabiliriz. Bu amaçla basit bir FTP sunucu uygulaması **in.ftpd** ve telnet sunucu uygulaması **in.telnetd** Linux yüklü sistemden SBC üzerine aktarılmalıdır. FTP ve telnet servisleri, ilgili portları sürekli dinlemeyecektir. Bunun yerine her iki port (ve olursa ileride diğer servislerin de ihtiyaç duyduğu portlar) **inetd** (*Internet Daemon*) programı tarafından dinlenecek, porta veri geldiğinde ilgili program arka planda çalıştırılacak, gelen istek programa yönlendirilecektir. **inetd**, **busybox** içerisinde bulunmadığından programın çalıştırılabilmesi için Linux yüklü sistemden **/usr/sbin/inetd** dosyasının da SBC üzerine aktarılması gereklidir. Ayrıca **inetd** kendi içinde **tcp** uygulama yönlendiricisi olarak **/usr/sbin/tcpd** uygulamasını kullandığından gene bu dosyanın da aktarılması gerekmektedir.

Burada dikkat edilmesi gereken bir nokta, artık SBC üzerine attığımız uygulamaların çok fazla kütüphaneye bağımlı olmaya başlamalarıdır. Temel işlerden daha özel işlere doğru çıktığımızda uygulamaların kullandığı kütüphanelerin sayısı da artmaktadır. **tcpd** bunun en basit örneğini oluşturuyor. Linux yüklü bilgisayarımızda aşağıdaki gibi **tcpd** uygulamasının ihtiyaç duyduğu kütüphanelerin neler olduğunu öğrenebiliriz:

```
laptop:~/docbook/embedded$ ldd /usr/sbin/tcpd
libwrap.so.0 => /lib/libwrap.so.0 (0x4001b000)
libc.so.6 => /lib/libc.so.6 (0x40024000)
libnsl.so.1 => /lib/libnsl.so.1 (0x40134000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Son üç kütüphane SBC sistemimizde zaten mevcut iken **libwrap.so.0** henüz mevcut değildir. Dolayısıyla **tcpd** uygulamasının çalışabilmesi için bu dosyanın da dosya sistemi üzerine atılması gereklidir. Aksi takdirde uygulama çalışmayacak ve aşağıdaki gibi bir hata mesajı verecektir:

```
/usr/sbin/tcpd: error while loading shared libraries: libwrap.so.0: cannot open
shared object file: No such file or directory
```

Bu aşamadan sonra SBC üzerine atacağımız çalıştırılabilir her uygulama için buradaki işlem adımlarını tekrarlayarak gerekli kütüphanelerle birlikte atmalıyız.

Inetd programı, hangi portları dinleyeceğini ve gerektiğinde çalıştırması gereken uygulamaların neler olduğunu **/etc/inetd.conf** ayar dosyasından okur. **man inetd** ve **man inetd.conf** komutlarıyla program

ve ayar dosyasının biçimi hakkında sisteminizden yardım alabilirsiniz. Ftp ve telnet servisleri için gerekli `inetd.conf` dosyası aşağıdaki gibi olmalıdır:

Örnek 6. /etc/inetd.conf

```
# /etc/inetd.conf dosyamız
ftp  stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.ftpd
telnet stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.telnetd
```

Bu dosyanın satırlarındaki ilk sözcük servisin ismini göstermektedir. Buradan alınan isim bilgisi (`ftp`, `telnet`) `/etc/services` dosyası içerisinde aranacaktır. Bu durumda `/etc/services` dosyası Linux yüklü bir sistemden aynen kopyalanmalı veya sadece aşağıdaki kayıtları içerecek şekilde oluşturulmalıdır:

```
telnet      23/tcp
ftp-data    20/tcp
ftp         21/tcp
```

`inetd`, `telnet` vb. gibi servislerin ilk ayarlanması süreci boyunca `syslogd` programının arka planda çalışıyor olması çıkabilecek problemlerin çözümünde oldukça faydalı bilgiler sağlayacaktır. `syslogd` açık olduğu müddetçe, `syslog` mekanizmasını kullanan programlar hata durumlarını günlüğe yazabileceğinden `/var/log` dizini altındaki dosyaların incelenmesiyle sorunun nereden kaynaklandığını bulmak çok daha kısa zaman alacaktır. `syslogd` programını çalıştırmak için komut satırında `syslogd` yazmanız yeterlidir. Her açılışta çalışması istenirse bu defa `/etc/init.d/rcS` dosyası içerisinde herhangi bir satıra

```
syslogd &
```

kayı eklenmelidir. Kullanımına artık gerek kalmadığı görüldüğünde `rcS` içerisinde bu satır kaldırılmalıdır (gömülü bir sistemde bu anlamda günlük tutulması, sistem belirli bir kararlılığa ulaştıktan sonra mutlaka önlenmelidir).

Bu ayarlamalar yapıldıktan sonra `rcS` içerisinde `inetd` şeklinde `inetd` servisinin açılışta çalışabilmesi için gerekli satırın bulunduğundan emin olunmalıdır. Sistemi yeniden başlatmadan `inetd` programını test edebilmek için komut satırından `inetd` yazılması yeterlidir. Eğer bir hata oluşursa `/var/log/messages` dosyası incelenmelidir.

6.2. FTP Sunucu Kurulumu

SBC üzerine kuracağınız FTP sunucu için dikkat etmeniz gereken bazı noktalar vardır. Bir kere amacımız sadece basit anlamda dosya aktarımını gerçekleştirebilmektir. Karmaşık özelliklere sahip çok büyük bir FTP sunucu değil, olabildiği kadar küçük, sağlam ve güvenilir bir uygulama bizim için yeterlidir. Bir diğer önemli özellik, FTP sunucunun PAM desteği olmadan doğrudan sistemdeki `passwd` ve `shadow` dosyaları ile çalışabilmesi gerekliliğidir. Aksi takdirde FTP sunucuyu çalıştırmak için SBC üzerindeki sistemimize PAM desteği vermek zorunda kalabiliriz ki bu da "Türkler için ayda nasıl yürünür?" konulu bir eğitim kadar gereksizdir.

Yukarıda bahsettiğim koşullar ve yıllardan beri kendini kanıtlamış sağlam yapısı nedeniyle ben `bsd-ftpd` sunucusunu kullanmaya karar verdim. `bsd-ftpd`, OpenBSD FTP sunucusunun Linux işletim sistemine taşınmış halidir. Programı <http://www.eleves.ens.fr:8080/home/madore/programs/> adresinden indirebilirsiniz. Arşivi indirip açtıktan sonra içerisinde çıkan `Makefile` dosyasını bir metin düzenleyicide açarak dosyanın başındaki ilgili yerleri aşağıdaki hale getiriniz:

```
#CFLAGS = $(OPT_CFLAGS) $(EXTRA_CFLAGS) -DTCPWRAPPERS -DUSE_PAM -
-DAUTO_UNCOMPRESS -DINTERNAL_LS
#LIBS = $(EXTRA_LIBS) -lutil -lwrap -lnsl -lcrypt -lpam -ldl
## If you prefer shadow password support, try this:
```

```
CFLAGS = $(OPT_CFLAGS) $(EXTRA_CFLAGS) -DTCPPWRAPPERS -DUSE_SHADOW -\n-DAUTO_UNCOMPRESS -DINTERNAL_LS\nLIBS = $(EXTRA_LIBS) -lutil -lwrap -lnsl -lcrypt
```

Bu sayede programın PAM ile çalışmak üzere değil, sistemdeki kullanıcı veritabanıyla çalışabilecek şekilde derlenmesini sağladık.

Makefile üzerinde gerekli değişiklikleri yaptıktan sonra **make** komutu ile programı derleyebiliriz. Derleme işlemi bittiğinde yaklaşık 80 kB büyüklüğünde **ftpd** uygulaması oluşacaktır. **strip ftpd** komutu ile uygulama içerisinde ihtiyaç duymayacağımız sembollerin çıkartılmasını da sağlayınca uygulamanın boyu 65 kB seviyelerine inecektir. Bu uygulamayı SBC sistemimizdeki `/usr/sbin` dizini altına `in.ftpd` adıyla taşımalıyız.



Bilgi

bsd-ftpd uygulamasını derlemek için GNU Libc kütüphanelerin yanısıra `libwrap0-dev` paketinden çıkan geliştirme kütüphanelerine de ihtiyaç vardır (`tcpd.h` vb. için).

Derlediğimiz uygulamayı SBC üzerine atıp arka planda **inetd**'nin de çalıştığından emin olunca Linux yüklü sistemimizden SBC üzerindeki FTP sunucumuzu test edebiliriz:

```
laptop:~$ ftp 192.168.0.155\nConnected to 192.168.0.155.\n220 embed FTP server (Version 6.5/OpenBSD, linux port 0.3.3) ready.\nName (192.168.0.155:demirten): root\n331 Password required for root.\nPassword:\n230 User root logged in.\nRemote system type is UNIX.\nUsing binary mode to transfer files.\nftp> ls /boot\n200 PORT command successful.\n150 Opening ASCII mode data connection for '/bin/ls'.\ntotal 764\n-rw-r--r--  1 root  root    7964 Jun  1 10:49 boot-menu.b\n-rw-r--r--  1 root  root     512 Jun  1 14:34 boot.0300\n-rw-r--r--  1 root  root    7964 Jun  1 10:50 boot.b\n-rw-r--r--  1 root  root     728 Jun  1 10:50 chain.b\n-rw-r--r--  1 root  root  744821 Jun  1 12:05 kernel\n-rw-----  1 root  root   12288 Jun  1 14:34 map\n226 Transfer complete.\nftp> bye\n221 Goodbye.\nlaptop:~$
```

Bu ekran çıktısı artık disket sürücüyü sistemden çıkarabileceğimizin müjdesini veriyor bize :) Peki ağ bağlantısı sağlayamayanlar ne yapacak? Bunun için [Seri Portların Kullanımı](#) (sayfa: 32) bölümüne bakın.

6.3. Telnet Sunucu Kurulumu

Bir önceki bölümde anlatılan FTP sunucu kurulumuna benzer şekilde şimdi de **telnet** servisini sisteme kuralım. FTP'ye oranla bu kurulumumuz daha basit olacak çünkü uygulamayı yeniden derlemeye ihtiyacımız yok.

telnet servisi için temelde `in.telnetd` ve `/usr/lib/telnetlogin` uygulamasının kurulumu yeterlidir. Hemen gerekli kütüphanelerin bizde olup olmadığına bakalım:

```
laptop:~$ ldd /usr/sbin/in.telnetd
libutil.so.1 => /lib/libutil.so.1 (0x4001b000)
libc.so.6 => /lib/libc.so.6 (0x4001f000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Buradaki çıktıda yer alan `libutil.so.1` kütüphanesi SBC sistemimizde mevcut değil. Bu durumda taşımamız gereken dosyalar arasına bunu da ekliyoruz. FTP servisimiz çalıştığına göre hemen FTP kullanarak dosyaları SBC üzerine aktaralım:

```
laptop:~$ ncftp -u root 192.168.0.155
NcFTP 3.1.3 (Mar 27, 2002) by Mike Gleason (ncftp@ncftp.com).
Connecting to 192.168.0.155...
embed FTP server (Version 6.5/OpenBSD, linux port 0.3.3) ready.
Logging in...
Password requested by 192.168.0.155 for user "root".

    Password required for root.

Password:
User root logged in.
Logged in to 192.168.0.155.
ncftp /root > cd /usr/sbin
ncftp /usr/sbin > put /usr/sbin/in.telnetd
/usr/sbin/in.telnetd:                31.73 kB   993.88 kB/s
ncftp /usr/sbin > cd /usr/lib
ncftp /usr/lib > put /usr/lib/telnetlogin
/usr/lib/telnetlogin:                 5.60 kB   147.83 kB/s
ncftp /usr/lib > cd /lib
ncftp /lib > put /lib/libutil.so.1
/lib/libutil.so.1:                   7.25 kB   193.60 kB/s
ncftp /lib >
```



Bilgi

FTP üzerinden gönderdiğiniz dosyaların izinleri kaybolabilir. Çalıştırılabilir dosyalar FTP ile gönderildiklerinde bu özelliklerini kaybedeceklerdir. Bu nedenle SBC sisteminde `chmod` komutu ile gerekli erişim hakları verilmelidir. Bu örnek için gerekli komut `chmod 755 /usr/sbin/in.telnetd /usr/lib/telnetlogin` şeklindedir.

Her şey hazır olduğuna göre Linux yüklü bilgisayarımızdan SBC üzerindeki `telnet` servisini test edelim:

```
laptop:~$ telnet 192.168.0.155
Trying 192.168.0.155...
Connected to 192.168.0.155.
Escape character is '^]'.

Linux 2.4.20 (embed) (ttypl)

embed login: ali
Password:

BusyBox v0.60.5 (2003.05.29-21:02+0000) Built-in shell (ash)
Enter 'help' for a list of built-in commands.
```

```
~ $
```

İçerdeziz :) **telnet** servisimiz düzgün bir şekilde çalışıyor.



Dikkat

FTP ve telnet servislerini bu şekilde ayarladıktan sonra `/etc/init.d/rcS` dosyası içerisinde **inetd** uygulamasının her sistem açılışında başlatılması için

```
/usr/sbin/inetd
```

satırı eklenmelidir (sonunda & yok).

7. Seri Portların Kullanımı

Önceki bölümde ethernet arayüzünü yapılandırarak ağa dahil olma, FTP ve telnet servislerini kullanma konularını işledik. Bu sayede FTP kullanarak dosyaları disketlerle taşıma zahmetinden kurtulurken telnet servisini kullanarak SBC'ye bir monitor bağlamadan çalışabilme şansına sahip oluyorduk. Ancak sisteminizde ethernet desteği yoksa benzeri kolaylıkları seri portu kullanarak sağlayabiliriz. Sisteminizde ethernet desteği olsa bile buradaki adımları uygulayarak seri portu farklı amaçlar için kullanabilirsiniz.

Buradaki adımları uygulayabilmeniz için seri iletişim kablosuna ihtiyacınız vardır. Bu kablo aslında iki ucu *dişi* DSUB girişli bir telefon kablosunun bir tarafındaki Transmit Data bacağının diğer taraftaki Receive Data, Receive Data bacağının ise diğer taraftaki Transmit Data'ya bağlandığı basit bir kablodur ve sadece bu iki bağlantıyı yapmak aslında yeterlidir. Ancak, akış kontrolleri (*hardware flow control*) için RTS/CTS vb. sinyallerinin de iki sistemin seri portu arasında taşınabilmesi gereklidir. Bu nedenle aşağıdaki diyagrama göre yapacağınız bir seri iletişim kablosu daha çok işinize yarayacaktır: (İkili bükülü çiftler *çiftler* sütununda gösterilmiştir.)

Seri iletişim kablosunun bağlantı bilgisi

	25 Pin	9 Pin	Çiftler	9 Pin	25 Pin	
FG (Frame Ground)	1	1	X	–	1	FG
TD (Transmit Data)	2	3	1	2	3	RD
RD (Receive Data)	3	2	1	3	2	TD
RTS (Request To Send)	4	7	2	8	5	CTS
CTS (Clear To Send)	5	8	2	7	4	RTS
SG (Signal Ground)	7	5	–	5	7	SG
DSR (Data Set Ready)	6	6	3	4	20	DTR
DTR (Data Terminal Ready)	20	4	3	6	6	DSR

7.1. Seri Porttan Dosya Aktarımı

Önceki bölümde anlatılan niteliklere uygun seri iletişim kablosu edinebildiyse şimdi seri portlar arasında dosya transferi için gerekli ayarlamaları yapabiliriz. Bunun için çok gerekli olmamakla birlikte, seri portu kullanılacak ise başka amaçlar için de bize yardımcı olabilecek **minicom** uygulamasını sisteme kurabiliriz. Hemen gerekli kütüphanelerin neler olduğunu öğrenelim:

```
laptop:~/docbook/embedded$ ldd /usr/bin/minicom
libncurses.so.5 => /lib/libncurses.so.5 (0x4001b000)
libc.so.6 => /lib/libc.so.6 (0x4005a000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Çıktıdan görüldüğü gibi **minicom** uygulamasının yanısıra SBC sistemimizde henüz bulunmayan `libncurses.so.5` kütüphanesinin de taşınması gereklidir. Bu dosyaları (ağ bağlantımızın olmadığını varsayarsak) diskette SBC üzerine taşımamız gerekir.

minicom uygulamasının çalışabilmesi için `/usr/share/terminfo` dizini altında kullanacağımız uçbirim türlerine uygun dosyalar bulunmalıdır (Aslında bu gereklilik kullanılan `ncurses` kütüphanesinden kaynaklanmaktadır). SBC üzerinde `linux`, `xterm` ve `xterm-color` uçbirim türlerinin kullanılacak olduğunu düşünerek Linux yüklü sistemden aşağıdaki dosyaları, SBC üzerinde aynı dizine taşıyın:

```
/usr/share/terminfo/l/linux
/usr/share/terminfo/x/xterm
/usr/share/terminfo/x/xterm-color
```



Bilgi

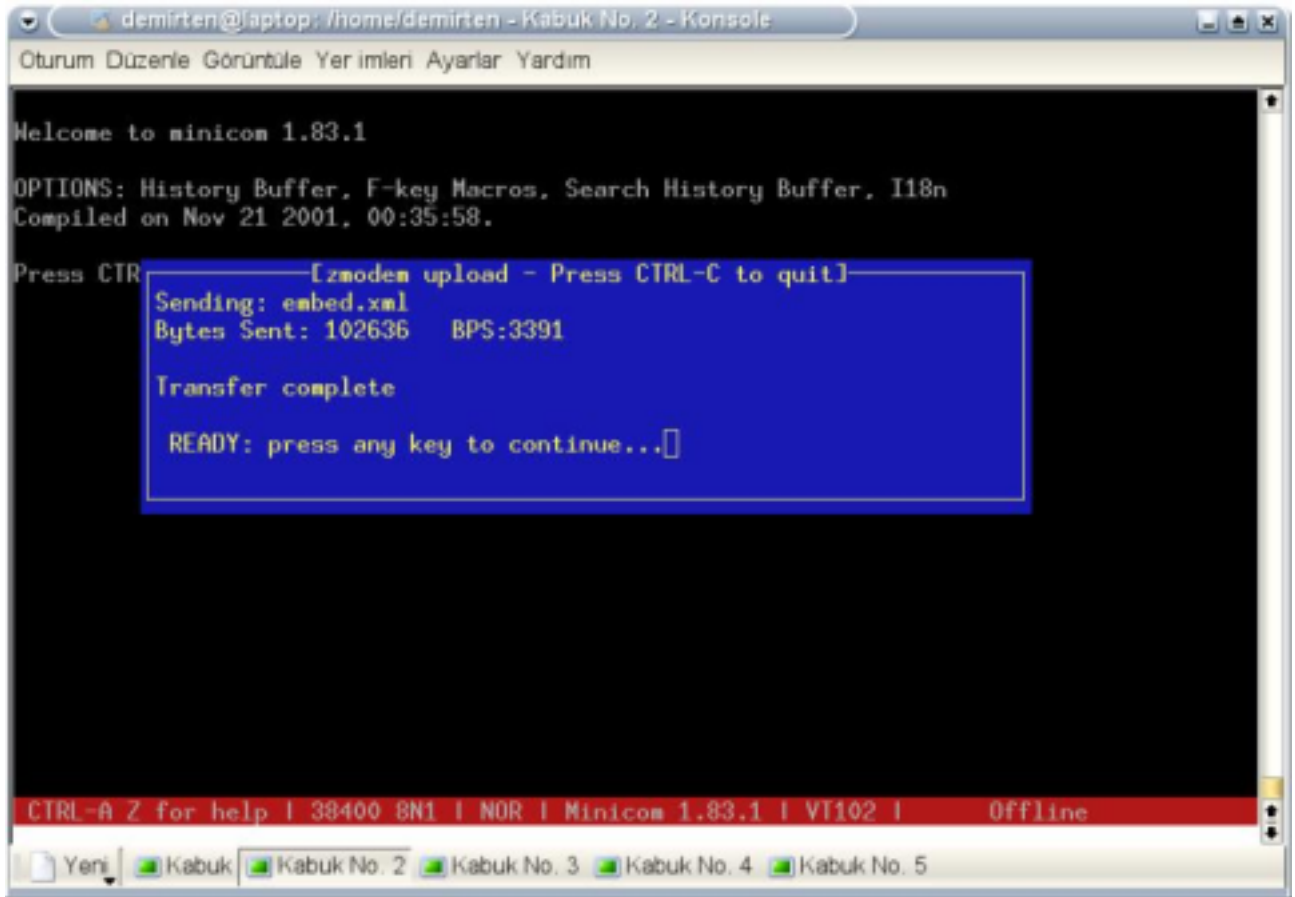
SBC üzerinde bu dizinler olmadığından oluşturmalısınız:

```
mkdir -p /usr/share/terminfo/l  
mkdir -p /usr/share/terminfo/x
```

Programı **minicom -c on** komutuyla çalıştırırsanız renk desteğini etkin hale getirebilir daha güzel bir ekran ile çalışabilirsiniz.

Her iki sistemde **minicom**'u çalıştırıp seri port ayarlarını yaptıktan sonra (port, baudrate vb.) dosya aktarım işlemlerini gerçekleştirebiliriz. Bunun için ZModem protokolünü kullanabiliriz. **minicom** tek başına bunu yapamaz, **lrzsz** uygulamasından çıkan **sz** ve **rz** programlarına ihtiyaç duymaktadır. Linux yüklü sisteminizden **/usr/bin/sz** ve **/usr/bin/rz** uygulamalarını SBC üzerine aktardıktan sonra **minicom** uygulaması içerisinde **Ctrl - A - S** ile dosya aktarım menüsüne ulaşabilir, oradan ZModem protokolünü ve aktarmak istediğiniz dosyaları seçerek iki sistem arasında veri alışverişini yapabilirsiniz (her iki tarafta **minicom** uygulaması açık olmalı).

Şekil 9. Minicom ile dosya transferi



7.2. Seri Konsol

Seri port kullanarak Linux sisteminize erişebilmek, özellikle SBC'ye bir monitor takmak istemediğinizde oldukça faydalı bir özelliktir. Bu yöntemde de *seri iletişim kablosu* kullanılmalıdır. Seri konsol hakkında ayrıntılı bilgiler Linux çekirdek kaynak kodlarından çıkan `Documentation/serial-console.txt` belgesinde mevcuttur. Burada hızlıca seri konsolu nasıl ayarlabileceğinizden bahsedeceğim.

Öncelikle işe çekirdekten başlamak gerekiyor. Seri konsol üzerinden bir Linux sistemine erişilecek ise, o sistemde kullanılacak olan çekirdekte bu destek etkin hale getirilmelidir. İlgili seçenek, çekirdek bileşenlerini seçim

menülerinden *Character devices* bölümü altındaki *Support for console on serial port* seçeneğidir.

Çekirdek kısmı tamamlandıktan sonra şimdi bir seri portu bu amaçla kullanacağımızı belirtmemiz gerekiyor. Bunun için `/etc/inittab` dosyasının altına aşağıdaki gibi bir kayıt eklemeliyiz:

```
ttyS1::respawn:/sbin/getty -L ttyS1 38400 vt100
```

Bu örnekte `ttyS1` yani COM2 portuna seri konsol olarak bağlanılabileceğini belirtiyoruz. İletişim hızı 38400 bps olarak ayarlanmış başka bir sistemden, seri port aracılığıyla bu sistemin COM2 portuna veri gönderildiğinde **getty** programı devreye girecek ve kimlik doğrulama aşamasından sonra kabuğa düşülebilecektir. Bu amaçla *NIX sistemlerde **minicom**, windows sistemlerde ise **HyperTerminal** gibi programlar kullanılabilir.

Yukarıdaki gibi ayarlamış olduğumuz bir seri konsol ile SBC sistemini yükledikten sonra (ve de `/sbin/init` çalıştıktan sonra) seri porttan sisteme girmemiz mümkün olmaktadır. Ancak bazen LILO satırına da müdahale etmemiz gerekebilir veya Linux çekirdeğinin yüklenirken normalde ekrana çıkardığı mesajları görmek isteyebiliriz. Bu aşamada henüz **init** programı çalışmaya başlamadığından seri konsol kullanımımıza hazır değildir. Seri konsolu LILO'dan itibaren kullanmak istediğimizde bu defa `/etc/lilo.conf` ayar dosyası içerisine aşağıdakine benzer bir kayıt ekleyerek **lilo** programına seri konsolu kullanmak istediğimizi belirtebilir ve istemiş olduğumuz verileri normalde olduğu gibi seri port üzerinden başka bir sisteme alabiliriz.

```
## Ortak ayarlar kısmına aşağıdaki satır eklenir
serial = 1,38400n8      ## 1 = ttyS1, 38400 baud rate, no parity, 8 bits

## çekirdek ayarları kısmına aşağıdaki satır eklenir
## Böyle bir kullanımda çıktı hem COM2 portuna hem de tty0'a gönderilir
append = "console=ttyS1,38400n8 console=tty0"
```

Bu satırları ekledikten sonra değişikliklerin etkin olması için **lilo -v** komutunu çalıştırmalısınız.



Bilgi

GRUB kullanmadığım için aynı işlemlerin GRUB'da nasıl yapıldığı hakkında bir fikrim yok, yapan biri olursa ve bana da iletirse buraya ekleyebilirim.

8. Linux Açılış Süreci

Bu bölüm doğrudan gömülü sistemlerde Linux kullanımıyla ilgili olmasa da, sisteme daha fazla hakim olabilmek, Linux açılışında sırasıyla neler olup bittiğini öğrenmek ve de en önemlisi bir sonraki bölümde anlatılacak olan *Initial Ramdisk* kullanımını daha iyi açıklayabilmek amacıyla belgeye eklenmiştir. Özellikle **initrd** kullanacak iseniz hangi işlemi nerede yapmanız gerektiğini, bunun sistem açılışını hangi aşamalarda nasıl etkileyeceğini bilmeniz gereklidir, aksi takdirde işler içinden çıkılmaz bir hal alacaktır.

Şimdi bilgisayarınıza elektrik verildiği andan itibaren neler olduğuna sırasıyla bakalım. Bu bölümde SBC sistemimizi değil, normal bir masaüstü bilgisayarı örnek alalım. SBC'ler genelde daha özel BIOS ve donanımlara sahip olduğundan ekranda göreceğiniz mesajlar farklı olacaktır.

Bilgisayarınızın açma butonuna bastınız, anakartınıza elektrik gelmeye başladı. Bakalım neler olacak.

Anakart BIOS'u, ekran kartının BIOS'unu tetikler

Bilgisayarınıza elektrik verildiğinde anakartınız üzerindeki BIOS uygulaması çalışmaya başlar ve o da ilk iş olarak ekran kartınızın BIOS'unu çalışması için tetikler. Ekran kartı BIOS'u çalışır ve kendini ilklendirir. Çoğu ekran kartı markası ve modeliyle ilgili ekrana bir şeyler yazar. Bu nedenle dikkat ederseniz ilk açılışta daha anakartın BIOS bilgilerinin gösterildiği ekran gelmeden, öncelikle ekran kartıyla ilgili bilgiler görüntülenir.



İpucu

Bilgisayarınızı yeniden başlattığınızda (*soft reset*) ekran kartı BIOS'u ilklendirme işlemlerini yeniden yapmaz, daha doğrusu anakart tarafından bu amaçla tetiklenmez.

```
ATI Technologies Inc Radeon Mobility M6 LW
32768 kB

VGA/VBE BIOS, Version 2.2
....
```

ve ekran temizlenir.

Anakartınızın BIOS'u kendini ilklendirir

Bu aşamada anakart BIOS'u çalışarak donanımların varlığından haberdar olur. Bellek, disket sürücü vb. gibi aygıtları test eder. Ekrana aşağıdakine benzer mesajlar çıkarır:

```
Memory Test: 131072K OK
Award Plug and Play BIOS Extension v1.0A
Copyright (C) 1998, Award Software, Inc.
Press DEL to enter SETUP
```

ve ekran temizlenir.

SCSI Controller BIOS'u tetiklenir

Sisteminizde bir SCSI kartı var ise SCSI veriyoluyla ilgili ilklendirmeleri yapar ve aşağıdakine benzer mesajlar çıkarır:

```
Adaptec AHA-2940 Ultra/Ultra W BIOS v 1.23
(C) 1996 Adaptec, Inc. All Rights Reserved.
>>> Press for SCSISelect(TM)
      SCSI ID:LUNNUMBER #:# 0:0 - MICROP 3243-19 1128RQAV - Drive C: (80h)
```

ve ekran temizlenir.

Donanım Özet Bilgisi

Bu aşamada BIOS'unuz donanımınız hakkındaki bilgileri özetler (BIOS'tan yapılacak ayarlarla bu ekranların çıkması engellenmiş olabilir veya anakartınızın BIOS'u bu türden bir bilgi sunmayabilir).

Award Software, Inc.

CPU Type	Cyrix M II/IBM 6x86MX	Base Memory	640K
Co-Processor	Installed	Extended Memory	130048K
CPU Clock	233	Cache Memory	1024K
Diskette Drive A	1.44M, 3.5 in.	Display Type	EGA/VGA
Diskette Drive B	None	Serial Port(s)	3F8 2F8
Pri. Master Disk	None	Parallel Port(s)	278
Pri. Slave Disk	None	Bank 0/1 DRAM Type	None
Sec. Master Disk	None	Bank 2/3 DRAM Type	EDO
Sec. Slave Disk	None	Bank 4/5 DRAM Type	EDO

Ardından eğer varsa önyüklemeye sektörü virüslerine karşı BIOS'taki virüs kontrol kodu çalıştırılır ve önyüklemeye sektörü bilgisinin değişip değişmediği test edilir:

```
!!!! Trend ChipAwayVirus On Guard!!!! Now Detecting Boot Sector Type Virus...
ChipAwayVirus BIOS Version 1.62
Verifying DMI Pool Data.....
```

ve ekran temizlenir.

LILO çalışır

Ardından önyüklemeye için kullandığınız ortamın Ana Önyüklemeye Kaydı (MBR) okunur. Lilo kullandığınızı varsayarsak LILO içerisinden yapılacak ayarlamalarla değiştirilebilir olmasına karşın temelde ekrana LILLO mesajı çıkartılır. Bu mesajdaki her bir karakterin aslında özel bir anlamı vardır. Bazen LILLO yerine LI veya LIL deyim takılı kaldığını görebilirsiniz.

İlk "L" karakteri, **lilo**'nun kendini daha rahat hissedeceği 0x0009A000 adresine taşınması sonrasında ekrana çıkartılır.

"l" karakteri önyükleyici kodunun ikinci aşamasına geçilmeden hemen önce çıkartılır.

lilo'nun ikincil önyükleyici kodu ikinci "L" karakterini ekrana çıkartır, çekirdeğin bölümlerini gösteren tanımlayıcıları 0x0009d200 adresine yükler ve son olarak işlem bitince ekrana "O" karakterini çıkartır.

Ardından yapılan **lilo** ayarlamalarına göre değişebilecek bir `boot` komut istemi gelir. Burada birazdan yükleyecek olduğumuz Linux çekirdeğine parametre aktarabiliriz (örneğin kök dosya sisteminin ne olacağı).



Bilgi

Eğer **lilo** kullanılmıyorsa ve önyükleyici kodu Linux çekirdeğinin başını gösteriyorsa `linux/arch/i386/boot/bootsect.S Loading...` mesajını ekrana çıkartır.

Linux çekirdeği çalışır

`/linux/arch/i386/boot/setup.S` kod parçası işlemciyi gerçek kipten 32 bit korumalı kipe geçirir. Sıkıştırılmış çekirdek (*bzImage*) açılır ve 0x00100000 adresine yüklenir. `linux/arch/i386/head.S` ve `linux/arch/i386/boot/compressed/misc.c` kodları sıkıştırılmış çekirdeği açma işlemlerini tamamlar ve işlemcinin yazmaçlarını (*register*) ayarlar. Bu aşamalarda aşağıdaki satır ekrana bastırılır:

```
Uncompressing Linux... Ok. Booting kernel.
```

Intel x86 mimarisine özgü `setup.S` görevini tamamladıktan sonra 0x00100000 adresine atlama (*jump*) yapar ve genel Linux kodunun çalışmasını sağlar:

İşlemci, konsol ve bellek ilklendirmeleri

Kesmelerin tanımlandığı `linux/init/main.c`'de `start_kernel(void)` çalışmaya başlar. Ardından `linux/kernel/module.c` konsol ve PCI veriyolu için gerekli sürücülerini yükler. Bu noktadan itibaren çekirdek mesajları bellekte saklanmaya başlar, bu mesajlara `/bin/dmesg` ile bakılabilir. Mesajların kalıcı olarak saklanabilmesi için çoğu zaman sistem tarafından `/var/log` dizini altında ayrıca bir dosyada kaydedilir. Sonra aşağıdaki satır `linux/init/version.c` tarafından ekrana çıkartılır:

```
Linux version 2.4.20 (root@laptop) (gcc version 2.95.4 20011002 ǀ
(Debian prerelease)) #16
Cts May 3 19:06:51 EEST 2003
```

Ardından `console_init(..)` işlevi tarafından aşağıda mesaj ekrana çıkartılır:

```
Console: colour VGA+ 80x25
```

Bu işlevden hemen sonra `calibrate_delay()` işlevi çalışır ve aşağıdaki mesaj ekrana çıkartılır:

```
Calibrating delay loop... 2392.06 BogoMIPS
```

Sonra `mem_init()` işlevi çalışır ve işlem bitince aşağıdaki mesajı verir:

```
Memory: 257316k/262116k available (900k kernel code, 4412k reserved, ǀ
410k data, 64k init, 0k highmem)
```

Ardından önbellek alanları düzenlenir ve işlemci türü test edilir, işlemcinin kabiliyetleri tespit edilir.

```
Dentry cache hash table entries: 32768 (order: 6, 262144 bytes)
Inode cache hash table entries: 16384 (order: 5, 131072 bytes)
Mount-cache hash table entries: 4096 (order: 3, 32768 bytes)
Buffer-cache hash table entries: 16384 (order: 4, 65536 bytes)
Page-cache hash table entries: 65536 (order: 6, 262144 bytes)
CPU: L1 I cache: 0K, L1 D cache: 8K
CPU: L2 cache: 512K
Intel machine check architecture supported.
Intel machine check reporting enabled on CPU#0.
CPU:      After generic, caps: bfeb9fff 00000000 00000000 00000000
CPU:      Common caps: bfeb9fff 00000000 00000000 00000000
CPU: Intel Mobile Intel(R) Pentium(R) 4 - M CPU 1.80GHz stepping 07
Enabling fast FPU save and restore... done.
Enabling unmasked SIMD FPU exception support... done.
Checking 'hlt' instruction... OK.
POSIX conformance testing by UNIFIX
```

PCI veriyolu ilklendirmeleri

`mpci_init()` işlevi `linux/arch/i386/kernel/bios32.c` aşağıdaki mesajları çıkartır:

```
PCI: PCI BIOS revision 2.10 entry at 0xf0e40, last bus=2
PCI: Using configuration type 1
```

Hemen ardından `pci_init()` işlevi çalışarak PCI veriyolunu test etmeye başlar, alttaki mesajı çıkartır:

```
PCI: Probing PCI hardware
```

Ağ ilklendirmeleri

`socket_init()` işlevi ağ ile ilgili ilklendirmeleri yapar. `linux/net/socket.c` içerisinde aşağıdaki mesaj çıkartılır:

```
Linux NET4.0 for Linux 2.4
Based upon Swansea University Computer Society NET3.039
```

İlk süreç başlatılır (0): Kernel Idle Thread

Bu noktada `init()` işlevi tarafından `mkswapd_setup()` çağırılır ve aşağıdaki mesaj çıkartılır (*Linux çekirdeği tamamen aktif olduktan sonra çağırılan /sbin/init ile karıştırmayın*)

```
Starting kswapd
```

Aygıt sürücülerine ilişkin iklendirmeler

Çekirdek kodu bu noktadan sonra desteklediği aygıtlar ve dosya sistemleri için iklendirmeleri yapar, işlemlerini tamamlar ve ardından `/sbin/init` programını `fork()` ile çalıştırır.

Bu aşamalarda seri ve paralel portlar, karakter ve blok tabanlı erişim sağlanan aygıtlar, varsa SCSI veriyolu üzerindeki aygıtlar, ethernet kartı, farklı dosya sistemleri, PPP protokolü vb. tüm destekler iklendirilir ve kullanıma hazır hale getirilir. Kullanılan donanım ve çekirdeğin özelliklerine göre çok farklı mesajlar ekrana çıkartılır, aşağıdaki örnek de bunlardan biridir:

```
Serial driver version 5.05c (2001-07-08) with MANY_PORTS SHARE_IRQ ↵
SERIAL_PCI enabled
ttyS00 at 0x03f8 (irq = 4) is a 16550A
ttyS01 at 0x02f8 (irq = 3) is a 16550A
PCI: Found IRQ 11 for device 00:1f.6
PCI: Sharing IRQ 11 with 00:1f.5
PCI: Sharing IRQ 11 with 02:07.1
Uniform Multi-Platform E-IDE driver Revision: 6.31
ide: Assuming 33MHz system bus speed for PIO modes; override with ↵
idebus=xx
ICH3M: IDE controller on PCI bus 00 dev f9
PCI: Found IRQ 9 for device 00:1f.1
PCI: Sharing IRQ 9 with 02:07.2
ICH3M: chipset revision 2
ICH3M: not 100% native mode: will probe irqs later
   ide0: BM-DMA at 0x8400-0x8407, BIOS settings: hda:DMA, hdb:pio
   ide1: BM-DMA at 0x8408-0x840f, BIOS settings: hdc:DMA, hdd:pio
hda: IC25N040ATCS04-0, ATA DISK drive
hdc: TOSHIBA DVD-ROM SD-R2102, ATAPI CD/DVD-ROM drive
ide0 at 0x1f0-0x1f7,0x3f6 on irq 14
ide1 at 0x170-0x177,0x376 on irq 15
blk: queue c027fde4, I/O limit 4095Mb (mask 0xffffffff)
hda: 78140160 sectors (40008 MB) w/1768KiB Cache, CHS=4864/255/63, ↵
UDMA(100)
Partition check:
   hda: hda1 hda2 hda3 hda4 < hda5 hda6 >
Floppy drive(s): fd0 is 1.44M
```

İklendirme işlemleri bittikten sonra kök dosya sistemi bağlanır ve `/sbin/init` programı çalıştırılır. Çekirdek son olarak aşağıdaki satırları ekrana çıkartarak açılış işlemini tamamlar. Bu aşamadan sonra `dmesg` ile gördüğünüz çekirdek mesajları artık `/sbin/init` tarafından gerçekleştirilen sistem açılışı, sonradan eklenen, çıkarılan çekirdek modülleri vb. hakkında olacaktır, yükleme işlemi çekirdek için tamamlanmıştır.

```
VFS: Mounted root (ext3 filesystem) readonly.
Freeing unused kernel memory: 64k freed
```

Çekirdek içerisinden başlatılan *idle thread* (süreç numarası 0) tarafından çalıştırılan `/sbin/init` uygulaması daima sistemdeki 1 numaralı süreçtir. Geri kalan tüm süreçlerin anası durumundadır. Çalıştığında ekrana *INIT: version 2.76 booting* gibi bir mesaj çıkarır ve `/etc/inittab` dosyasını okuyarak gerekli işlemleri yapmaya başlar.

9. Initial Ramdisk (Initrd)

10. X Window System Kurulumu

11. Türkçe Nasıl?

11.1. Yerel (Locale) Desteği

Programlarınızın Türkçe yerelinden haberdar olabilmesi için yapmamız gereken bir kaç işlem var. Ancak bu işlemlere geçmeden önce bir hatırlatma yapmak istiyorum. Eğer beklediğiniz etkiye ulaşmak için sisteme hangi dosyaları kopyalamanız gerektiğini bir türlü bulamıyorsanız, SBC sistemine **strace** uygulamasını kurmanızı şiddetle öneririm. **strace** çok küçük fakat inanılmaz işe yarayan bir uygulamadır. Programların hangi sistem komutlarını ve sinyallerini kullandıklarını gösterir. Peki doğru dosyayı bulmakta bize nasıl yardımcı olabilir? Hemen bir örnek verelim:

Örnek 7. Strace kullanımı

```
/ # export LANG=tr_TR
/ # date
Sat Jun 7 19:20:18 UTC 2003
/ # strace date
...
open("/usr/lib/locale/locale-archive", O_RDONLY|O_LARGEFILE) = -1 ENOENT
(No such file or directory)
...
```

Ekranınız binlerce satırla dolabilir, ama o satırlar arasından yukarıdaki gibi bir tanesi çok işimize yarayacak. Görüldüğü gibi **date** programı çalışma esnasında `/usr/lib/locale/locale-archive` dosyasını okumaya çalışıyor ancak bizde öyle bir dosya olmadığı için okuyamıyor. Bu örneği benzeri uygulamalar için de sıkıştığımız zaman tekrar edebiliriz.

Yukarıdaki örnekte bizde eksik olduğunu gördüğümüz dosyaya gerçekten ihtiyaç vardır. Bu dosyayı Linux sistemimizden SBC üzerine, aynı yere aktarmalıyız: `/usr/lib/locale/locale-archive`

`locale-archive` dosyasını SBC üzerine aktardıktan sonra **date** komutumuzun çıktısına tekrar bakalım:

```
/ # date
Cts Haz 7 19:26:59 UTC 2003
```

Gördüğümüz gibi gün ve ay isimleri Türkçe oldu.

Türkçe yerel ayarlamaları için yapmamız gereken en temel işlem, `/usr/share/locale` dizin yapısını SBC üzerinde oluşturmaktır. Bunun için aşağıdaki dosyaları SBC üzerine aktarmalıyız:

```
/usr/share/locale/locale.alias
/usr/share/locale/tr
/usr/share/locale/tr/LC_MESSAGES
/usr/share/locale/tr/LC_MESSAGES/libc6.mo
```

Son adım olarak karakter çevirimleri için aşağıdaki dosyalar da aynı dizinlerle SBC üzerine aktarılmalıdır:

```
/usr/lib/gconv/gconv-modules
/usr/lib/gconv/ISO8859-9.so
```

Dosyaların tamamı SBC üzerine aktarıldıktan sonra artık `LANG` değişkenimizin değerini `tr_TR` olarak atayabiliriz. Hemen bu işlemi yapıp yerellerin çalışıp çalışmadığını test edelim:

```
/ # export LANG=tr_TR
/ # ls /yokboylebirdosya
ls: /yokboylebirdosya: Böyle bir dosya ya da dizin yok
```

Yaptığımız ayarlar işe yarıyor, *No such file or directory* mesajı yerine *libc6.mo* içerisinden alınan *Böyle bir dosya ya da izin yok* mesajı görüntülenmektedir.

Her sistem açılışında yerel ayarlarının Türkçe olması için `/etc/init.d/rcS` içerisine

```
export LANG=tr_TR
```

satırını ekleyebiliriz.

11.2. Konsolda Türkçe Desteği

Konsolda Türkçe yazıtlarının görüntülenmesi ve Türkçe klavye kullanımı için bir miktar uğraşmamız gerekiyor. Bu işlemi yapabilmek için standart bir yöntem mevcut değil. Mümkün olduğu kadar hızlı bir şekilde Türkçe desteğini verebilmek için işlemlerimize başlayalım.

Öncelikle Türkçe klavye ayarlamalarından bahsedelim. Bunun için Linux yüklü sisteminizden `trq.kmap.gz` dosyasını kopyalamanız gerekiyor. Bu dosya genellikle `/usr/share/keymaps/i386/qwerty` dizini altında bulunur. Dosyayı buradan alıp SBC üzerinde `/usr/share/keymaps` adında bir izin yaratıp bu raya atalım (*i386 .. gibi alt izinleri de yaratmamıza gerek yok*). Test için başka bir izin içerisine, bu sıkıştırılmış `trq.kmap.gz` dosyasını açarak dosyanın içeriğine bir metin düzenleyici ile bakın. Eğer `include euro` gibi satırlar varsa bu durumda `include` edilen bu dosyaları da taşımamız gerekiyor. Bendeki sistemde *euro*, *linux-keys-bare* ve *linux-with-alt-and-altgr* için `include` tanımları mevcut. Bu nedenle aşağıdaki dosyaları Linux yüklü sistemimdeki `/usr/share/keymaps/include` dizini altından kopyalayıp SBC üzerindeki `/usr/share/keymaps` dizini altına atıyorum.

```
/usr/share/keymaps/include/euro.inc.gz
/usr/share/keymaps/include/linux-keys-bare.inc.gz
/usr/share/keymaps/include/linux-with-alt-and-altgr.inc.gz
```

Eğer bu belgedeki örnek üzerinden gidiyor ve dosya sistemi olarak *minix* seçmiş iseniz `linux-with-alt-and-altgr.inc.gz` dosyasını kopyalarken *File name too long* şeklinde bir hata almış olmalısınız :) Bu hata mesajını almanız normaldir çünkü *minix* dosya sisteminde bir dosya ismi maksimum 30 karakter uzunluğunda olabilir. Bu tamamen dosya sisteminde kaynaklanan bir sıkıntıdır. Eğer *minix* yerine *ext2* kullanıyor olsaydık böyle bir sorunumuz kalmayacaktı.



Bilgi

Aslında ilk geliştirilen *minix* dosya sisteminde dosya ismi uzunluğu maksimum 14 karakterdir. Burada bahsedilen *minix* versiyon 2'dir ve standart haline gelmiştir. Versiyon 2 bir takım iyileştirmeler içermektedir ve dosya adlarının 30 karaktere çıkması bunların başında gelmektedir.

Şimdi ne yapacağız? Elbette bu sorunun üstesinden gelmek için pek çok farklı yöntem mevcut. Ancak biz `trq.kmap.gz` dosyası içerisinden `include` satırını ve dosya ismini değiştirmek yerine sadece uzun olan dosyayı açarak `.gz` uzantısından kurtaracağız. Bu sayede dosya ismi 28 karaktere incek ve `linux-with-alt-and-altgr.inc` şekline alacaktır. Klavye kodlarını yükleyen programımız dosyanın hem sıkıştırılmış hem de açık haline baktığı için sorun olmayacaktır. Dosyayı önce Linux yüklü sisteminizde **gunzip** ile açtıktan sonra disket, ağ veya seri port üzerinden SBC'ye transfer edebilirsiniz.

Gerekli klavye dosyalarını attıktan sonra şimdi bunları sisteme yükleyebilmek için gerekli olan **loadkeys** uygulamasını SBC sistemimiz üzerine taşıyalım. Bunun için öncelikle Linux yüklü sistemimizde aşağıdaki komutla uygulamanın ihtiyaç duyduğu kütüphaneleri öğrenelim:

```
laptop:~/docbook/embedded$ ldd /bin/loadkeys
libcfont.so.0 => /lib/libcfont.so.0 (0x4001b000)
```

```
libctutils.so.0 => /lib/libctutils.so.0 (0x40020000)
libconsole.so.0 => /lib/libconsole.so.0 (0x40025000)
libc.so.6 => /lib/libc.so.6 (0x40035000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Buradaki `libcfont.so.0`, `libctutils.so.0` ve `libconsole.so.0` kütüphaneleri SBC sistemimiz üzerinde henüz mevcut değil ve bu nedenle onlarında taşınması gerekiyor. Gerekli dosyaları taşıdıktan ve **loadkeys** uygulamasının çalıştırma yetkisine sahip olduğundan emin olduktan sonra klavyemiz için gerekli kodların aşağıdaki komutla yüklenmesini sağlayabiliriz:

```
/ # loadkeys /usr/share/keymaps/trq.kmap.gz
Loading /usr/share/keymaps/trq.kmap.gz
```

Artık klavyemiz hazır.

Klavye ile işimiz bittikten sonra sırada Türkçe konsol yazıtıtipini ayarlamak var. Ayrıntıya girmeden doğrudan nasıl yapacağınızı anlatmaya çalışacağım, ancak her zaman olduğu gibi kullandığımız uygulamaların kılavuz sayfalarına (*manual*) vakit buldukça bakmanızda fayda var.

Öncelikle gerekli font dosyalarını SBC üzerine aktaralım. Bunun için öncelikle SBC üzerinde `/usr/share/consolefonts` ve `/usr/share/consoletrans` dizinlerini oluşturmalıyız. Ardından Linux sisteminizdeki `/usr/share/consoletrans/iso09.acm.gz` ve `/usr/share/consolefonts/iso9-16.psf.gz` dosyalarını SBC üzerindeki aynı dizinlere kopyalayınız.

Yazıtıtiplerini sisteme taşıdıktan sonra **consolechars** uygulamasını da Linux yüklü sistemimizden taşımamız gerekiyor. İhtiyaç duyulan kütüphaneleri kontrol edip uygulamayı SBC üzerindeki `/bin` dizini altına kopyalamalı ve çalıştırma hakları yoksa vermeliyiz. Şimdi **<Alt – F1>** ile girilen birinci sanal konsolda çalışıyorsanız Türkçe fontları görebilmek için aşağıdaki komutları girmelisiniz:

```
/ # consolechars -f iso9-16
/ # consolechars --tty=/dev/tty1 -f iso9-16 -m iso09
/ # echo -en "\033(K" > /dev/tty1
/ # echo -en "\017" > /dev/tty1
```

Şimdi içerisinde bolca Türkçe karakterler olan bir dosyayı ağ, disket veya seri port üzerinden SBC üzerine aktararak **cat dosya_adi** komutuyla karakterlerin ekranda düzgün olarak görünüp görünmediğini test edin. Eğer bir problem varsa işlem adımlarını tekrarlayın (özellikle yukarıdaki komutları dikkatlice yazmayı deneyin).

Buraya kadar ki adımları sorunsuz bir şekilde atlattı iseniz şimdi de her sistem açılışında otomatik olarak Türkçe ayarlamalarının yapılmasını sağlayalım. Dikkat ederseniz yazıtıtipi ayarlamalarını sanal konsolların sadece biri için yaptık. Sistemimizdeki ilk dört konsol için Türkçe yazıtıtipi ayarlamalarının yapılmasını istiyorsak `/etc/config` dosyamızın sonuna aşağıdaki gibi bir satır ekleyelim:

```
TR_KONSOL="tty1 tty2 tty3 tty4"
```

Ardından `/etc/init.d/rcS` kabuk programımızın sonuna aşağıdaki satırları ekleyelim:

```
echo "Klavye ve fontlar ayarlanıyor"
loadkeys /usr/share/keymaps/trq.kmap.gz

consolechars -f iso9-16

for tty in ${TR_KONSOL}
do
    consolechars --tty=/dev/$tty -f iso9-16 -m iso09
```

```
echo -en "\033(K" > /dev/$tty
echo -en "\017" > /dev/$tty
done
```

Bu sayede her açılışta ilk dört konsol için gerekli font ayarlamaları da yapılmış olacaktır. Bu sayıyı artırmak veya azaltmak isterseniz tek yapmanız gereken `/etc/config` dosyanızdaki `TR_KONSOL` değişkeninin değerine ilgili `tty` adlarını eklemek veya çıkarmak olacaktır.



Bilgi

Konsolda Türkçe problemi burada anlattıklarımızla çözümlüyor olmasına rağmen aynı şeyi BusyBox içerisinde çıkan programlar için söylemek o kadar kolay değil. Busybox ile birlikte gelen kabuk olan `ash` ve gene Busybox içinden çıkan `vi` editörü ile Türkçe konusunda başarılı olamadım. Ancak Linux yüklü bir sistemden kopyaladığım `bash` kabuğu ve `vi` metin düzenleyici ile Türkçe karakterleri görüp yazabildim. Eğer aynı işi BusyBox içerisindeki araçlarla yapabilir ve eposta ile beni de bilgilendirirseniz çok sevinirim.

11.3. X Window System Türkçe Desteği

Dış Bağlantılar

Belge içinde belirtilmiş dış bağlantılar varsa bunların adresleri başvuru numaralarıyla burada listelenmiş olacaktır.

1

[../howto/kernel-nasil.pdf](#)