

Sistem Programlama

Ders 6

Dr. Öğr. Üyesi Mehmet Dinçer Erbaş
Bolu Abant İzzet Baysal Üniversitesi
Mühendislik Fakültesi
Bilgisayar Mühendisliği Bölümü

chown, fchown ve lchown fonksiyonları

- chown fonksiyonu bir dosyanın kullanıcı numarasını ve grup numarasını değiştirmek için kullanılabilir.

```
#include <unistd.h>
```

```
int chown(const char *pathname, uid_t owner, gid_t group);
```

```
int fchown(int filedes, uid_t owner, gid_t group);
```

```
int lchown(const char *pathname, uid_t owner, gid_t group);
```

Dönüş: ok ise 0, hata ise -1.

- Eğer sahip veya grup argümanı -1 ise önceki değer değiştirilmez.
- BSD-tabanlı sistemlerde sadece yönetici bir dosyanın sahibini değiştirebilir.

chown, fchown ve lchown fonksiyonları

- POSIX.1 standartlarında `_POSIX_CHOWN_RESTRICTED` sabiti bulunmaktadır. Bu sabitin değerine göre:
 - Sadece yönetici bir dosyanın kullanıcı numarasını değiştirebilir.
 - Yönetici olmayan bir kullanıcı grup numarasını değiştirebilmesi için işlemin dosyanın sahibi olması, sahibin -1 olarak verilmiş olması veya kullanıcı numarasına eşit olması ve verilen grup numarası işlemin efektif kullanıcı numarası veya destekleyici grup numarasına eşit olması gerekmektedir.

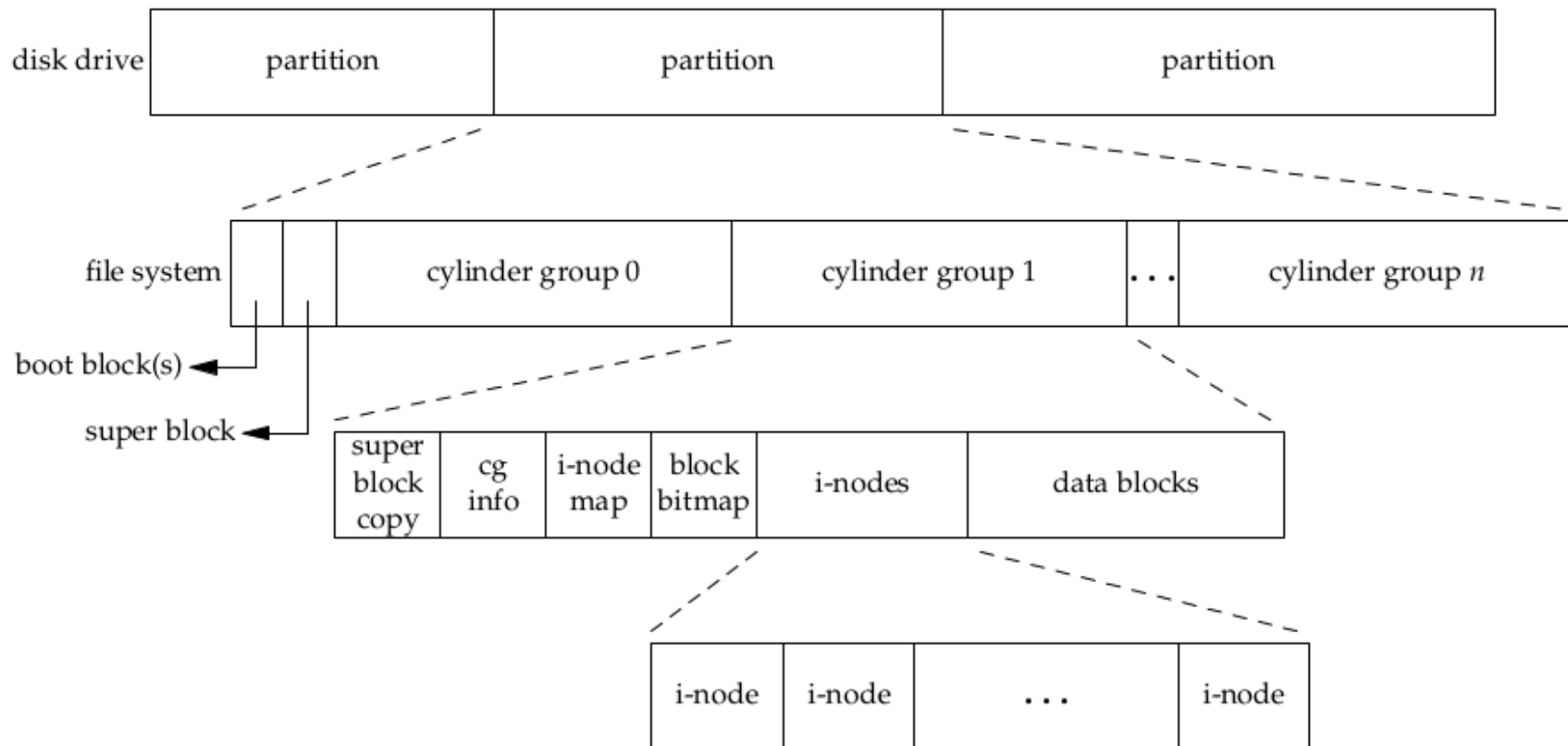
Dosya büyüklükleri

- Dosyanın stat yapısındaki `st_size` alanı dosyanın byte olarak büyüklüğünü verir.
 - Bu alan sadece normal dosyalar, klasörler ve sembolik linkler için anlamlıdır.
- Normal dosyalar için 0 büyüklük olabilir.
- Klasörler için genellikle 16 veya 512 gibi sayıların katı olan büyüklükleri vardır.
- Sembolik linklerin büyüklüğü dosya ismindeki byte sayısıdır.

Dosya sistemleri

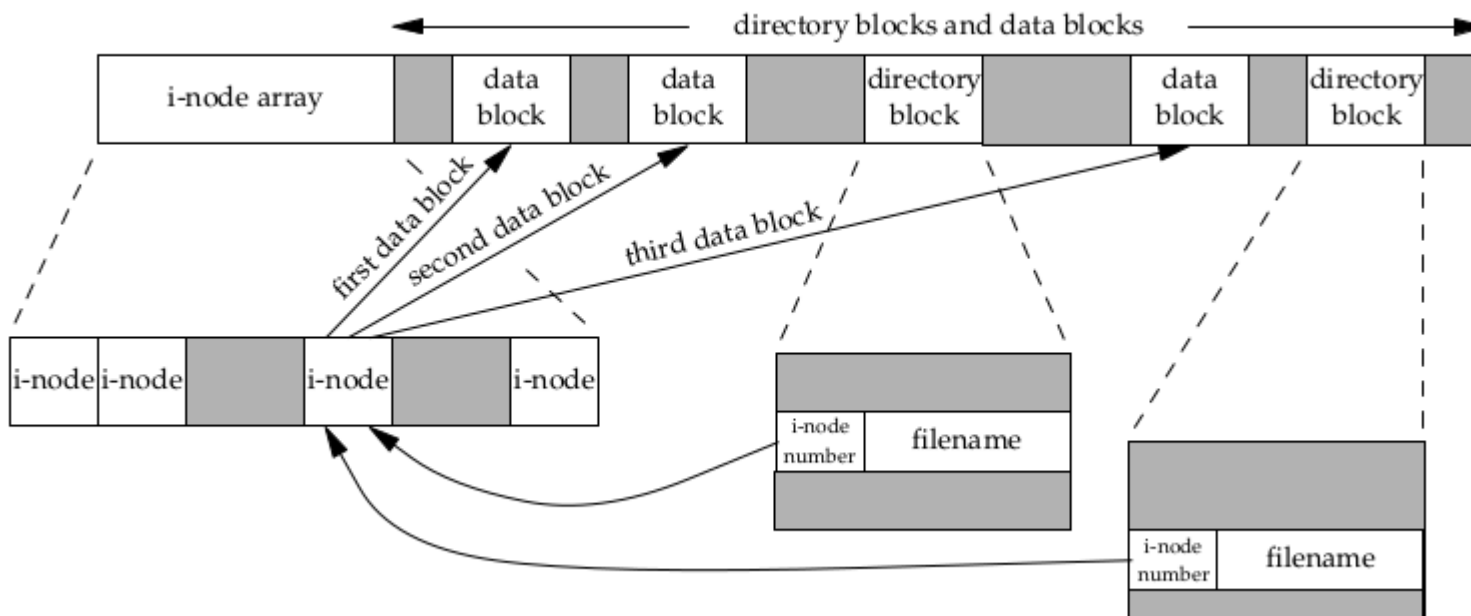
- Disk birden fazla mantıksal bölmeye ayrılabilir.
- Her mantıksal bölme kendi dosya sistemine sahip olabilir.
- i-node'lar bir dosya hakkında birçok bilginin saklandığı sabit uzunlukta kayıtlardır.
 - Herhangi bir dosya açıldığında bu dosyanın i-node bilgisine bir işaretçi atanır.

Dosya sistemleri



Dosya sistemleri

- Bir silindir grubunun i-node ve veri kısmına yakından baktığımızda aşağıdaki gibi bir görüntü oluşur.



Dosya sistemleri

- Şekilde aşağıda belirtilen durumlar gözlemlenebilmektedir:
 - Birden fazla klasör kaydı aynı i-node kaydına işaret etmektedir.
 - Her i-node içerisinde kendisine işaret eden klasör kayıtlarının sayısı saklanır.
 - Bir dosyanın sistemden silinebilmesi için kendine işaret eden klasör kaydı sayısının 0 olması gerekmektedir.
 - Bu sebeple bir klasör kaydını silen işlem unlink olarak adlandırılır.
 - Her dosyanın stat yapısındaki `set_nlink` alanı dosyaya işaret eden kayıt sayısını saklar.
 - Bu tip işaret linklerine hard link (sıkı bağ) adı verilir.
 - Diğer link tipine sembolink link (sembolik bağ) denir.
 - Sembolik linklerde işaret edilen dosyanın ismi saklanır.

Dosya sistemleri

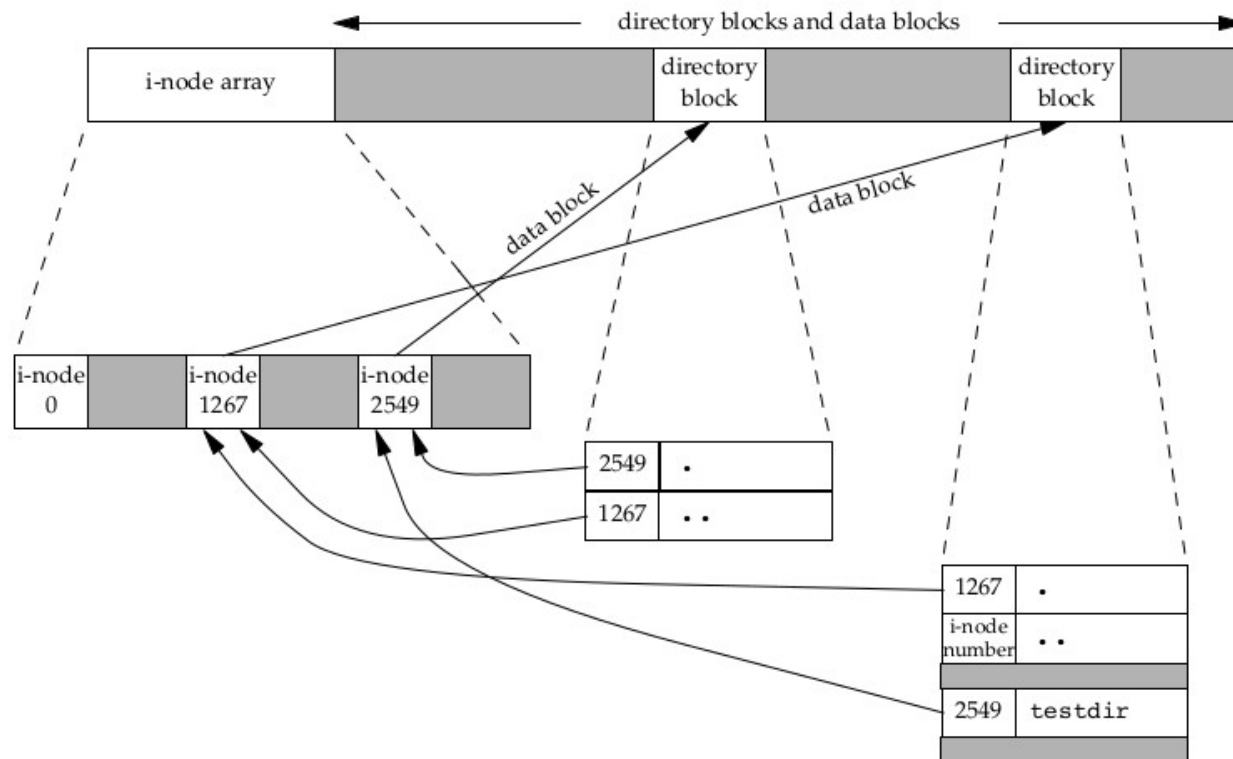
- Şekilde aşağıda belirtilen durumlar gözlemlenebilmektedir:
 - i-node bir dosyanın her türlü bilgisini saklar.
 - Örneğin dosya tipi, dosya üzerindeki izinler, dosyanın büyüklüğü, dosyanın disk üzerinde fiziksel olarak saklandığı bloklara işaretçiler vb.
 - stat yapısındaki bilgilerin büyük çoğunluğu i-node'dan gelir.
 - Klasör kaydında ise sadece dosyanın ismi ve i-node numarası saklanır.
 - Bir klasörde bulunan i-node numarası aynı dosya sistemindeki bir i-node'a işaret ettiği için, bir klasörde başka bir dosya sistemindeki i-node için kayıt tutulmaz.
 - Bu sebeple dosyalar arasında sıkı bağ oluşturan ln komutu başka bir dosya sistemindeki dosyaya link oluşturamaz.

Dosya sistemleri

- Şekilde aşağıda belirtilen durumlar gözlemlenebilmektedir:
 - Bir dosyanın dosya sistemini değiştirmeden ismini değiştirmek istediğimizde, dosyanın içeriğinin yerinin değiştirilmesine gerek yoktur.
 - Tek yapılması gereken klasörde, aynı i-node'a işaret eden yeni isimle bir kayıt oluşturmak ve eski kaydı unlink yapmaktır (yani eski kaydı silmek).
 - Bu durumda i-node'daki link sayısı aynı kalacaktır.
 - Bahsettiğimiz işlem mv operasyonudur.

Dosya sistemleri

- Klasörlerin link sayıları benzer şekilde hesaplanmaktadır.
- Örneğin bulunduğumuz klasörde **testdir** isminde bir klasör oluşturalım.



link, unlink, remove ve rename fonksiyonları

- link (2)

```
#include <unistd.h>
```

```
int link(const char *existingpath, const char *newpath);
```

Dönüş: OK ise 0, hata ise -1.

- Bulunan bir dosyaya sıkı bağ oluşturur (hard link).
- Bu fonksiyon ile bulunan bir dosyaya yeni bir klasör kaydı oluşur.
 - Klasör kaydı var ise hata döner.
 - Yeni bağlantının son kısmı oluşturulur. Yoladının geri kalanı bulunmalıdır.
- Yeni klasör kaydının oluşması ve bağ sayacının artırılması atomik olarak yapılmalıdır.
- POSIX.1 dosya sistemleri arasında bağ oluşturma imkanı vardır ancak çoğu Unix versiyonu buna izin vermez.
- Sadece yönetici klasörlere bağ verebilir.

link, unlink, remove ve rename fonksiyonları

- unlink (2)

```
#include <unistd.h>
```

```
int unlink(const char *pathname);
```

Dönüş: OK ise 0, hata ise -1.

- Klasör kaydını siler ve bağ sayacını bir azaltır.
- Eğer dosyaya başka bağlar var ise, dosya diğer linklerden halen erişilebilir.
- Link sayısı 0 olduğunda dosyanın içeriği silinir.
- Ayrıca dosya başka bir dosya tarafından açık tutuluyorsa, dosyanın içeriği silinmez.
 - Bir dosya kapandığında kernel dosyayı açık tutan işlem sayısını kontrol eder. Dosyayı açık tutan işlem sayısı 0 ise, kernel bağ sayısını kontrol eder. Bağ sayısı da 0 ise dosya içeriği silinir.

link, unlink, remove ve rename fonksiyonları

- Örnek18

```
$ ls -l tempfile          look at how big the file is
-rw-r----- 1 sar      413265408 Jan 21 07:14 tempfile
$ df /home                check how much free space is available
Filesystem      1K-blocks      Used Available Use% Mounted on
/dev/hda4        11021440    1956332    9065108   18% /home
$ ./a.out &              run the program in Figure 4.16 in the background
1364                the shell prints its process ID
$ file unlinked           the file is unlinked
ls -l tempfile           see if the filename is still there
ls: tempfile: No such file or directory  the directory entry is gone
$ df /home                see if the space is available yet
Filesystem      1K-blocks      Used Available Use% Mounted on
/dev/hda4        11021440    1956332    9065108   18% /home
$ done                    the program is done, all open files are closed
df /home              now the disk space should be available
Filesystem      1K-blocks      Used Available Use% Mounted on
/dev/hda4        11021440    1552352    9469088   15% /home
now the 394.1 MB of disk space are available
```

link, unlink, remove ve rename fonksiyonları

- remove

```
#include <stdio.h>
```

```
int remove(const char *pathname);
```

Dönüş: OK ise 0, hata ise -1.

- Bir dosya veya klasörü silmek için ayrıca remove fonksiyonunu kullanabiliriz.
- Bir dosya için kullandığımızda remove unlink ile aynıdır. Bir klasör için kullandığımızda remove rmdir ile aynıdır.

link, unlink, remove ve rename fonksiyonları

- rename

```
#include <stdio.h>
```

```
int rename(const char *oldname, const char *newname);
```

Dönüş: OK ise 0, hata ise -1.

- Eğer `oldname` bir dosya ismi ise
 - Eğer `newname` bulunmakta ve bir klasör değil ise, `newname` silinir ve `oldname` yerine `newname` ismi geçer.
 - `newname` bulunmakta ve bir klasör ise hata döner.
 - Bu işlemin yapılabilmesi için `oldname` ve `newname` dosyalarının bulunduğu klasörlerde `w+x` hakları bulunmalıdır.

link, unlink, remove ve rename fonksiyonları

- rename
- Eğer `oldname` bir klasör ise
 - Eğer `newname` bulunmakta ve boş bir klasör ise, `newname` silinir, `oldname` yerine `newname` geçer.
 - Eğer `newname` var ve bir dosya ise hata döner.
 - Eğer `oldname`, `newname`'in ön parçası ise hata döner.
 - `oldname` ve `newname` in bulunduğu klasörlerde w+x hakları olmalıdır.

Sembolik bağlar

- Sembolik bağlar bir dosyaya dolaylı işaretçidir.
 - Sıkı bağlar dosyanın i-node bilgisine işaret ederler. Sembolik bağlar ise klasör kaydına işaret ederler.
- Sembolik bağlar, sıkı bağlar ile ilgili kısıtlamalar nedeniyle oluşturulmuştur.
 - Sıkı bağ oluşturabilmek için dosyaların aynı dosya sisteminde olması gerekir.
 - Bir klasöre sıkı bağ ancak yönetici tarafından oluşturulabilir.
- Sembolik bağlarda ise bu tür kısıtlamalar yoktur.
- Sembolik bağlar kullanılarak bir dosya veya bütün dosya hiyerarşisi sistemdeki başka bir yere taşınabilir.

Sembolik bağlar

- Sembolik bağlar ile bir fonksiyon kullandığınızda, bu fonksiyonun sembolik bağı takip edip etmediğini bilmelisiniz.
 - Takip ediyorsa sembolik bağın işaret ettiği dosyayı etkiler.
 - Etmiyorsa, etkilemez.

Sembolik bağlar

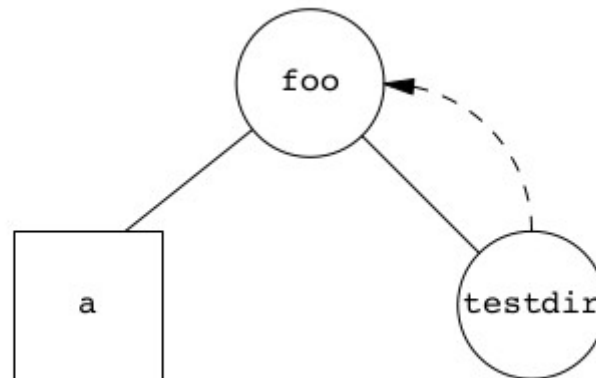
Function	Does not follow symbolic link	Follows symbolic link
access		•
chdir		•
chmod		•
chown		•
creat		•
exec		•
lchown	•	
link		•
lstat	•	
open		•
opendir		•
pathconf		•
readlink	•	
remove	•	
rename	•	
stat		•
truncate		•
unlink	•	

Sembolik bağlar

- Sembolik bağlar kullanarak dosya sisteminde döngüler oluşturabiliriz.
 - Bu durumda birçok fonksiyon bu durumu farkedemez ve ELOOP hatası döner.

```
$ mkdir foo
$ touch foo/a
$ ln -s ../foo foo/testdir
$ ls -l foo
```

total 0					
-rw-r-----	1	sar	0	Jan	22 00:16 a
lrwxrwxrwx	1	sar	6	Jan	22 00:16 testdir -> ../foo



Sembolik bağlar

- Sembolik bağlar kullanarak dosya sisteminde döngüler oluşturabiliriz.
 - Yukarıdaki şekilde klasörler oluşturursak ve `ftw(3)` fonksiyonu ile dosyaları listelersek.

```
foo
foo/a
foo/testdir
foo/testdir/a
foo/testdir/testdir
foo/testdir/testdir/a
foo/testdir/testdir/testdir
foo/testdir/testdir/testdir/a
```

- Bu sorunu ortadan kaldırmak kolaydır.
 - `foo/testdir` için `unlink` yaparız. `unlink` sembolik bağları takip etmediği için sorun çözülür.
- Ancak sıkı bağ ile döngü oluşturulursa bunun çözümü oldukça zordur.

- Bu sebeple bir klasöre sıkı ba oluşturma yetkisi sadece sistem yöneticisine verilmiştir.

Sembolik bağlar

- Sembolik bağlar bazen kafa karıştırabilir.

```
$ ln -s /no/such/file myfile
```

```
$ ls myfile
```

```
$ cat myfile
```

```
cat: myfile: No such file or directory
```

```
$ ls -l myfile
```

```
lrwxrwxrwx 1 sar 25 Jan 22 00:26 myfile → /no/such/file
```

- Örnekte görüldüğü üzere olmayan bir dosyaya sembolik link oluşturmak mümkündür.

symlink ve readlink fonksiyonları

- symlink fonksiyonu ile bir sembolik bağ oluşturabiliriz.

```
#include <unistd.h>
```

```
int symlink(const char *actualpath, const char *sympath);
```

Dönüş: OK ise 0, hata ise -1.

- sympath isminde yeni bir klasör kaydı oluşur ve bu bağ actualpath dosya veya klasörüne işaret eder.
- actualpath bulunmasa bile hata vermez.
- actualpath ve sympath aynı dosya sisteminde olmak zorunda değildir.

symlink ve readlink fonksiyonları

- open fonksiyonu sembolik bağı takip ettiği için bağı kendisini açmak ve içeriğini okumak için bir fonksiyona ihtiyacımız vardır.
- Bu işlem readlink fonksiyonu ile yapılabilir.

```
#include <unistd.h>
```

```
ssize_t readlink(const char* restrict pathname, char *restrict buf,  
size_t bufsiz);
```

Dönüş: OK ise okunan byte sayısı,
hata ise -1.

- Bu fonksiyon open, read ve close işlemlerini birleştirir.
- Eğer başarılı olursa buf önbelleğine yerleştirilen byte sayısını döner.
- Önbellekteki sembolik link içeriği sonunda null değer bulunmaz.