dd, device dump, device destroy

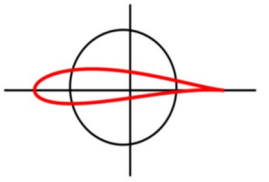
Bir dosyayı okur ve başka bir dosyaya kopyalar.

cp komutundan farklı olarak mount edilmemiş cihazları, blok blok okuyabilir.

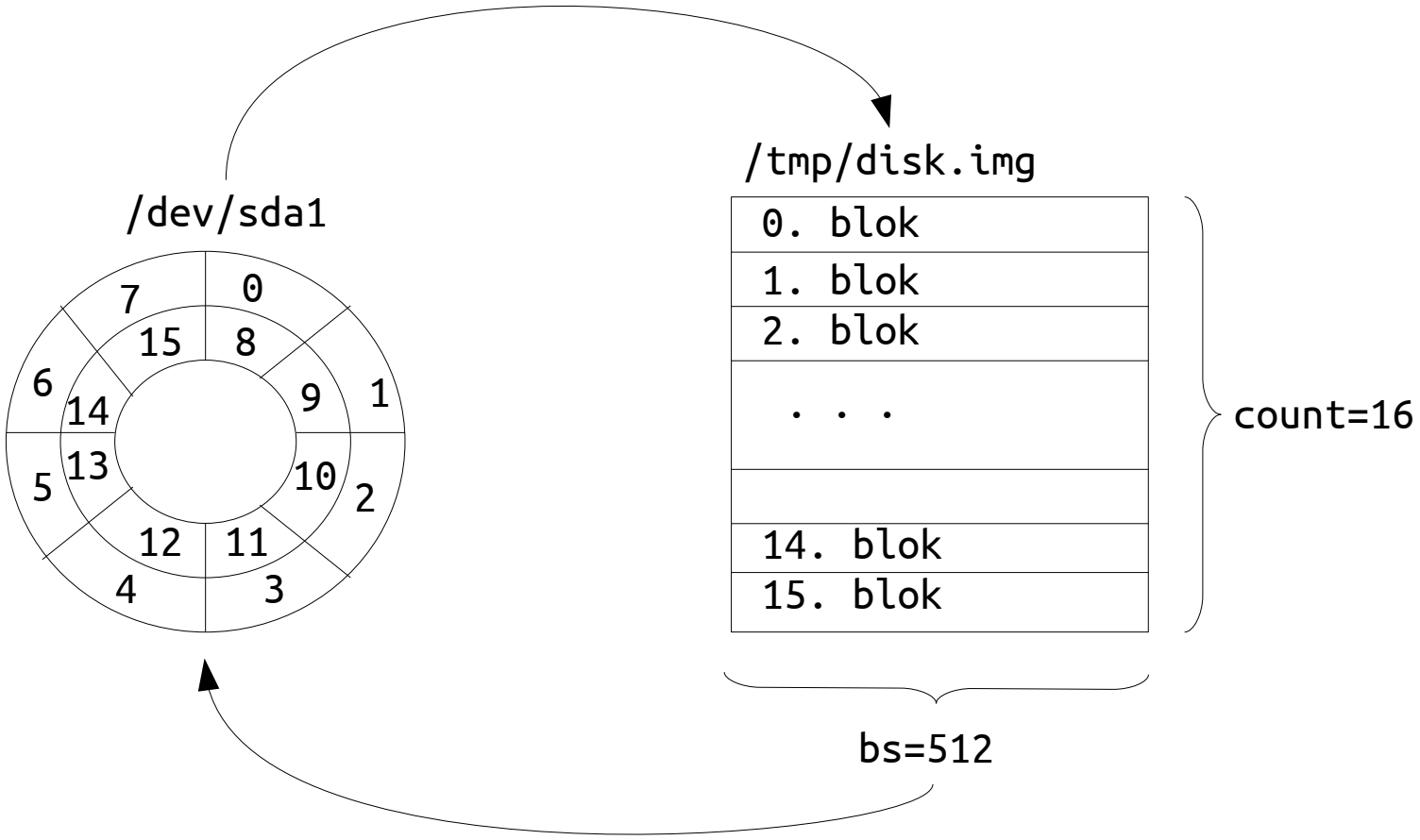
cp komutu, sadece mount edilmiş dosya hiyerarşisini takip ederek çalışabilir.

```
$ dd if=/dev/zero of=riscv_disk bs=1M count=64    # komutunun sözde kodu ...
```

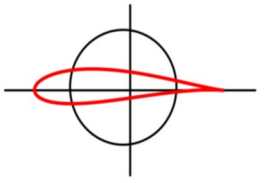
```
dd(){
    if= open("/dev/zero", O_RDONLY);    // if=/dev/zero
    of= open("riscv_disk", O_WRONLY);   // of=riscv_disk
    char bs[1024*1024];                 // bs=1M
    count= 64;                          // count=64
    for(i= 0; i< count; i++){
       n= read(if, bs, sizeof(bs));     // if'den bs[] içine n<=1M adet 0 oku
       write(of, bs, n);                // okunan sıfırları of dosyasına yaz
       }
}
```

$ dd if=/dev/sda1 of=/tmp/disk.img count=16 bs=512

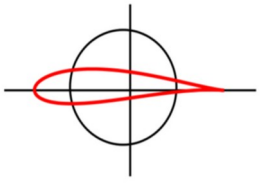


$ dd if=/tmp/disk.img of=/dev/sda1 count=16 bs=512

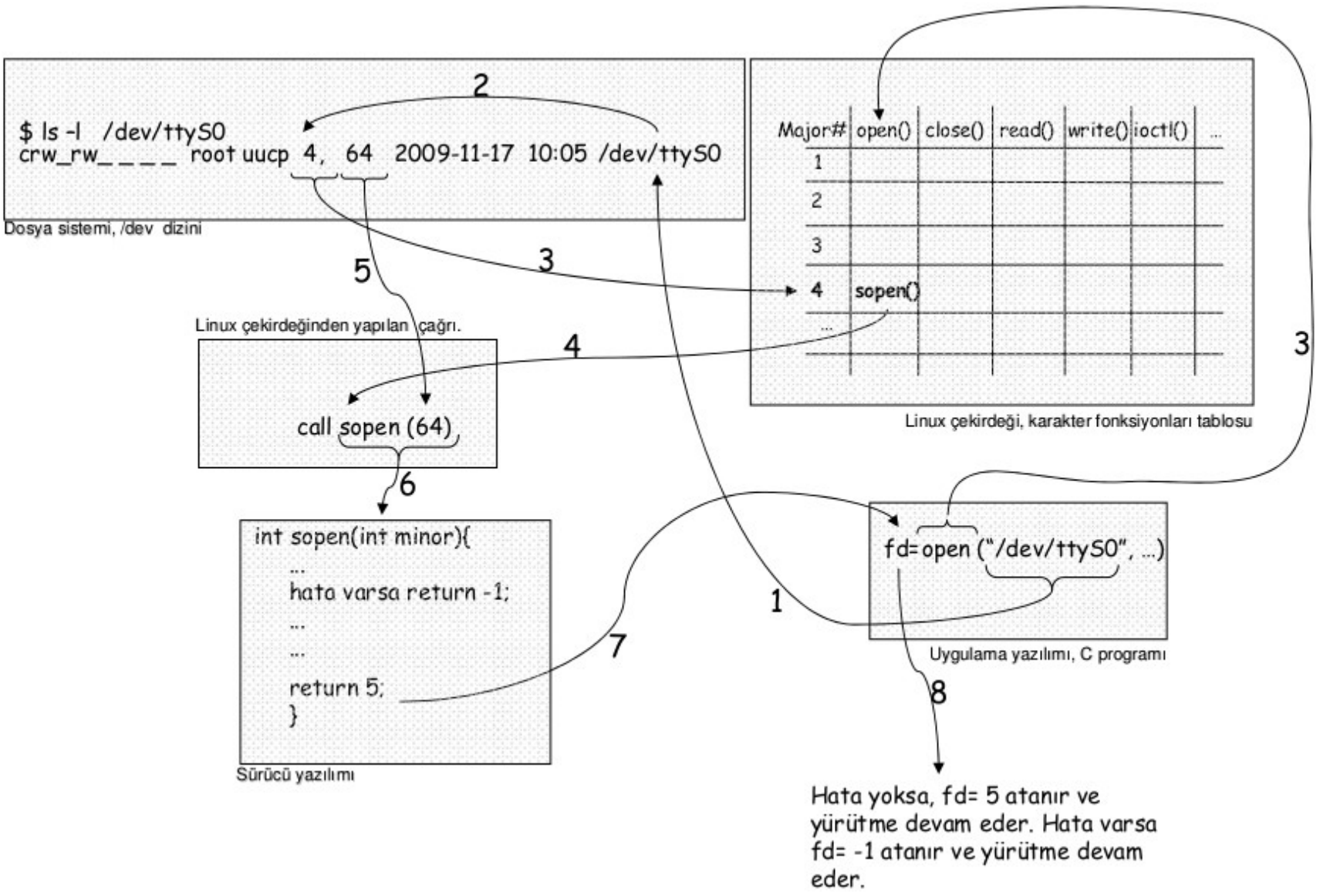


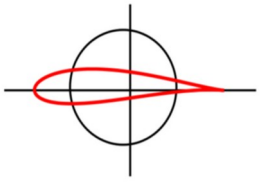
Character Device Functions Switch Table

	Major#	open()	close()	read()	write()	ioctl()	...
bellek aygıtları	1						
pseudo tty slave aygıtları	2						
pseudo tty aygıtları	3						
tty aygıtları	4	**sopen()**	**sclose()**	**sread()**	**swrite()**	**sioctl()**	
diğer aygıtlar	...						

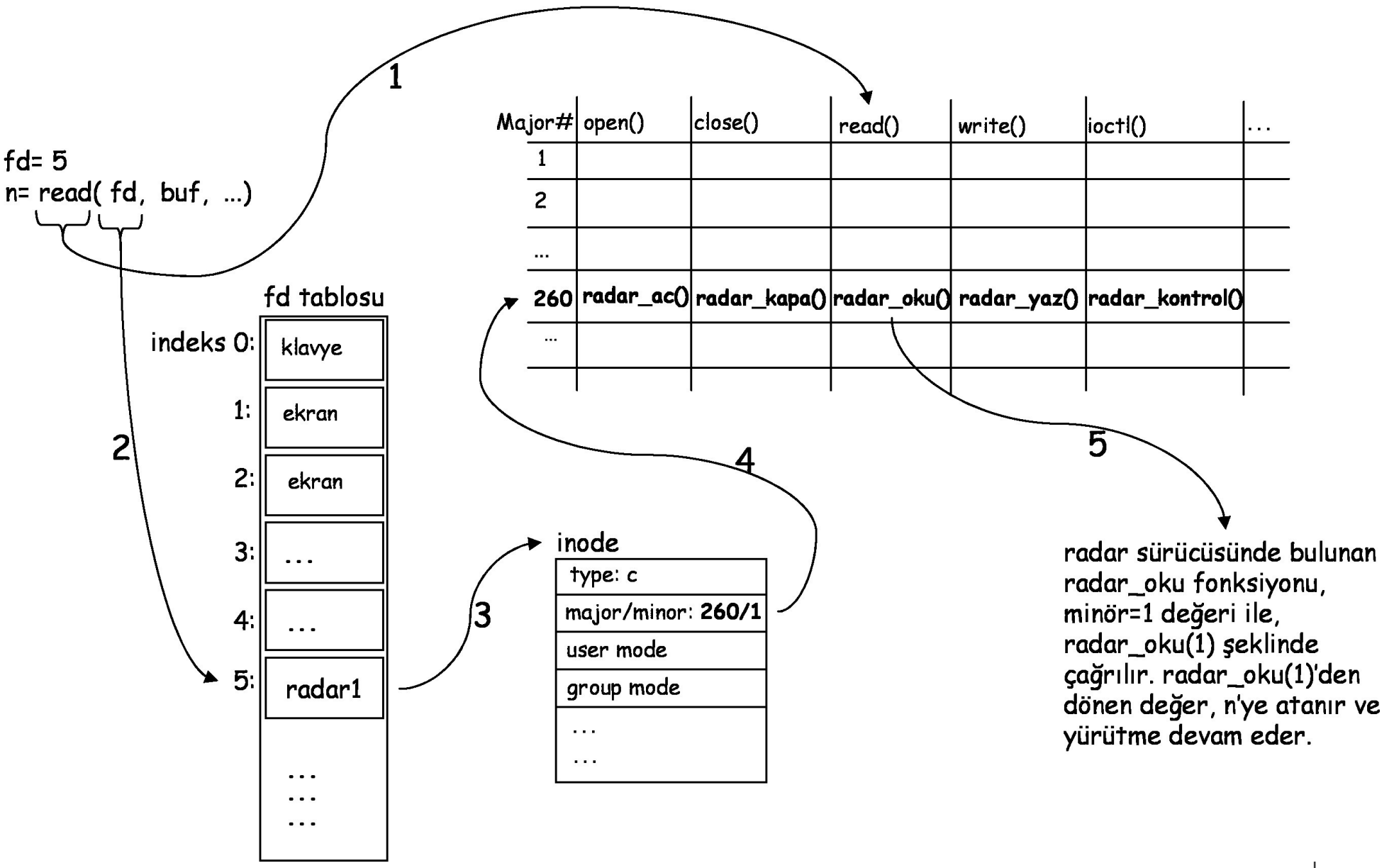


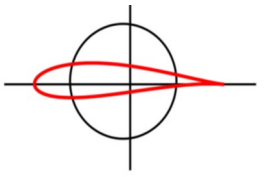
VFS Open


$ ls -l /dev/ttyS0
crw_rw____ root uucp 4, 64 2009-11-17 10:05 /dev/ttyS0
Dosya sistemi, /dev dizini
2
5
3
Linux çekirdeğinden yapılan çağrı.
call sopen (64)
4
6
int sopen(int minor){
...
hata varsa return -1;
...
...
return 5;
}
Sürücü yazılımı
7
Major# open() close() read() write() ioctl() ...
1
2
3
4 sopen()
...
Linux çekirdeği, karakter fonksiyonları tablosu
3
1
fd=open ("/dev/ttyS0", ...)
Uygulama yazılımı, C programı
8
Hata yoksa, fd= 5 atanır ve yürütme devam eder. Hata varsa fd= -1 atanır ve yürütme devam eder.

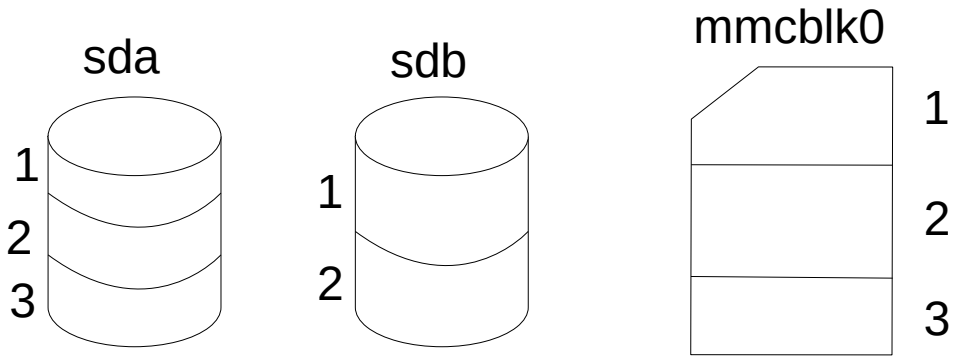


VFS Read


fd= 5
n= read(fd, buf, ...)
1
2
3
4
5
Major# open() close() read() write() ioctl() ...
1
2
...
260 radar_ac() radar_kapa() radar_oku() radar_yaz() radar_kontrol()
...
fd tablosu
indeks 0: klavye
1: ekran
2: ekran
3: ...
4: ...
5: radar1
...
...
...
inode
type: c
major/minor: 260/1
user mode
group mode
...
...
radar sürücüsünde bulunan radar_oku fonksiyonu, minör=1 değeri ile, radar_oku(1) şeklinde çağrılır. radar_oku(1)'den dönen değer, n'ye atanır ve yürütme devam eder.



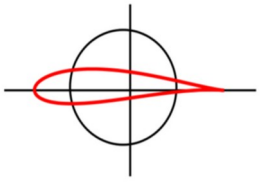
Disklerin İsimlendirilmesi



sda1
sda2
sda3

sdb1
sdb2

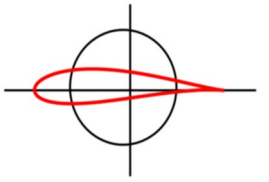
mmcblk0p1
mmcblk0p2
mmcblk0p3



/dev/null

Linux sisteminin çöp tenekesidir

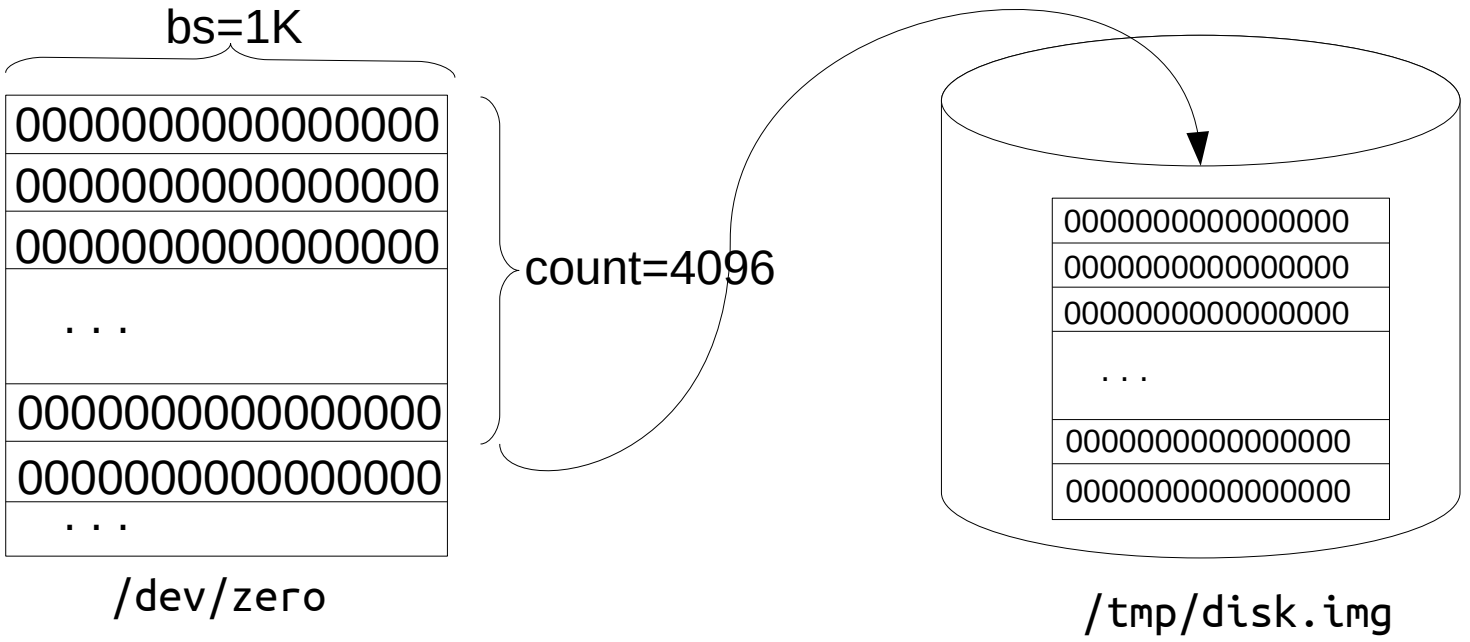
```
$ nohup prog 1>/dev/null 2>/tmp/prog.err &
$ dd if=/dev/sda of=/dev/null
```

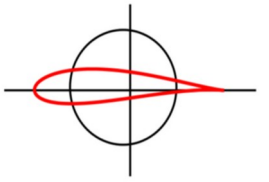


/dev/zero

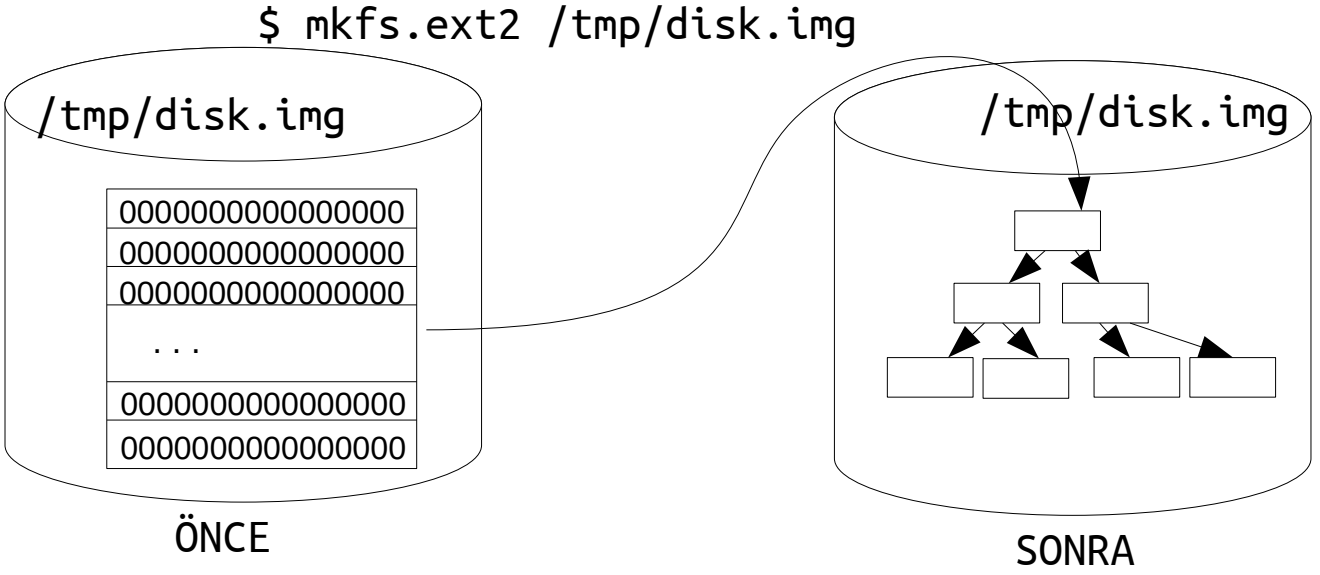
İçinde sonsuz adet “sıfır” bulunan bir dosyadır.

```
$ dd if=/dev/zero of=/tmp/disk.img bs=1K count=4096
```

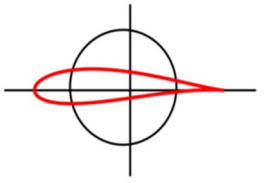




loop device




```
$ mkfs.ext2 /tmp/disk.img
$ mkdir /mnt/sanal.disk
$ mount -o loop /tmp/disk.img /mnt/sanal.disk
...
$ umount /mnt/sanal.disk
$ gzip disk.img
```

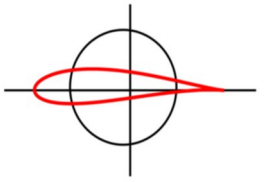
/dev/random /dev/urandom

/dev/random cihazı sistemdeki entropi değişimine göre keyfi sayı üretir.

Sistemde entropi değişikliği yoksa sayı üretilmez. Bundan dolayı üzerinde iş yapılmayan makinelerde sayıların üretilme hızı çooook yavaştır. Fakat nitelikli keyfi sayılar üretilir.

/dev/urandom cihazı hızlı keyfi sayı üretir. Üretilen sayılar, keyiflik bakımdan çok da nitelikli değildir.

```
$ dd if=/dev/random

$ cat /dev/random | hexdump -C

$ dd if=/dev/urandom of=/dev/sdc   # DENEMEYİN  !!!
```

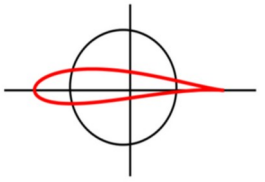


/dev/full

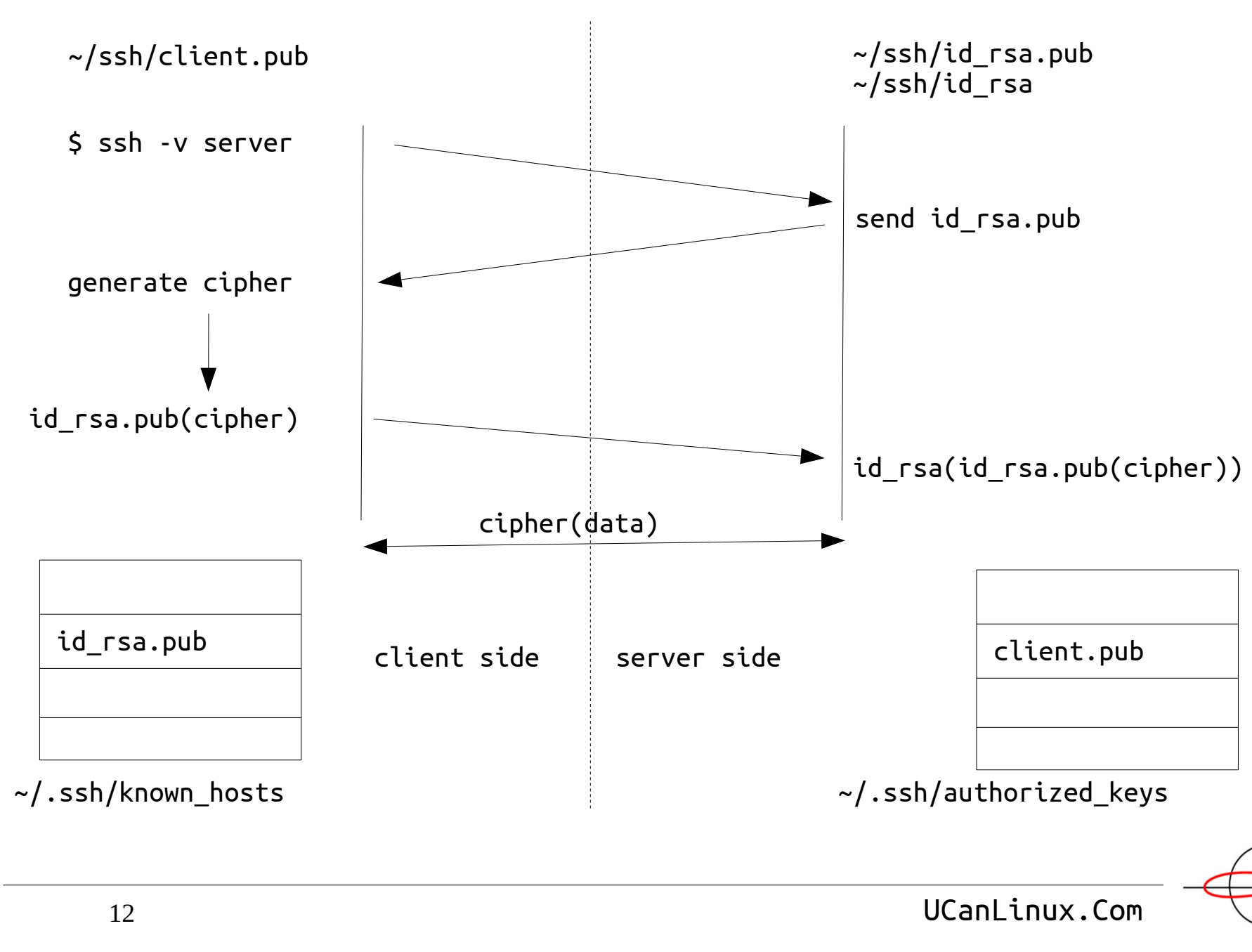
Bu cihaz tam dolu bir cihazı temsi eder.
Uygulama programlarında “tam dolu dosya” testi için kullanılabilir.

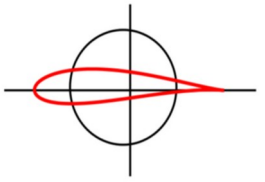
Bu cihazdan veri okunabilir ama yazılamaz.
Veri yazılacağı zaman cihaz dolu veya dosya dolu hatası alınır.

```
$ echo 123 > /dev/full
$ echo $?
```

ssh


~/ssh/client.pub
~/ssh/id_rsa.pub
~/ssh/id_rsa
$ ssh -v server
send id_rsa.pub
generate cipher
id_rsa.pub(cipher)
id_rsa(id_rsa.pub(cipher))
cipher(data)
id_rsa.pub
client side
server side
client.pub
~/.ssh/known_hosts
~/.ssh/authorized_keys

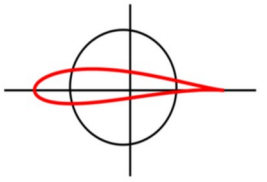


Standard bir paketin native/cross derlenmesi

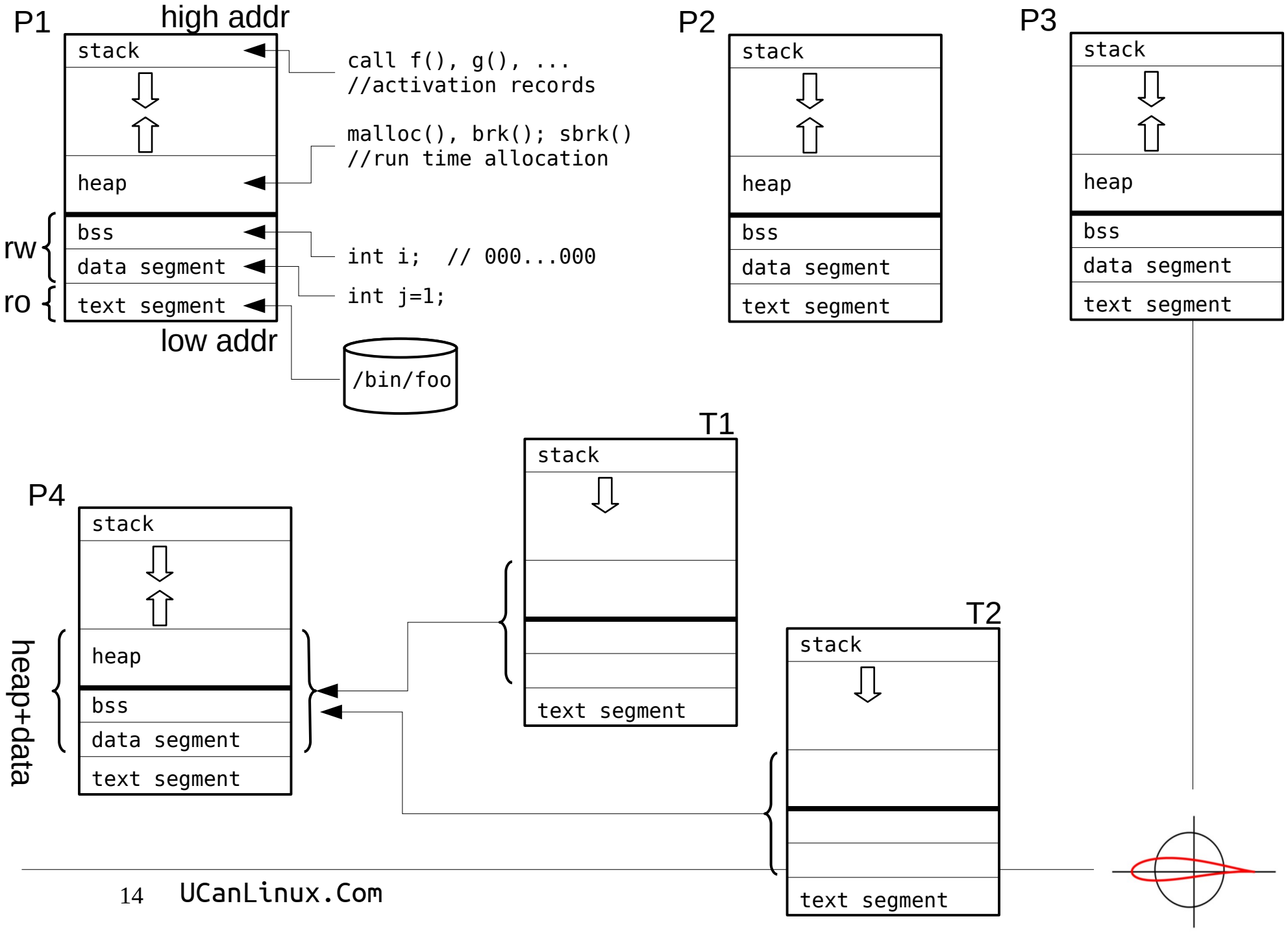
Linux paketleri genelde “configure/make/make install” ile derlenir.

Örnek, zlib paketinin derlenmesi

```
$ wget https://zlib.net/current/zlib.tar.gz
$ tar xvf zlib.tar.gz
$ mkdir /tmp/RootFS
$ cd cd zlib-1.3.1
$ ./configure --prefix=/tmp/RootFS  # native
$ CROSS_PREFIX=arm-linux-gnueabihf-  ./configure -prefix=/tmp/RootFS  # cross
$ make install

$ cd /tmp/RootFS
$ ls -l
$ tree .

$ cd lib/pkgconfig
$ ls
$ more zlib.pc
$ export PKG_CONFIG_PATH=/tmp/RootFS/lib/pkgconfig
$ pkgconf --cflags --libs zlib
```

Proccess Memory Layout

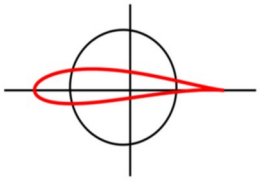


32 bit'lik makineler için 4GB'lık sanal bir bellek ayrılır.
Genel yapı verilmiştir.
Segment adreslerini keyfi kaydırma, mmap alanı ve linux için en tepede ayrılan alanlar ayrıca gösterilmemiştir.

```
int p[4096];
int main(){
    return 0;
}
```

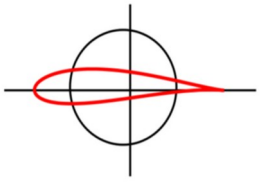
```
$ gcc prog.c -o prog    // int p[4096];
$ size prog
   text     data      bss      dec      hex  filename
   1418      544    16416    18378     47ca  prog
$ ls -l prog
-rwxrwxr-x 1 nazim nazim 16488 Nis 30 05:52 prog

$ gcc prog.c -o prog   // int p[4096]= {3};
$ size prog
   text     data      bss      dec      hex  filename
   1418    16944        8    18370     47c2  prog
$ ls -l prog
-rwxrwxr-x 1 nazim nazim 32888 Nis 30 05:52 prog
```

bss: block started by symbol or better save space :)



File Operations

```
struct file_operations {
    struct module *owner;
    loff_t (*llseek) (struct file *, loff_t, int);
    ssize_t (*read) (struct file *, char __user *, size_t, loff_t *);
    ssize_t (*write) (struct file *, const char __user *, size_t, loff_t *);
    ssize_t (*aio_read) (struct kiocb *, const struct iovec *, unsigned long, loff_t);
    ssize_t (*aio_write) (struct kiocb *, const struct iovec *, unsigned long, loff_t);
    int (*readdir) (struct file *, void *, filldir_t);
    unsigned int (*poll) (struct file *, struct poll_table_struct *);
    long (*unlocked_ioctl) (struct file *, unsigned int, unsigned long);
    long (*compat_ioctl) (struct file *, unsigned int, unsigned long);
    int (*mmap) (struct file *, struct vm_area_struct *);
    int (*open) (struct inode *, struct file *);
    int (*flush) (struct file *, fl_owner_t id);
    int (*release) (struct inode *, struct file *);
    int (*fsync) (struct file *, loff_t, loff_t, int datasync);
    int (*aio_fsync) (struct kiocb *, int datasync);
    int (*fasync) (int, struct file *, int);
    int (*lock) (struct file *, int, struct file_lock *);
    ssize_t (*sendpage) (struct file *, struct page *, int, size_t, loff_t *, int);
    unsigned long (*get_unmapped_area)(struct file *, unsigned long, unsigned long, unsigned
long, unsigned long);
    int (*check_flags)(int);
    int (*flock) (struct file *, int, struct file_lock *);
    ssize_t (*splice_write)(struct pipe_inode_info *, struct file *, loff_t *, size_t,
unsigned int);
    ssize_t (*splice_read)(struct file *, loff_t *, struct pipe_inode_info *, size_t, unsigned
int);
    int (*setlease)(struct file *, long, struct file_lock **);
    long (*fallocate)(struct file *file, int mode, loff_t offset,
              loff_t len);
};
```

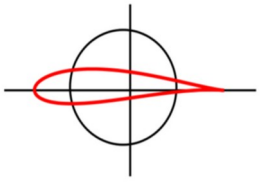
Dosya Sistemleri

Dosya ve dizinlerin, hiyerarşik bir yapıda tutulduğu veri yapılarıdır.
Gömülü sistemlerde genelde aşağıdaki dosya sistemleri kullanılır.

- **vfat**
- **ext2**
- **ext3**
- **ext4**
- **tmpfs**
- **ramdisk**
- **initramfs**
- **cpio**
- **NFS**
- **cramfs**
- **romfs**
- **squashfs**
- **zram**
- **eCryptfs**
- **ubifs**
- **pseudo/virtual fs**

Dosya sistemleri ve parametreleri amaca uygun seçilmelidir.
Kötü seçim kaynak israfına ve fiziksel yapının bozulmasına sebeb olabilir.

Linux pek çok dosya sistemine destek verir.
Fakat çok az dosya sistemi Gömülü Sistemler için uygundur.
Dosya sistemi öncelikle cihaza uygun seçilmelidir.
Prensip olarak swap kullanılmamalıdır.

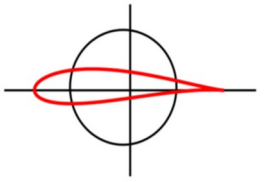


Dosya sistemlerinin kurulduğu bazı ortamlar,

SD/MMC/eMMC	vfat, ext2, ext3, ext4, ...
NAND/NOR Flash	ubifs, jffs2, yaffs2, ...
RAM:	ramdisk, tmpfs, ramfs, zramfs, ...
Network	NFS, samba, ...
Pseudo/virtual	devfs, procfs, sysfs, pipefs, debugfs, ...

Seçilen her dosya sistemi için çekirdeğe uygun destek verilmelidir.
Dosya sistemleri genelde çekirdekte çok yer kaplarlar.
Kullanılmayan dosya sistemleri çekirdeğe eklenmemelidir.
Kök dosya sistemi hariç, bütün dosya sistemleri modül olarak derlenebilir.

cat /proc/filesystems komutu ile çekirdeğin o an için desteklediği dosya sistemleri incelenebilir.



VFAT

MS-Windows dosya sistemidir.

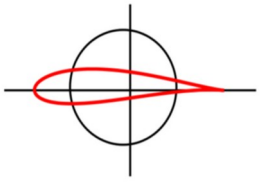
255 karaktere kadar dosya isimlerine izin verir.

Linux ihtiyaçlarını karşılayamaz.
Sahiplik, mod, sembolik link vs. mevcut değildir.
Asla kök dosya sistemi vfat olmamalıdır.

İç yapısı aşırı basit olduğu için daha çok u-boot gibi açılış-yükleyicileri tarafından kullanılır.

Genelde SD/MMC kartların 1. bölümüne kurulur.

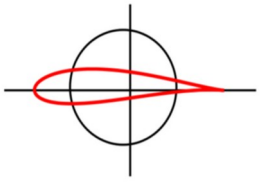
```
$ cd /tmp

$ dd if=/dev/zero of=disk1.img bs=1M count=32
32+0 records in
32+0 records out
33554432 bytes (34 MB) copied, 0,0855615 s, 392 MB/s

$ mkfs.vfat disk1.img
mkfs.vfat 3.0.14 (23 Jan 2023)

$ mkdir /mnt/disk1

$ mount disk1.img /mnt/disk1
```

```
$ df /mnt/disk1
Filesystem     1K-blocks  Used Available Use% Mounted on
/dev/loop0         32686     0     32686   0% /mnt/disk1

$ ls -l /mnt/disk1
total 0

$ mkfs.vfat /media/user/BOOT
mkfs.vfat 3.0.14 (23 Jan 2023)

$ mkdir /mnt/disk1

$ mount disk1.img /mnt/disk1

$ df /mnt/disk1
Filesystem     1K-blocks  Used Available Use% Mounted on
/dev/loop0         32686     0     32686   0% /mnt/disk1

$ ls -l /mnt/disk1
total 0
```

Çekirdek desteği aşağıdaki gibi verilir.

```
   File systems -->
         DOS/FAT/NT Filesystems -->
               <*> VFAT (Windows-95) fs support
```



ext2

Linux'un en eski dosya sistemlerinden biridir.
Çok uzun zamandan beri ext2 ile ilgili hiç bir “bug report” yapılmamıştır.

Kök dosya sistemi için gerekli bütün ihtiyaçları karşılar.
Masaüstü sistemlerde artık kullanılmamaktadır.
Ani kapanmalara karşı çok dayanıksızdır.

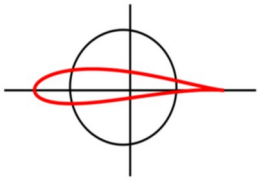
MMC cihazlarında, RO modu ile bağlanarak kullanılmalıdır.
Kök dosya sistemi olarak, RW modunda bağlanmamalıdır.

Kök dosya sistemi RW modunda kullanılacaksa, önce ro bağlanmalı,
sonra fsck yapılmalı, gerekirse repair edilmeli ve sonra rw bağlanmalıdır.

```
$ mkdir /mnt/disk2

$ dd if=/dev/zero of=disk2.img bs=1M count=32
32+0 records in
32+0 records out
33554432 bytes (34 MB) copied, 0,069849 s, 480 MB/s

$ mkfs.ext2  disk2.img
$ mount disk2.img /mnt/disk2
$ df /mnt/disk2
```

Çekirdek desteği aşağıdaki gibi verilir.

```
File systems -->
   <*> Second extended fs support
```

ext3

ext2'nin devamı ve log tabanlıdır.
Ani kapanmalara karşı çok dayanıklıdır.
Dosya bütünlüğü çok zor bozulur.
Recovery durumu çok hızlıdır.

Yerini ext4 sistemine bırakmıştır.
MMC cihazlarında, RW modu ile bağlanarak kullanılabilir.
Log tabanlı bir sistem olduğu için, diske yazılacak bütün
veriler önce bir log tablosunda tutulur, daha sonra diske yazılır.

Crash durumunda sonra e2fsck çalıştırmaya gerek yoktur.
Log sayesinde dosya bütünlüğü her zaman sağlanır.

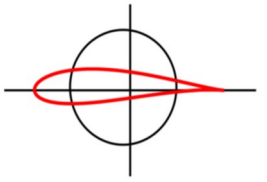
ext3 = journal + ext2

ext2 ve ext3 arasında her zaman geçiş yapılabilir.
Performans ve dosya bütünlüğü arasında ters ilişki vardır.

data=write_back crash sonrası yazılmayan veri kalabilir.
data=ordered Daha güvenlidir ama yavaştır. Tavsiye edilir.

```
$ dd if=/dev/zero of=disk3.img bs=1M count=32
$ mkfs.ext3  disk3.img
$ mkdir /mnt/disk3 && mount disk3.img /mnt/disk3 &&  df /mnt/disk3
```

```
File systems -->
   <*> Ext3 journalling file system support
      [*]   Default to 'data=ordered' in ext3
```

ext4

ext3'ün devamıdır ama ext3 ile uyumlu değildir.

ext4 dosya sistemi, ext3'ü mount edebilir ve ext3'e göre daha iyi performans gösterir.

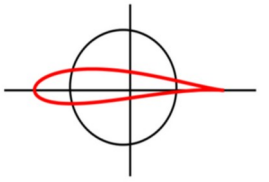
Doğrudan ext4 kullanılması tavsiye edilir.

ext3, ext2 ile uyumlu idi.
Her iki yönde de geçiş yapılabiliyordu.

Örnek dosya sistemi kuruluşu ve çekirdek desteği ext2/ext3 gibidir.

Gömülü sistemlerde, MMC tarafında kök dosya sistemi olarak rw modunda mount edilebilir.

ext2 sisteminin rw modunda bağlanması tavsiye edilmez.



tmpfs

Sanal bellektei (virtual memory) kurulan bir dosya sistemidir.

virtual memory = ram + swap

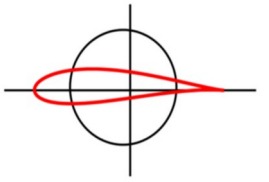
Doldukça büyür, içi boşaldıkça küçülürler.
Ayrılan kapasiteyi RAM'dan çalmazlar.
Swap edilebilirler.

Disk cache sisteminin değiştirilmiş bir hali gibidir.
Çok az yer kaplar.
Bundan dolayı seçilmese bile çekirdek tarafında desteği vardır.
/dev/shm, /dev gibi pek çok cihaz, bu dosya sistemini kullanır.

Kapasite verilmezse, RAM'in yarısı kabul edilir.
Çalışma anında kapasite artırılabilir.

umount edildiği an veriler kaybolur.
Güç kesildiği an veriler kaybolur.

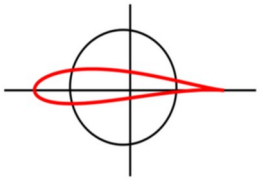
```
$ mkdir /mnt/disk4

$ mount -t tmpfs tmpfs /mnt/disk4
```

```
$ df /mnt/disk4
Filesystem     1K-blocks  Used Available Use% Mounted on
tmpfs            2011344     0   2011344   0% /mnt/disk4

$ free
             total       used       free     shared    buffers cached
Mem:       4022692    2092160    1930532          0     599184 892832
-/+ buffers/cache:     600144    3422548
Swap:      2097148          0    2097148
```

mount edilen bütün dosyalar, işleri bitince umount edilmelidir.
32MB için, ext3 dosya sisteminin maliyeti %15'tir.

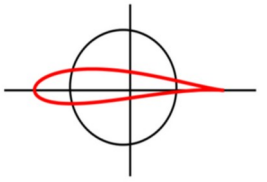
```
$ df /mnt/disk?

Filesystem     1K-blocks  Used Available Use% Mounted on

/dev/loop0         32686     0     32686   0% /mnt/disk1
/dev/loop1         31729   395     29696   2% /mnt/disk2
/dev/loop2         31729  4508     25583  15% /mnt/disk3
tmpfs            2011344     0   2011344   0% /mnt/disk4
```



```
$ mount | grep disk
/tmp/disk1.img on /mnt/disk1 type vfat (rw)
/tmp/disk2.img on /mnt/disk2 type ext2 (rw)
/tmp/disk3.img on /mnt/disk3 type ext3 (rw)
tmpfs on /mnt/disk4 type tmpfs (rw)

$ umount /mnt/disk1
$ umount /mnt/disk2
$ umount /mnt/disk3
$ umount /mnt/disk4
```

Çekirdek desteği aşağıdaki gibi verilir.

```
File systems -->
    Pseudo filesystems -->
        [*] Virtual memory file system support (former shm fs)
```

nfs

Network File System, ağ üzerinden dosyalara erişim tekniğidir.
Diğer bir deyişle, ağ üzerinden dizin paylaşım yöntemidir.

Kök dosya sistemi ağ üzerinden bağlanabilir ya da
host üzerindeki herhangi bir dizin export edilebilir.

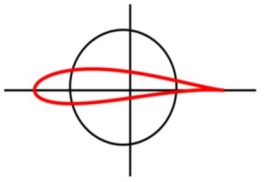
/etc/exports
/tftpboot 10.0.0.0/24(rw,insecure, no_subtree_check,async,no_root_squash)

$ exportfs -avr ile yapılan güncellemeler sunucuya bildirilir.

no_root_squash sayesinde root yetkisi ile işlem yapılabilir.
insecure ile güvenli portların dışında da port numarası kullanılabilir.
no_subtree_check ile dışarı taşan dizinlere erişim sağlanır.

Ağ için çekirdek parametreleri en genel hali ile aşağıda verilmiştir.

ip=<client-ip>:<server-ip>:<gw-ip>:<netmask>:<hostname>:
<device>:<autoconf>:<dns0-ip>:<dns1-ip>



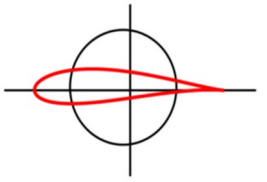
```
autoconf parameters: Documentation/filesystems/nfsroot.txt

off or none: don't use autoconfiguration (do static IP assignment instead)
on or any:   use any protocol available in the kernel (default)
dhcp:        use DHCP
bootp:       use BOOTP
rarp:        use RARP
both:        use both BOOTP and RARP but not DHCP
                    (old option kept for backwards compatibility)
Default: any
```

Örnek çekirdek parametreleri aşağıda verilmiştir.

```
ip=10.0.0.111:10.0.0.4:10.0.0.4:255.255.255.0:test1:eth0:off
   root=/dev/nfs
   rw
   nfsroot=10.0.0.4:/tftpboot/RootFS
```

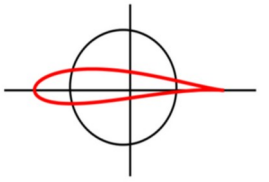
Sıradan bir dizini mount etmek için,

```
$ mount -t nfs 10.0.0.4:/tftpboot/app /mnt/nfs
 ...
$ umount /mnt/nfs
```

Kök dosya sistemi NFS üzerinden kullanılacaksa çekirdek aşağıdaki gibi derlenir.

```
[*] Networking support
    Networking options -->
      [*] TCP/IP networking
         [*]   IP: kernel level autoconfiguration

File systems -->
     [*] Network File Systems
     <*>   NFS client support
     [*]      Root file system on NFS
```

Pseudo File Systems

Sözde dosya sistemleri, kernel ve user space arasında iki yönlü bilgi alışverişi için tasarlanmıştır.
Dosya ve dizinler her zaman ya boot anında ya da tam kullanım anında kurulurlar ve sistem kapanınca yok olurlar. Fiziksel olarak bir diskte kurulmazlar bundan dolayı dosya boyları her zaman 0'dır ama içleri dolu olabilir.
Gömülü sistemlerde genelde açılış betikleri tarafından mount edilirler.
Çok kullanılan bazı dosya sistemlerinin mount edilmesi...

```
$ mount -t proc      proc   /proc
$ mount -t sysfs     sysfs  /sys
$ mount -t devtmpfs  none   /dev
$ mount -t devpts    devpts /dev/pts
$ mount -t tmpfs     tmpfs  /dev/shm
                     ^
                     bu kolonun tamamı none olabilir.
```