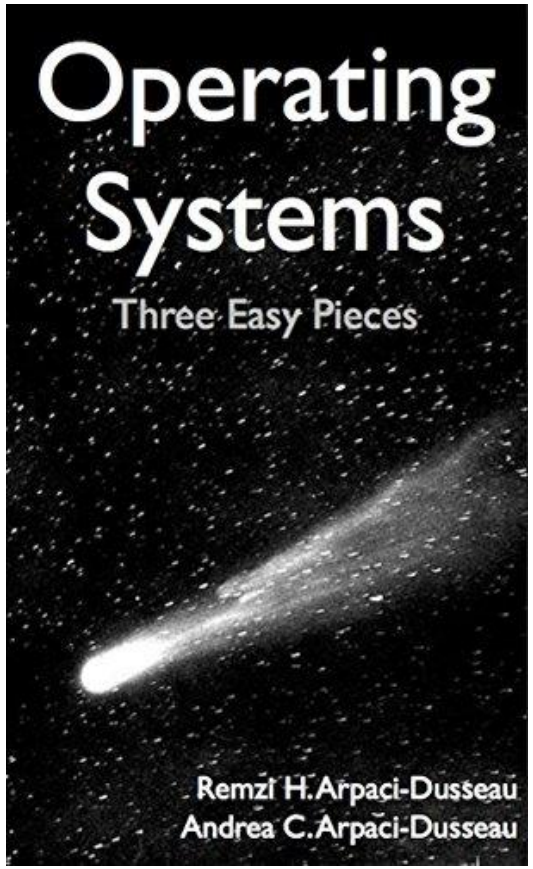


İşletim Sistemleri

12. Ders

Prof. Dr. Kemal Bıçakcı

Alıştırma Sorusu

/usr/ahmet dizininde bulunan ve göreceli yol adı ../mehmet/x olan dosyanın mutlak yol adı aşağıdakilerden hangisidir?

a. /usr/ahmet/mehmet/x
b. ../mehmet/x
c. /usr/mehmet/x
d. /mehmet/x

40. Dosya Sistemi Gerçekleştirimi

Operating System: Three Easy Pieces

``güvenli silme'' ne demektir?

``güvenli silme'' ne demektir?

Zihinsel (Mental) Model

güvenli silme için doğru metafor (parçalayıcı - shredder) kullanımı

Dosya Sistemi

Bir dosya sistemi implementasyonunun iki farklı yönü vardır:

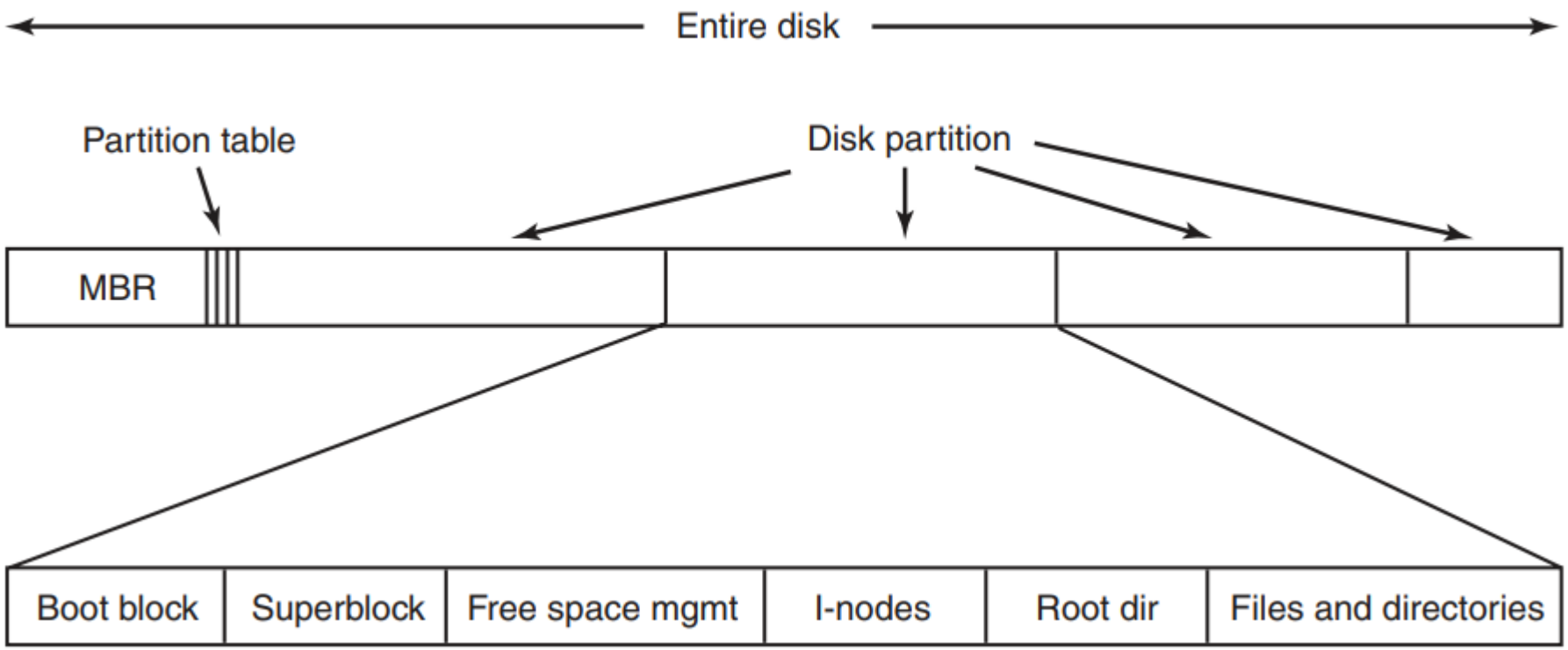
1. Veri yapıları:

- Verileri ve meta verileri düzenlemek için dosya sistemi tarafından disk üstünde ne tür yapılar kullanılır?

2. Erişim yöntemleri:

- `open()`, `read()`, `write()` gibi bir işlem tarafından yapılan sistem çağrıları veri yapıları ile nasıl eşleştirilir? Belirli bir sistem çağrısının yürütülmesi sırasında hangi yapılar okunur ve/veya yazılır?

Örnek bir Dosya Sistem Düzeni



Dosya Sistem Düzeni (Layout)

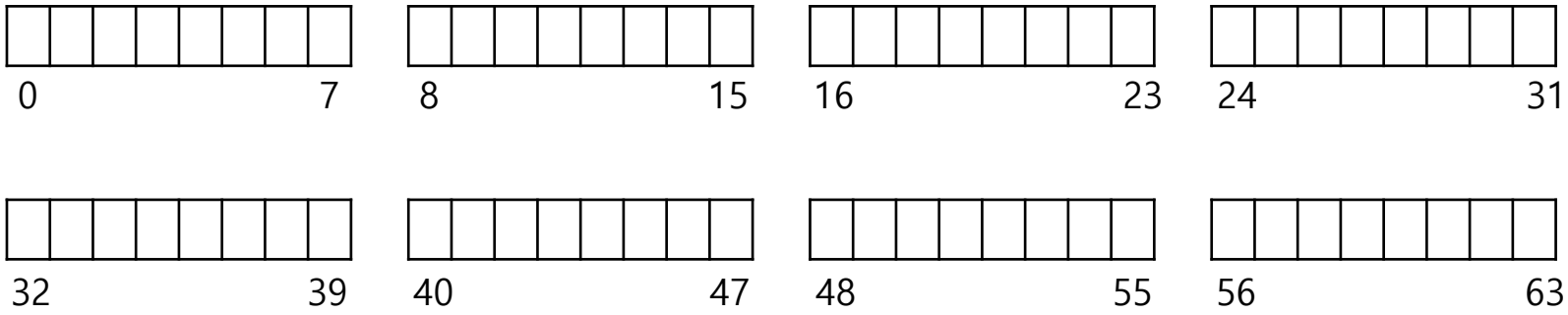
- Dosya sistemleri disklerde saklanır.
- Çoğu disk, bir veya daha fazla bölüme (partition) ayrılabilir. Her bölümde birbirinden farklı ve bağımsız dosya sistemleri bulunabilir.
- Diskin sektör 0'ına MBR (Ana Önyükleme Kaydı – Master Boot Record) adı verilir ve bilgisayarı başlatmak için kullanılır.
- MBR'nin sonu, bölüm tablosunu içerir. Bu tablo, her bölümün başlangıç ve bitiş adreslerini verir. Tablodaki bölümlerden biri etkin (aktif) olarak işaretlenmiştir.
- Bilgisayar önyüklendiğinde, BIOS MBR'yi okur ve çalıştırır. MBR programının yaptığı ilk şey, etkin bölümü bulmak, önyükleme bloğu adı verilen ilk bloğunu okumak ve çalıştırmaktır.
- Önyükleme bloğundaki program, o bölümde bulunan işletim sistemini yükler.

Dosya Sistemi Düzeni (Devam)

- Tutarlı olma adına, önyüklenebilir bir işletim sistemi içermese bile (gelecekte içerebileceği için) her bölüm bir önyükleme bloğuyla başlar.
- Bir önyükleme bloğuyla başlamak dışında, bir disk bölümünün düzeni dosya sistemine bağlı olarak farklılık gösterebilir.
- Genellikle dosya sisteminin ilk bloğu süper blok (superblock) olarak isimlendirilir. Bu blok, dosya sistemiyle ilgili tüm önemli parametreleri içerir ve bilgisayar başlatıldığında veya dosya sistemi ile ilk kez temasa geçildiğinde belleğe okunur.
- Süper bloktaki tipik bilgiler, dosya sistemi türünü, dosya sistemindeki blok sayısını ve diğer önemli sistem bilgilerini tanımlayan bir sihirli sayıyı (magic number) içerir.

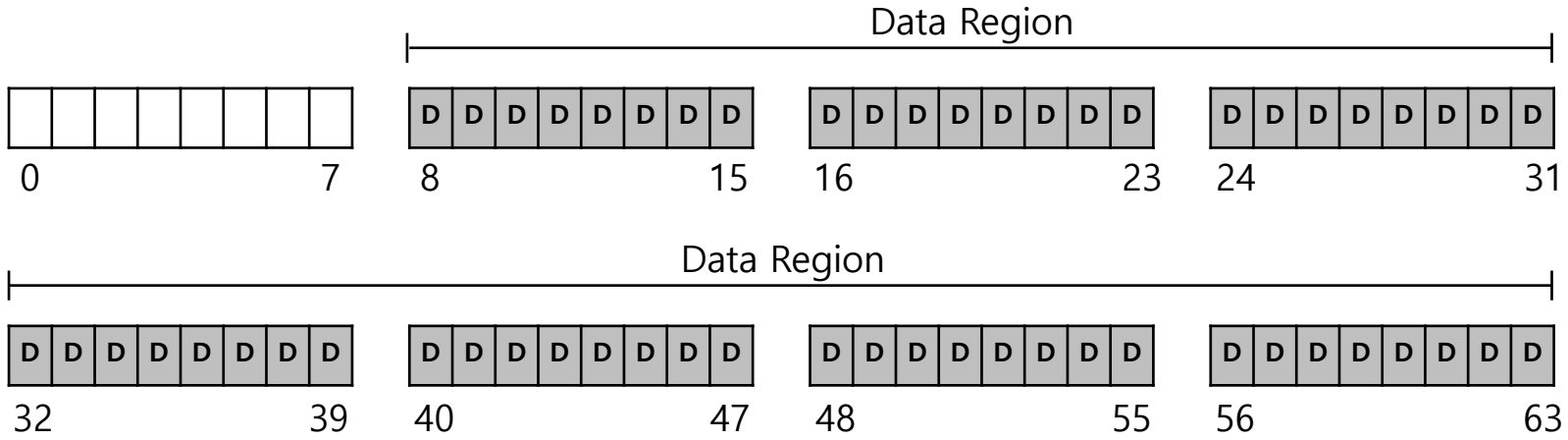
Dosya Sistemi Düzeni (Devam)

- Dosya sistemi veri yapısının genel organizasyonu ile ilgili bilgimizi daha da artıralım.
- İlk olarak, disk bloklara bölünür. Örnek bir blok boyutu 4 KB'dir.
- Bloklar *0 ile N−1* arasında adreslenir.



Veri Bölgesi

- **Veri bölgesi (Data region):** Kullanıcı verilerini depolamak için ayrılmış kısım

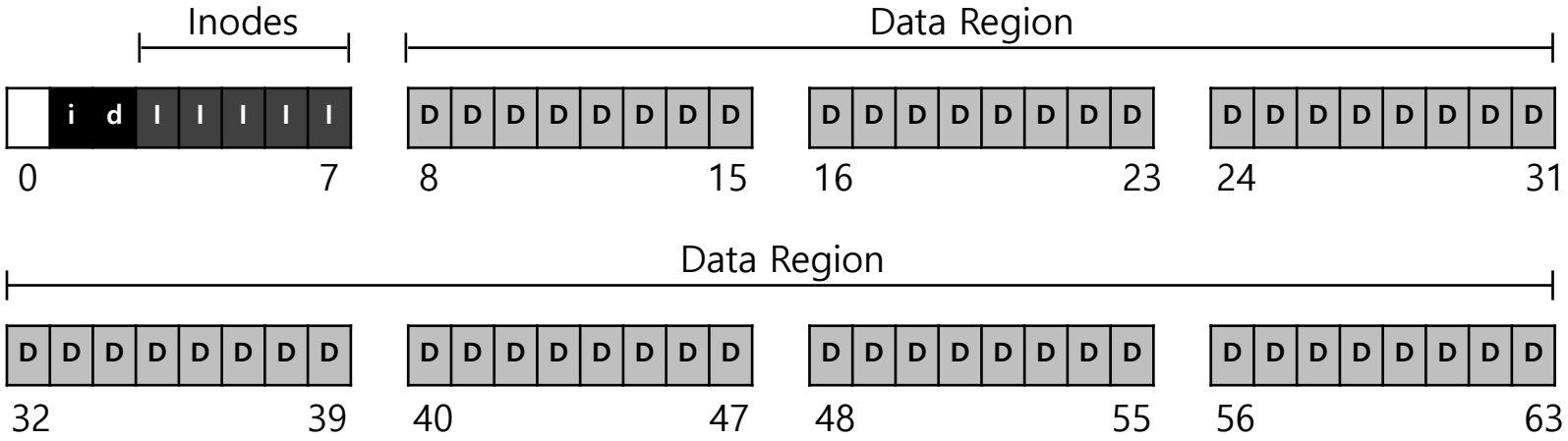


- Dosya sistemi, ayrıca, bir dosyanın hangi veri bloklarını içerdiğini, dosyanın boyutunu, sahibini vb. izlemek zorundadır → i-düğüm (i-node)

Bu i-düğümleri dosya sisteminde nasıl saklarız?

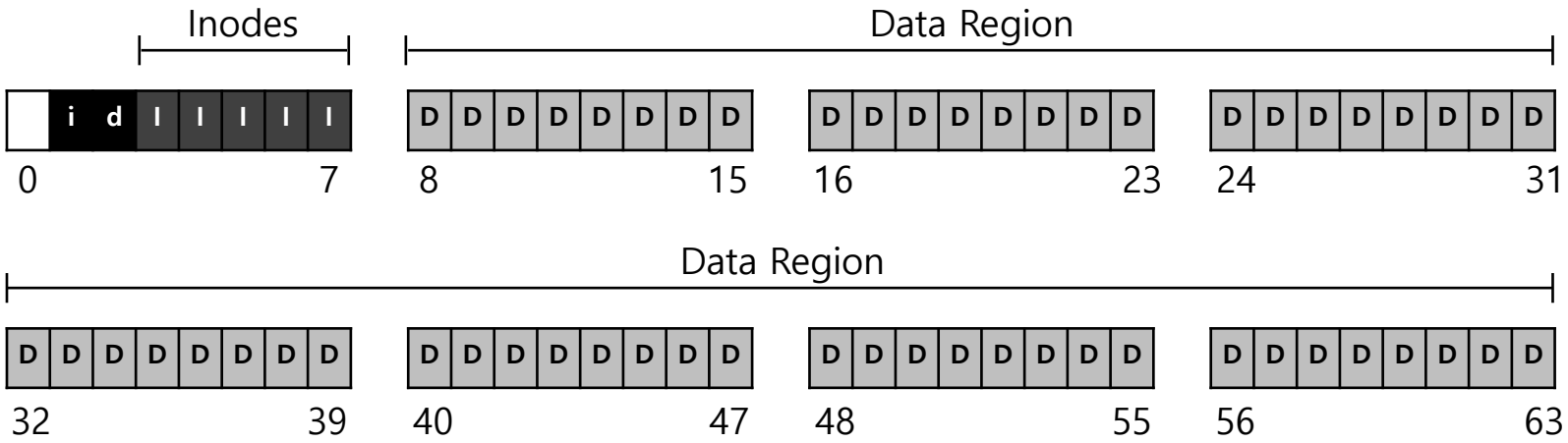
İ-düğüm Tablosu

- i-düğüm tablosu (inode table) için de yer ayırmak gerekir.
- Örnek: 3 ve 7 arasındaki bloklar.
  - i-düğüm büyüklüğü: 256 B
  - 4-KB'lık bir blok 16 i-düğüm tutabilir.
  - Dosya sisteminde toplam 80 i-düğüm vardır (maksimum dosya sayısı: 80)



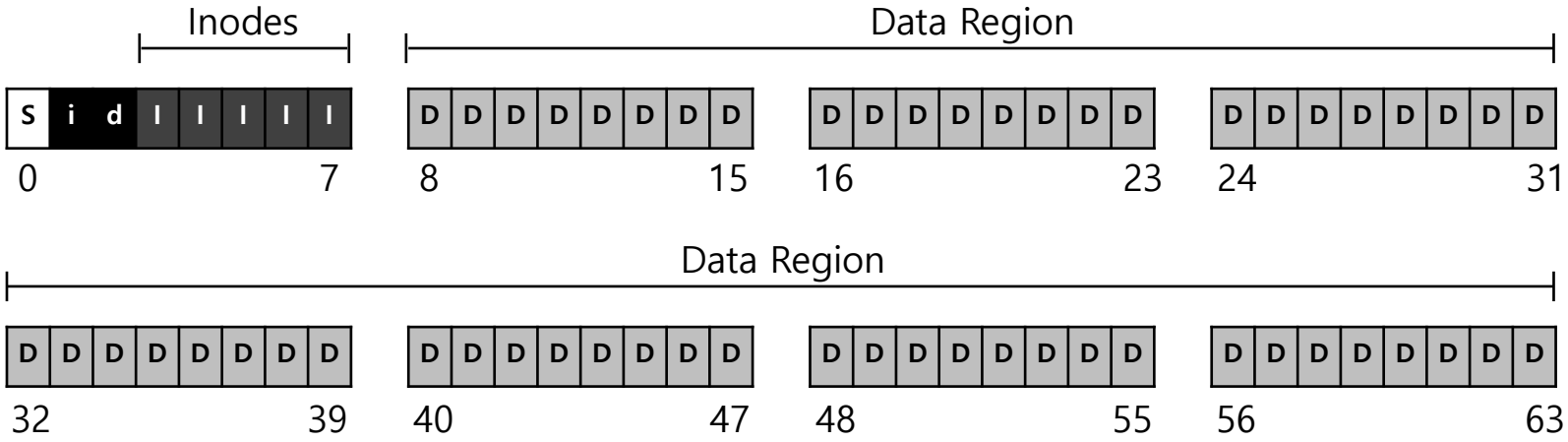
Tahsis Yapıları

- Tahsis yapıları, i-düğümlerin veya veri bloklarının boş veya tahsis edilmiş olup olmadığını izlemek içindir.
- Bitmap kullanılabilir, her bit, boş (0) veya kullanımda (1) olduğunu gösterir.
  - veri bitmap: veri bölgesi için
  - i-düğüm bitmap: i-düğüm tablosu için



Süper blok

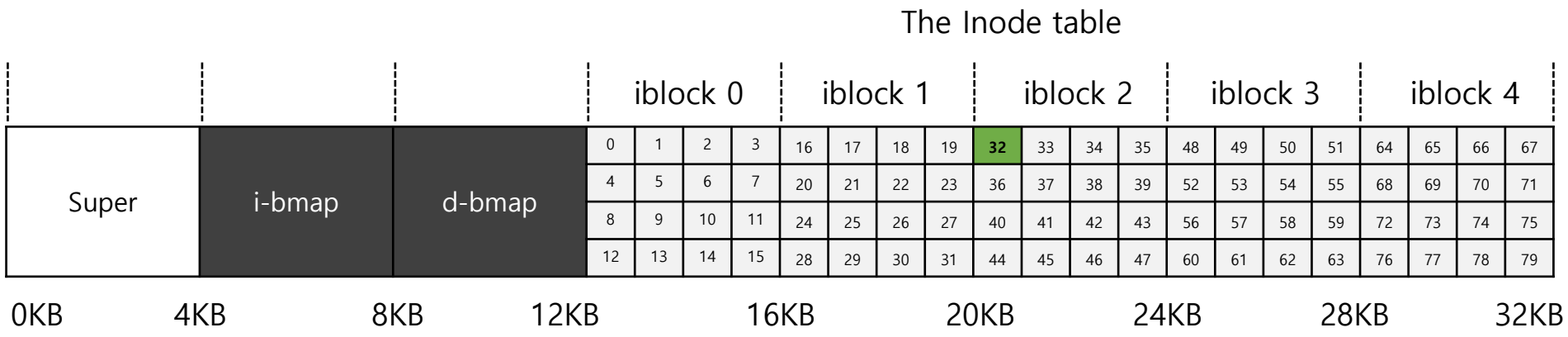
- Süper blok, belirli bir dosya sistemi için genel bilgileri içerir.
- Örnek: i-düğüm sayısı, i-düğüm tablosunun başlangıç konumu, vb.



- Bu nedenle, bir dosya sistemi kurulurken, işletim sistemi çeşitli bilgileri ilklendirmek için önce süper bloğu okuyacaktır.

İ-Düğüm Numarası

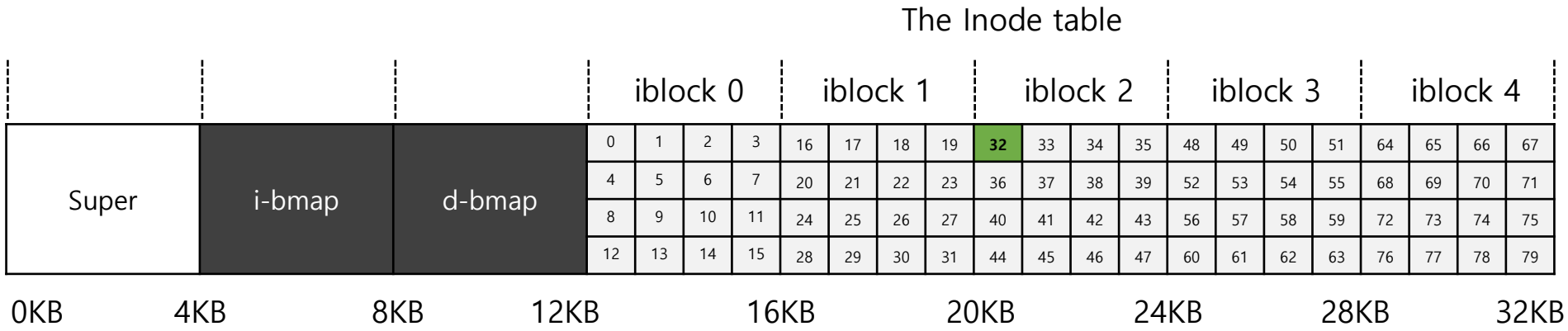
- Her i-düğüm, i-düğüm numarası ile adreslenir.
- Bu numaraya göre, dosya sistemi i-düğüm'ün diskte nerede olduğunu hesaplar.
  - Örnek: i-düğüm numarası: 32, i-düğüm uzaklığını hesapla (32 x büyüklük (i-düğüm)) (256 bayt) = 8192 = 8K
  - i-düğüm tablosunun başlangıç adresini ekle: (12 KB) + (8 KB) = 20 KB



İ-Düğüm ve Sektör No İlişkisi

- Disk; bayt adreslenebilir değil, sektör adreslenebilirdir.
- Disk, çok sayıda adreslenebilir sektörden oluşur (ör: 512 baytlık sektörler)
- Örnek: i-düğüm numarası: 32

```
blk: (i-node number * sizeof(inode))/blocksize = (32*256)/4096 = 2
sector:(blk*blocksize)+inodeStartAddr)/sectorsize=((2*4KB) +12 KB)/512 = 40
```



Örnek bir İ-Düğüm

Size	Name	What is this inode field for?
2	mode	can this file be read/written/executed?
2	uid	who owns this file?
4	size	how many bytes are in this file?
4	time	what time was this file last accessed?
4	ctime	what time was this file created?
4	mtime	what time was this file last modified?
4	dtime	what time was this inode deleted?
4	gid	which group does this file belong to?
2	links_count	how many hard links are there to this file?
2	blocks	how many blocks have been allocated to this file?
4	flags	how should ext2 use this inode?
4	osd1	an OS-dependent field
60	block	a set of disk pointers (15 total)
4	generation	file version (used by NFS)
4	file_acl	a new permissions model beyond mode bits
4	dir_acl	called access control lists
4	faddr	an unsupported field
12	i_osd2	another OS-dependent field

The EXT2 Inode

Çok Seviyeli İndeks

- Daha büyük dosyaları desteklemek için çok seviyeli indeks kullanılabilir.
- Dolaylı işaretçi (Indirect pointer), daha fazla işaretçi içeren bir bloğa işaret eder.
- Bu durumda, i-düğüm, sabit sayıda doğrudan işaretçiye (12) ve tek bir dolaylı işaretçiye sahiptir.
- Bir dosya yeterince büyürse, bir dolaylı işaretçi bloğu tahsis edilir, dolaylı işaretçi bilgisi de i-düğüm'de yer alır.
- Maksimum dosya büyüklüğü hesaplaması:

$$(12 + 1024) \text{ x } 4 \text{ K or } 4144 \text{ KB}$$

Çok Seviyeli İndeks (Devam)

- Çift dolaylı işaretçi, dolaylı bloklar içeren bir bloğa işaret eder.
- Ek 1024 x 1024 = 1 milyon 4KB blokla bir dosyanın daha da büyümesi mümkün olur.
  - Örnek: on iki doğrudan işaretçi, bir tek ve bir çift dolaylı blok → 4GB üzerinde maksimum dosya boyutu: (12+1024+1024^2) x 4KB
  - Üçlü dolaylı işaretçi de, çift dolaylı blok içeren bir bloğa işaret edebilecektir.
- Çoğu dosya sistemi çok seviyeli indeks yöntemini kullanır.
  - Örnek: Linux EXT2, EXT3, NetApp'ın WAFL, Unix dosya sistemi.
- Linux EXT4 ise işaretçiler yerine extent yöntemini kullanır.

Dosya İstatistikleri

Dosyaların çoğu küçüktür	**Kabaca 2K en yaygın boyuttur**
Ortalama dosya boyutu artmaktadır	**Neredeyse 200K ortalamadır**
Baytların çoğu büyük dosyalar için harcanır	**Birkaç büyük dosya disk alanının çoğunu kullanır**
Dosya sistemleri çok sayıda dosya içerir	**Ortalama olarak yaklaşık 100.000**
Dosya sistemlerinin kabaca yarısı doludur	**Diskler büyüse bile dosya sistemi -%50 dolu kalır**
Dizinler genellikle küçüktür	**Çoğunun birkaç girdisi vardır; 20 veya daha az**

Dosya Sistemi İstatistik Özeti

Dizinler

- Dizin, (girdi adı, inode numarası) çiftlerinin bir listesini içerir.
- Her dizinde en az iki dosya bulunur: geçerli dizin için . "nokta" ve üst dizin için .. "çift nokta«
- Örneğin, bir dizin üç dosyaya (`foo, bar, foobar`) sahip ise:

inum	reclen	strlen	name
5	4	2	.
2	4	3	..
12	4	4	foo
13	4	4	bar
24	8	7	foobar

Boş Alan Yönetimi

- Dosya sistemi, hangi i-düğümlerinin ve veri bloklarının boş olup olmadığını izler.
- Boş alanı yönetmek için iki basit bitmap'imiz vardır.
- Dosya yeni oluşturulduğunda, i-düğüm bitmap aranarak i-düğüm ayrılır ve diskteki bitmap güncellenir.
- Tahsis politikası, bitişik blokların tahsisini genelde tercih eder. Niye?

Erişim Yolları: Diskten Dosya Okuma

`open("/foo/bar", O_RDONLY):`

- Yol adını incele ve böylece istenen inode'u bulun.
- Dosya sisteminin kökünden başlamak gerekir (/).
- Çoğu Unix dosya sisteminde kök inode numarası 2'dir.
- Dosya sistemi, 2 numaralı inode'u içeren bloğu okur.
- Veri bloklarına olan işaretçiyi bulmak için kökün içeriğine bakılır.
- Bir veya daha fazla dizin veri bloğu okunarak “foo” dizinine ulaşılacaktır.
  - İstenen inode'a ("bar") ulaşılana kadar yol adı yinelemeli olarak incelenir.
  - İzinler kontrol edilir
  - Çağrıyı yapan işlem için bir dosya tanıtıcı tahsis edilir ve dosya tanımlayıcıyı kullanıcıya döndürülür.

Erişim Yolları: Diskten Dosya Okuma (Devam)

`read()`:

- Bloğun yerini bulmak için inode'a başvurarak dosyanın ilk bloğunu oku.
- İnode'u yeni bir son erişim zamanı ile güncelle.
- Dosya tanıtıcısı yardımı dosya ofseti için bellek içi açık dosya tablosunu güncelle

`close()`:

- Dosya tanıtıcısı tahsisini kaldır
- Dosya sisteminin gerçekten yapması gereken tek şey budur. Disk G/Ç'leri gerçekleşmez.

Erişim Yolları: Diskten Dosya Okuma (Devam)

	data bitmap	inode bitmap	root inode	foo inode	bar inode	root data	foo data	bar data[0]	bar data[1]	bar data[2]
open(bar)			read			read				
				read						
							read			
					read					
read()					read					
								read		
					write					
read()					read					
									read	
					write					
read()					read					
										read
					write					

Dosya Okuma Zaman Çizelgesi (Süre, Aşağıya Doğru İlerler)

Erişim Yolları: Diske Dosya Yazma

`write()` (dosyayı yeni içeriklerle güncellemek için):

- Dosya bir blok tahsis edebilir (mevcut bir bloğun üzerine yazılmıyorsa).
- Veri bloğunu ve veri bitmap'ini güncellemeniz gerekir.
- Beş G/Ç üretir:
  - veri bitmap'ini okumak için
  - bu bitmap'i yazmak için (yeni durumunu diske yansıtmak için)
  - inode'u okumak ve sonra yazmak için iki tane daha
  - bloğun kendisini yazmak için
- Dosya oluşturmak için, dosyayı içeren dizinde de alan ayırmak gerekecektir.

→ Çok yüksek G/Ç trafiğine neden olur.

Erişim Yolları: Diske Dosya Yazma (Devam)

	data bitmap	inode bitmap	root inode	foo inode	bar inode	root data	foo data	bar data[0]	bar data[1]	bar data[2]
create (/foo/bar)			read			read				
				read			read			
		read write								
							write			
					read write					
				write						
write()					read					
	read write							write		
					write					
write()					read					
	read write								write	
					write					
write()					read					
	read write									write
					write					

Dosya Oluşturma Zaman Çizelgesi (Süre, Aşağıya Doğru İlerler)

Önbellek (Caching) ve Arabellek (Buffering)

- Dosyaları okumak ve yazmak pahalıdır ve birçok G/Ç gerektirir.
- Örneğin, uzun bir yol adı (/1/2/3/…./100/file.txt) var ise herbir dizinin inode'unu ve en son dosya verilerini okumak gerekir.
  - Sadece dosyayı açmak için yüzlerce okuma!
- G/Ç trafiğini azaltmak için, dosya sistemleri, verileri önbelleğe almak için agresif bir şekilde sistem belleğini (DRAM) kullanır.
- İlk dosya sistemleri, popüler blokları tutmak için sabit boyutlu önbellek kullanmıştır.
  - Belleğin statik olarak bölümlenmesi israfa neden olabilir.
- Modem sistemler ise, dinamik bölümleme yaklaşımı ile birleşik sayfa önbelleği kullanır.
- Okuma G/Ç işlemlerinden büyük bir önbellek ile kaçınılabilir.

Önbellek (Caching) ve Arabellek (Buffering) (Devam)

- Yazma trafiğinin kalıcı olması için diske gitmesi gerekir. Bu sebeple, önbellek, yazma G/Ç'lerini azaltmaz diye düşünülebilir.
- Öte yandan, Dosya Sistemi, yazma performansı avantajları elde etmek için yazma arabelleğini kullanır:
  - Yazma işlemlerini geciktirerek dosya sistemi güncellemeleri daha küçük bir G/Ç kümesinde toplar.
  - Bir dizi yazma işlemini bellekte arabelleğe alarak, dosya sistemi daha sonra sonraki G/Ç'leri daha verimli bir şekilde planlayabilir (disk scheduling).
  - Yazmak tamamen gereksiz hale gelebilir (silinme durumu)
- Bazı uygulamalar, `fsync()` sistem çağrısı ile verilerin diske yazılmasını zorlar veya doğrudan G/Ç yapar.