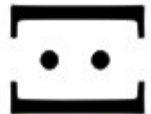
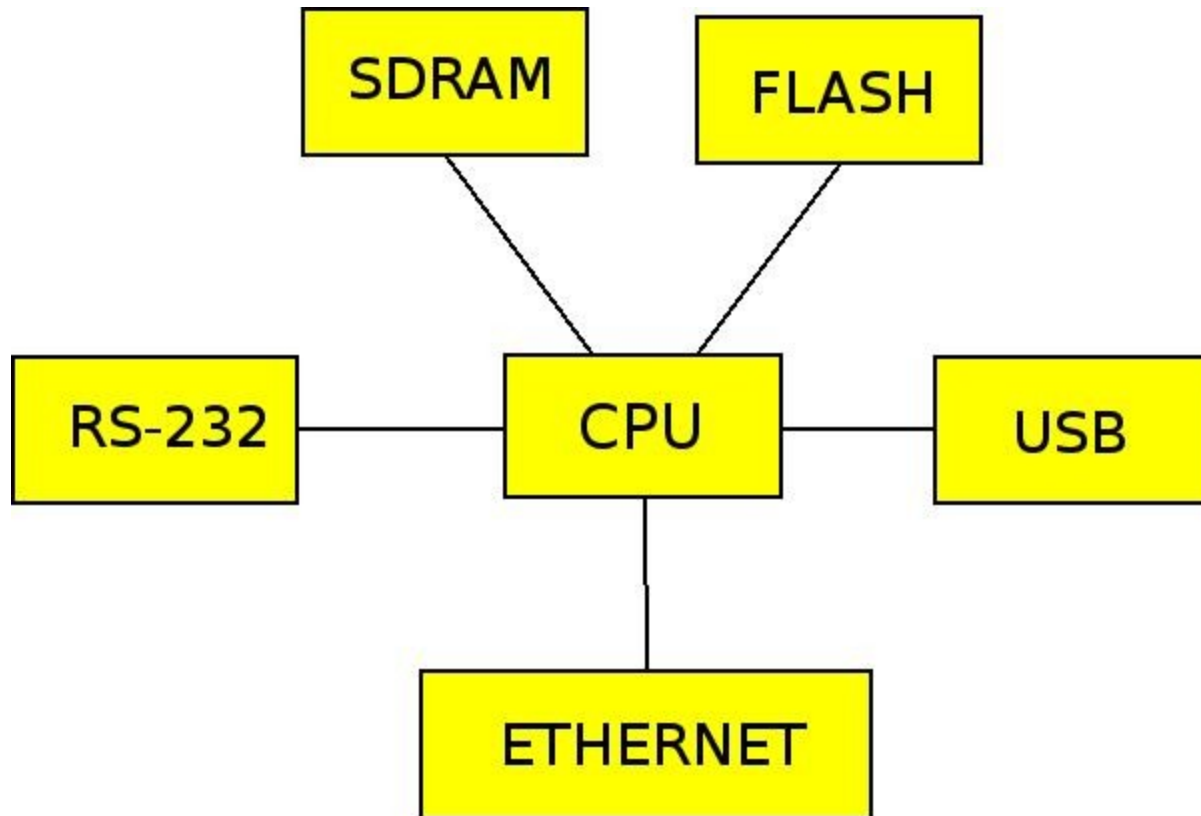


# İçerik

- Çapraz-geliştirme yöntemleri ve araçları
- Önyükleyiciler
- C kütüphaneleri
- Linux Çekirdeği
- Linux sürücüleri



# Örnek Gömülü Sistem



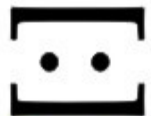
# Gömülü Sistemlerde Linux Kullanımı

- İşlemcimiz 32/64 bit mimaride mi?
- MMU var mı?
- ağ-tabanlı bir iş yapılacak mı?
- Gerçekleştirmesi zor, ve karmaşık arayüzler mi kullanacaksınız?
  - bluetooth
  - USB



# Gömülü Sistemlerde Linux Kullanımı

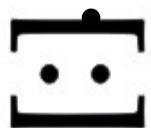
- Avantajlar:
  - Üstün donanım desteği
  - Açık-kaynak topluluğu
  - Topluluk ve şirket desteği (e-posta grupları, MontaVista software, ...)
- Dezavantajlar:
  - GPL ile ilgili korkular
  - sıkı gerçek-zamanlı uygulamalar



# Gömülü Sistemler Üzerinde Geliştirme

- Anasistem(Host): Geliştirmenin üzerinde yapıldığı sistem
  - örn: Masaüstü PC'niz
- Hedef(Target): Üretilen kodun üzerinde çalışacağı sistem
  - örn: Üzerinde ARM işlemcisi olan, Linux çalıştıracak olan yeni tasarladığınız ekmek kızartma makinasınız ;)

Örnek: x86 -> ppc8260, sparc -> omap5912



# Örnek Geliştirme Ortamı

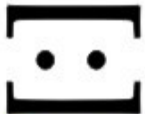


# Anasistem

Ethernet, IEEE802.11, USB



## Hedef sistem



# Çapraz Geliştirme Araç Zinciri

- Hedef ortamımızı kullanılabılır kılmak için:
  - binutils
  - gcc
  - c kütüphanesi (glibc | uclibc | newlib | ...)
  - önyükleyici (u-boot | grub | redboot | ...)
  - Kök dosya sistemi
  - Linux Çekirdeği



# Çapraz Geliştirme Araç Zinciri

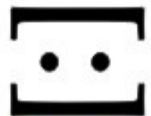
- Kendine eziyet etmeyi sevenler:
  - binutils
  - gcc
  - c kütüphanesi(uclibc, libc, ...)
  - Linux çekirdek başlıkları
- İşlerin aşağıda nasıl yürüdüğünü anlamak için iyi bir deneyim
- Biraz zahmetli





# Binutils

- ld: linker
- as: assembler
- objcopy: nesne dosyalarını kopyalamak ve tercüme etmek için
- objdump: nesne dosyalarıyla ilgili bilgileri görmek için(disassemble, relocation)
- readelf: elf formatındaki dosyalarla ilgili bilgileri okumak için



# Binutils

- addr2line, ar, nlmconv, nm, size, strip, ranlib, gprof, c++filt
- Yapılandırma: ./configure
  - --prefix
  - --host
  - --target
  - --enable-languages



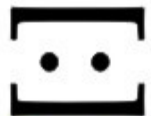
# Binutils

- --with-cpu
- --with-float
- Örnek yapılandırma:
  - `configure --prefix=/path/to/src/binutils-2.x/./configure --build=i386-linux-gnu --host=i386-linux-gnu --target=arm-linux-uclibcgnueabi --enable-languages=c --with-gnu-ld --with-float=soft --with-cpu=xscale --with-arch=armv5te`
- `make; make install`



# gcc

- binutils derledikten sonra sıra bootstrap derleyicimizde
- C'den başka bir dil için gcc derlemek istiyorsak, önce hedef platform için C derleyebilen bir derleyici ve C kütüphanesi derlememiz gerekiyor
- Ondan sonra bu C kütüphanesini kullanarak diğer diller için gcc derleyebiliriz

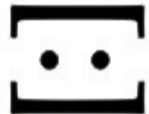
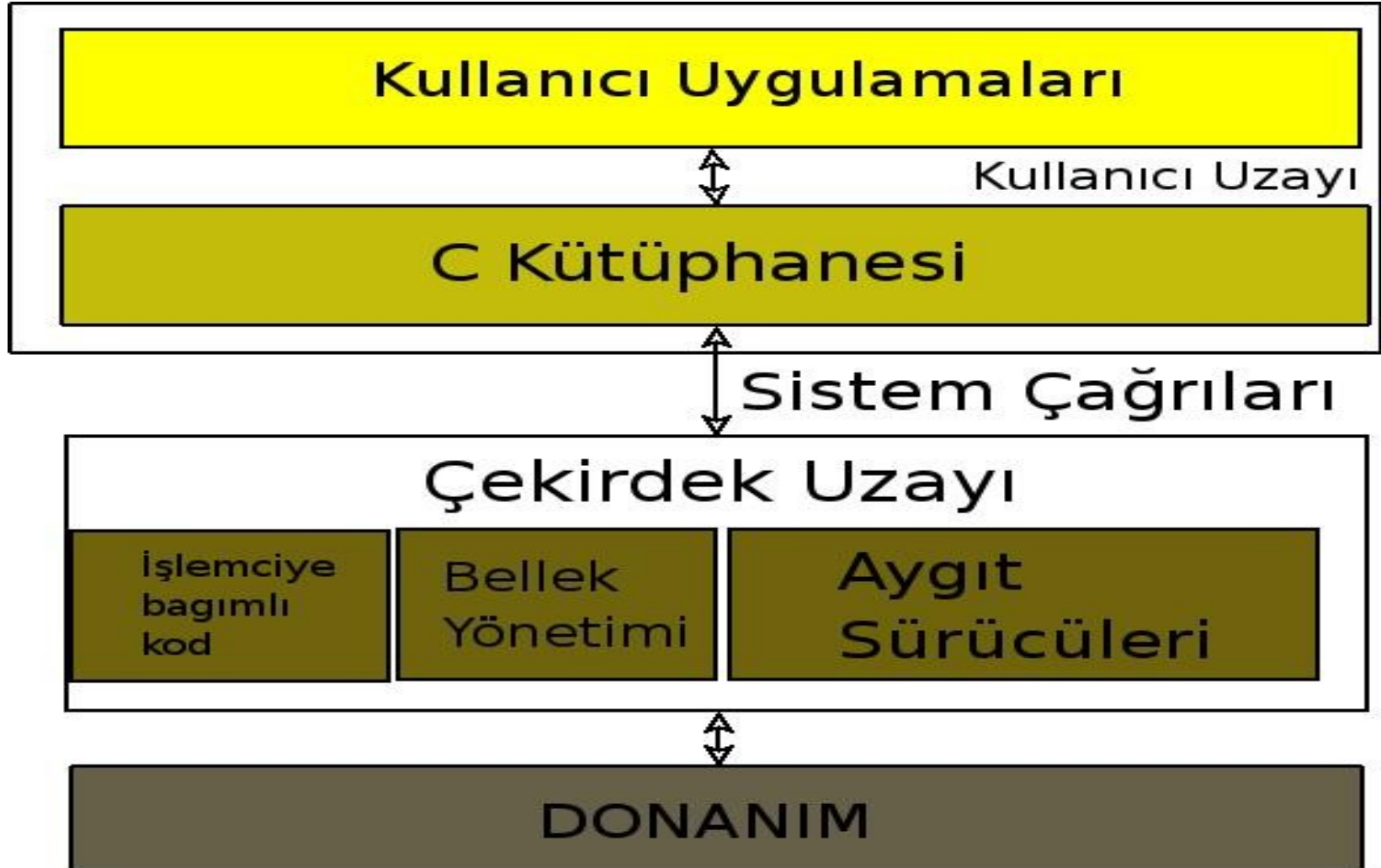


# gcc yapılandırması

- Yapılandırma için binutils'e verdiğiniz parameterlerle aynı parametlerini kullanın
- `make; make install`

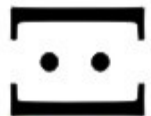


# Kullanıcı-Çekirdek-Donanım İlişkisi



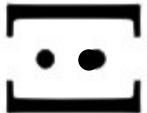
# C Kütüphaneleri

- uClibc
  - Küçük boyutlu, glibc uyumlu
- glibc
  - Standart gnu kütüphanesi
- dietlibc
  - Sadece statik bağlanmış dosyalar yaratabiliyor
- newlib
  - Red Hat tarafından desteklenen alternatif



# uClibc

- uClinux projesi kapsamında yazılmaya başlandı.
- Neredeyse tamamen glibc uyumlu
- Dinamik bağlama desteği
- Thread desteği
- libuClibc-0.9.28.so (arm için derlenmiş)
  - 305k





# uClibc

- Araç zincirimiz hazır ise
- ncurses tabanlı yapılandırma:
  - make menuconfig
    - Hedef mimariyle ilgili seçenekler
    - kütüphaneyle ilgili seçenekler(POSIX thread'leri, malloc implementationı)
    - ağ desteği(rpc, ipv6)
    - geliştirme ortamının yer
- make; make install



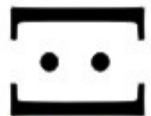
# Çapraz Geliştirme Araç Zinciri

- Daha insancıl yöntemler:
  - Buildroot (<http://buildroot.uclibc.org>)
  - Scratchbox (<http://www.scratchbox.org>)
  - ELDK(<http://www.denx.de/wiki/DULG/ELDK>)
  - crosstool(<http://www.kegel.com/crosstool/>)
  - Hazır derlenmiş başka araç-zincirleri(CodeSourcery, vs...)



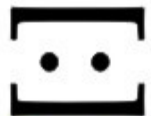
# Buildroot

- Makefile'lar ve yamalardan oluşuyor
- kök dosya sistemi otomasyonu
- Derledikleri:
  - busybox
  - linux çekirdeği
  - binutils
  - gcc
  - uClibc

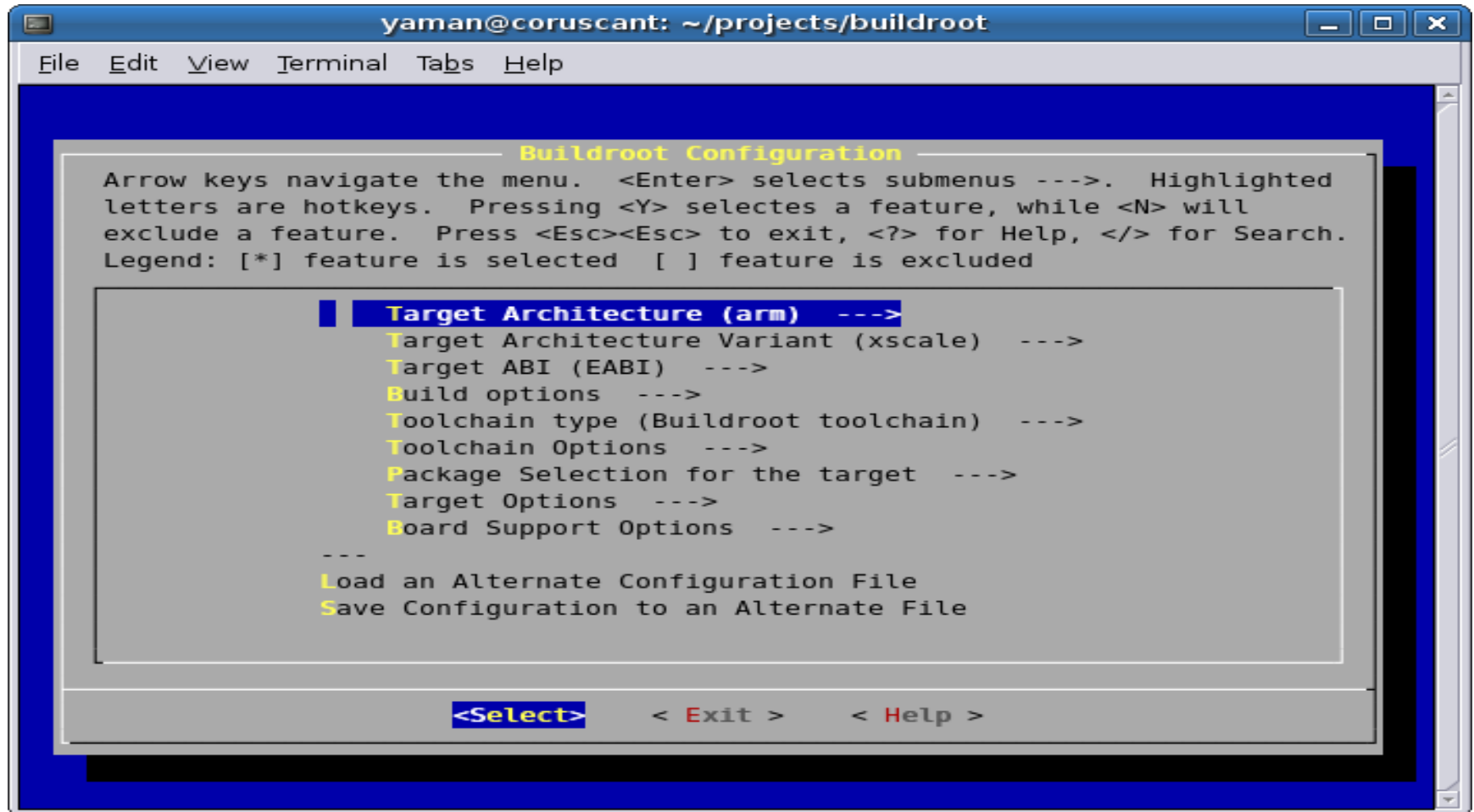


# Buildroot – Nasıl Çalışır?

- package
  - buildroot'un derleyip hedef kök dosya sistemine atabileceği araçlar ile ilgili Makefile'ları ve yamaları içerir
- toolchain
  - binutils, gcc, uclibc, ccache, gdb, kernel
- target
  - kök dosya sistemi ve önyükleyiciyi derlemek için gerekli Makefile'lar ve yamalar



# Buildroot



The screenshot shows a terminal window titled 'yaman@coruscant: ~/projects/buildroot'. The menu is titled 'Buildroot Configuration' and provides instructions on how to navigate using arrow keys, select submenus with Enter, and use hotkeys Y, N, Esc, and /. It also includes a legend for feature selection. The menu items are: Target Architecture (arm) (highlighted), Target Architecture Variant (xscale), Target ABI (EABI), Build options, Toolchain type (Buildroot toolchain), Toolchain Options, Package Selection for the target, Target Options, and Board Support Options. At the bottom, there are options to load or save alternate configuration files and navigation buttons: <Select>, <Exit>, and <Help>.

```
yaman@coruscant: ~/projects/buildroot
File Edit View Terminal Tabs Help

Buildroot Configuration

Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted
letters are hotkeys. Pressing <Y> selects a feature, while <N> will
exclude a feature. Press <Esc><Esc> to exit, <?> for Help, </> for Search.
Legend: [*] feature is selected [ ] feature is excluded

[*] Target Architecture (arm) --->
    Target Architecture Variant (xscale) --->
    Target ABI (EABI) --->
    Build options --->
    Toolchain type (Buildroot toolchain) --->
    Toolchain Options --->
    Package Selection for the target --->
    Target Options --->
    Board Support Options --->
    ---
    Load an Alternate Configuration File
    Save Configuration to an Alternate File

    <Select>    <Exit>    <Help>
```



# Buildroot

- `svn co svn://uclibc.org/trunk/buildroot`
- `make menuconfig`
- `make; make install`



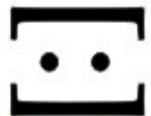
# Busybox

- çoğu unix aracının sadeleştirilmiş hallerini içerir
- Linux tabanlı gömülü sistemlerde yaygın olarak kullanılmakta olan bir paket
- Kolay yapılandırma, ve derleme.
- uclibc ile statik olarak link edildiğinde 450-500k civarı bir dosya boyutu.



# Busybox

- standart unix araçları
  - ls, cp, cat, rm, chmod, ... (~70 araç)
- ağ araçları
  - traceroute, nslookup, ifconfig, httpd, ... (~30 araç)
- arama araçları
  - find, grep
- arşiv araçları(rpm, dpkg, bunzip2),  
editorler(vi), dosya sistemi araçları(fsck), ...





# Busybox'ı Derleyip Çalıştırmak

- Derleme kısmı gene çok kolay
  - make menuconfig
  - make; make [PREFIX= /kök/fs/] install
- Busybox içinde derlediğiniz bir programı çağırarak için:
  - busybox ls
  - veya kurulum sırasında yarattığı sembolik link ile: ls



# FLASH hafıza yerleşimi



- İşlemci kontrolü önyükleyiciye verir
- Önyükleyici donanımı hazırlar
- Kontrolü çekirdeğe verir



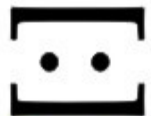
# Önyükleyiciler

- U-boot (<http://u-boot.sourceforge.net/>)
- Redboot (<http://www.cygwin.com/redboot/>)
- Lilo
- GRUB
- LAB(Linux As Bootloader)



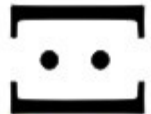
# Önyükleyicinin işlevi

- İşlemci çalışmaya başladığı anda denetimi ele alır
- BIOS görevi görür
- Bellek ilklendirmesi
- I/O denetçisi ilklendirmesi
- Grafik donanımı ilklendirmesi



# Önyükleyici seçerken

- Kullandığım mimariyi destekliyor mu?
- Arkasında topluluk desteği var mı?
- Yaygın olarak kullanılıyor mu?
- Kendi kullanacağım sisteme benzer bir sisteme taşınmış mı?
- İstediğim özelliklere sahip mi?



# Redboot

- eCos'un hal (Donanım soyutlama katmanı) üstüne yazılmış
- Ethernet ve seri port üzerinden program yüklenebiliyor
- TFTP üzerinden dosya sistemi imajı yüklenebiliyor
- ARM, MIPS, PPC, IA32 destekliyor.



# LAB

- Linux çekirdeğinin gereken kısımları alınarak oluşturulmuş
- ARM mimarisini destekliyor
- Dosya, MTD sistemleriyle alakalı kodu kullanıyor.
- <http://handhelds.org/cgi-bin/cvsweb.cgi/linux/kernel26/lab/modules/>



# U-Boot

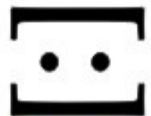
- Gömülü sistemlerde neredeyse standartlaşmış bir ürün
- Kolay taşınabilirlik
- Kullanan sayısı yüksek, destek bulması rahat
- MIPS, ARM, ppc, x86, SPARC
- TFTP/BootP desteği





# U-Boot'u Taşımak

- Önemli klasörler:
  - ../u-boot-1.1.x/cpu
    - işlemciye bağımlı kaynak kodu tutulur.
    - işlemcinizi destekleyip desteklemediğine buradan bakabilirsiniz.
  - ../u-boot-1.1.x/board
    - geliştirme boardlarıyla alakalı dosyalar
  - Kullanacağınız kartı tanıyorsa derlemesi make kart\_config; make demek kadar kolay



# U-Boot'u Taşımak

- İşlemcinizi destekliyorsa işiniz son derece basit
- Yapmanız gereken kartınızla aynı işlemciyi kullanan benzer bir kart yapılandırma dosyası bulmak(..../include/configs/ altında) ve o dosyayı kendi kartınıza uygun bir şekilde özelleştirmek



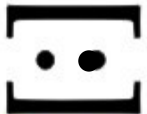
# U-Boot'u taşımak

- Daha sonra ../board klasöründe kendi kartınız için bir klasör yaratıp, diğer kartın dosyalarını buraya atıp özelleştirin
- Derleyin ve herşey yolunda gittiyse güle güle kullanın
- <http://www.phptr.com/articles/article.asp?p=674698&seqNum=4&rl=1>



# Linux Çekirdeği

- Son sürüm: 2.6.21
- 25 tane işlemci platformu desteği
  - \$LNXSRC/arch
- ~23000 dosya
- C'de yazılmış
- C kütüphanesi yok



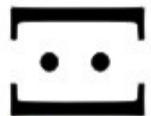
# Linux Çekirdeği

- alpha
- arm
- arm26
- avr32
- cris
- frv
- h8300
- i386
- ia64
- m32r
- m68k
- mips
- parisc
- powerpc
- ppc
- s390
- sh
- sh64
- sparc
- sparc64
- um
- v850
- x86\_64
- xtensa



# Çekirdek yapılandırma

- Linux çekirdeğini yapılandırmak için:
  - `make ARCH=<arch> [config|menuconfig|xconfig]`
  - İşlemci mimarisi
  - Gerekli arayüzleri seç(USB, i2c, ...)
  - Process Scheduling
  - I/O Scheduling
  - Ağ desteği
  - Dosya Sistemi



# I/O Scheduling

- I/O optimizasyonu için kullanılır
- Disk erişimi bilgisayarın gerçekleştirdiği en ağır işlemlerden biri
- Anticipatory
- Deadline
- CFQ(Completely-Fair Queueing)
- NO-OP



# Çekirdeği derleme

- make ARCH=<arch>  
CROSS\_COMPILE=<arch>-linux-
- make modules\_install  
INSTALL\_MOD\_PATH=<kökdosyasistemi>



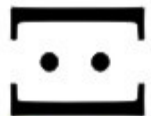


# Çekirdek Geliştirme Araçları

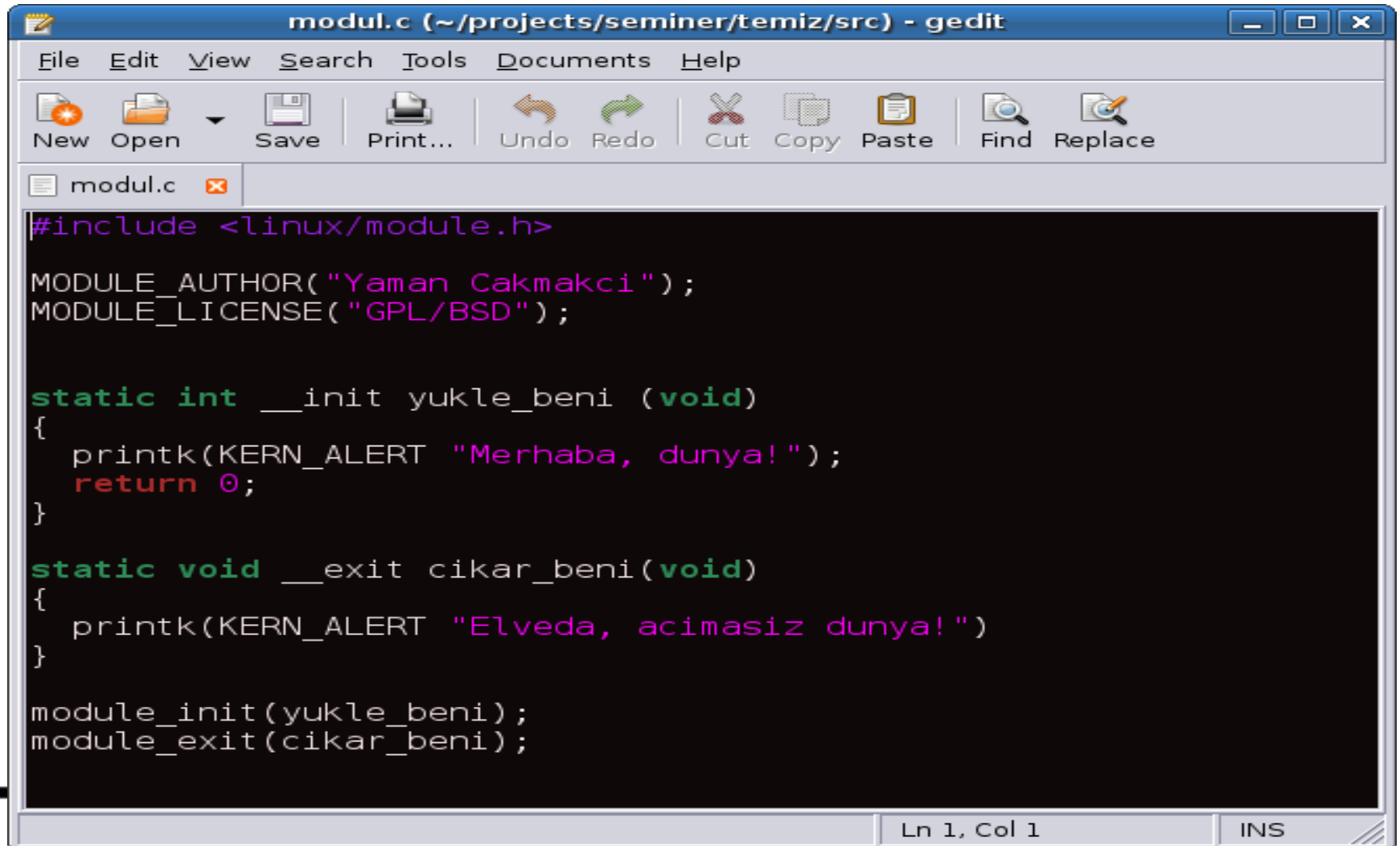
- ketchup: Çekirdek kaynak kodu indirmek ve güncellemek için
- quilt: Yama yönetimi
  - quilt push
  - quilt pop <patch>
- PatchSet: quilt'e alternatif
- Kscope: Kod okumak/analiz etmek için
- etags/ctags: kod içinde gezinmek için

# Linux Modülleri

- Linux çekirdeği monolitik yapıdadır
- Çekirdeğe dinamik olarak eklemeler yapmak için modüller kullanılır
- Sürücü geliştirirken büyük bir avantaj
- Her değişiklik sistemi baştan başlatmayı gerektirmez
- Çalıştırılacak kod çekirdekte çalışacağı için dikkatli yazılması gerekiyor



# Örnek Modül



```
modul.c (~/projects/seminer/temiz/src) - gedit
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
modul.c
#include <linux/module.h>

MODULE_AUTHOR("Yaman Cakmakci");
MODULE_LICENSE("GPL/BSD");

static int __init yukle_beni (void)
{
    printk(KERN_ALERT "Merhaba, dunya!");
    return 0;
}

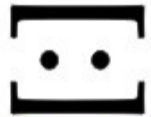
static void __exit cikar_beni(void)
{
    printk(KERN_ALERT "Elveda, acimasiz dunya!")
}

module_init(yukle_beni);
module_exit(cikar_beni);

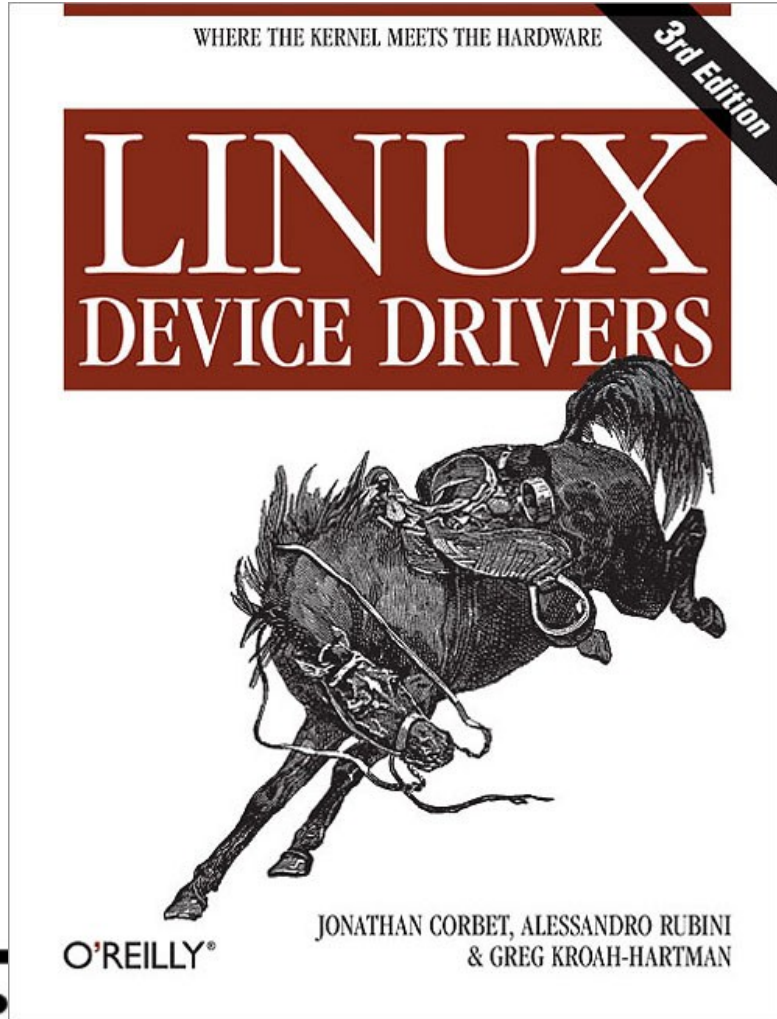
Ln 1, Col 1 INS
```

# Linux Aygıt Sürücüler

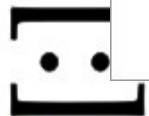
- Linux iki tür aygıt tanır
  - Karakter
    - Örn: Webcam, ekran kartı, ses kartı
  - Blok
    - Örn: hard-disk, loopback aygıtı
- Sürücüler ise üç türdür
  - Karakter
  - Blok
  - Ağ



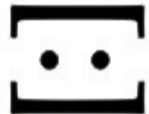
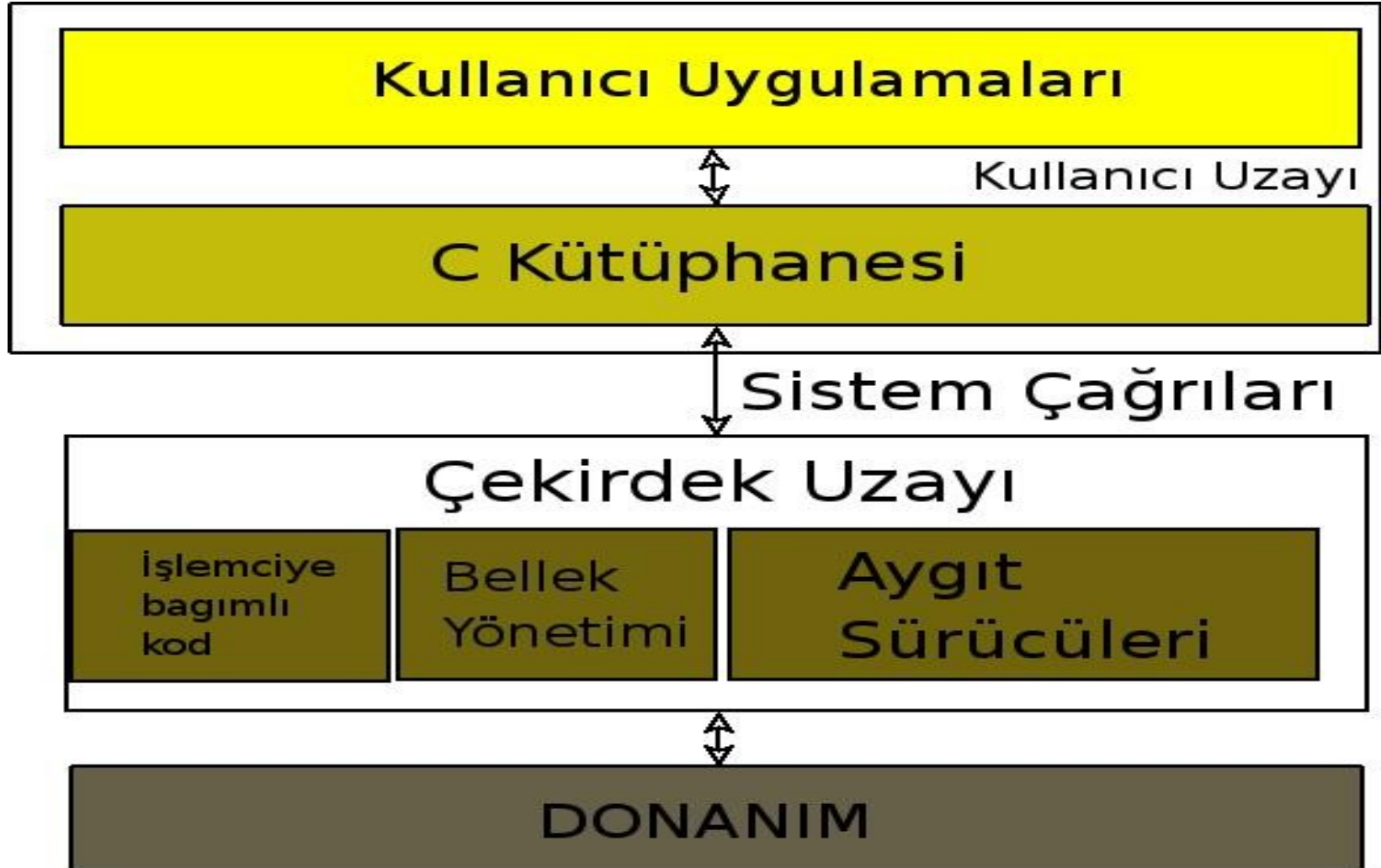
# REKLAMLAR



- Gömülü Linux veya çekirdek programlamayla uğraşacaksanız okumanız \*şart\* olan kitaplardan
- <http://lwn.net/Kernel/LDD3/>



# Kullanıcı-Çekirdek-Donanım İlişkisi



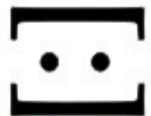
# Kök Dosya Sistemi(RootFS)

- Çekirdeğin sistem açılışında bağlayacağı '/' dizini
- Önyükleyici sonrası kullanılan bütün yazılımı içerir( Linux çekirdeği, busybox, ...)
- Sistemimizin yer sorunu olabilir, gereksiz hiçbir program bulundurmamakta fayda var



# Kök Dosya Sistemi(RootFS)

- ../bin
- ../boot
- ../dev
- ../etc
- ../lib
- ../mnt
- ../sbin
- ../usr
  - ../bin
  - ../lib
  - ../sbin





# Kök Dosya Sistemi'ni Yaratmak

- Herşeyimiz hazır ise
- ext2, ext3, reiserfs, jffs2, ...
- `mkfs.jffs2 -d ./rootfs -o rootfs.jffs2.img`



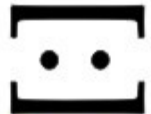
# strace

- Sistem çağrılarını takip etmek için kullanılır
- IPC, ağ, processler, sinyaller ile alakalı sistem çağrılarını filtreleyerek gösterebilirsiniz
  - Örn: `strace -e trace=network <dosya_adi>`
- Child process'leri de takip edebilirsiniz(-f)



# ltrace & mtrace

- ltrace:
  - C kütüphanesi çağrılarını takip etmek için
- mtrace:
  - bellek ile alakalı çağrıları takip etmek için
    - malloc(), realloc(), free()



# Gerçek-zaman

- Ingo Molnar'in 2.6-rt çekirdek ağacı
- RTLinux
- RTAI



# Gerçek-Zaman Yamaları

- realtime-preempt
  - <http://people.redhat.com/mingo/realtime-preempt/>
- realtime-lsm
  - <http://sourceforge.net/projects/realtime-lsm/>
- hr-timers
  - <http://high-res-timers.sourceforge.net/>

