Penn Engineering

ESE

Lecture #11 – Storage/Filesystems

ESE 150 – Digital Audio Basics

Based on slides ©2009—2018 DeHon
Additional Material © 2014 Farmer

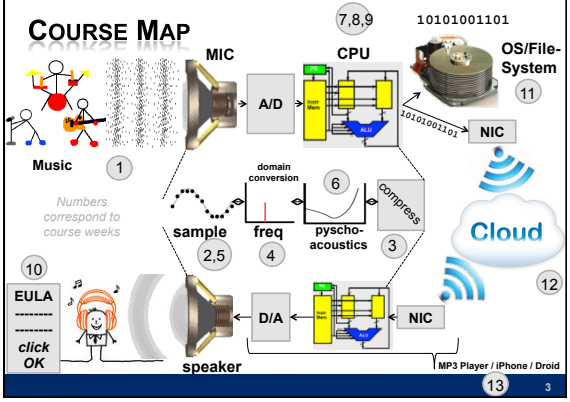
1

Lecture Topics

- Where are we on course map?
- Overview of Today's Lecture
- How/Where do we store our digital music?
  - Persistent Storage Technology
  - Filesystems
    - Abstraction 1 : files/directories
    - Abstraction 2: b-nodes/i-nodes
    - Abstraction 3: filesystems
- Next Lab

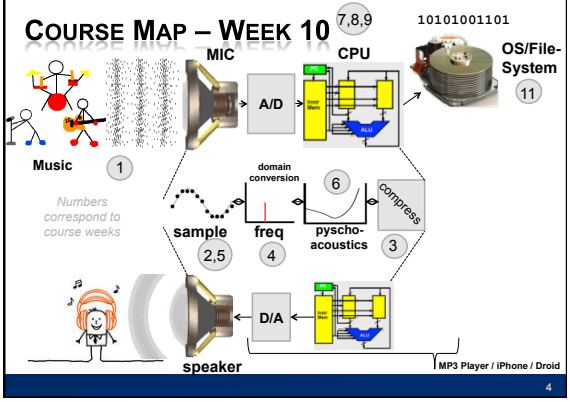
2

Course Map



3

Course Map – Week 10



4

What We'll Cover Today…

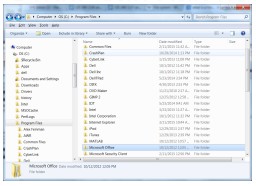
10101001101

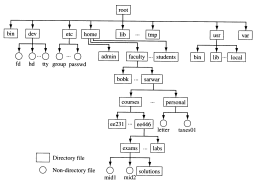
The mighty File System!

- Last Lecture…
  - We discussed the idea of an OS
    - Virtualizes Hardware
  - Key part of any OS is its filesystem…we'll talk about that today
  - From floppy disk/hard disk/compact disc/flash drive… "virtually" the same!

5

Store and Find Data





- MP3 encoded Songs on Smart Phone
- Assignments on your laptop
- Programs on eniac

6

Persistent Storage Technology

Flash-drives / **Hard-drives**

7

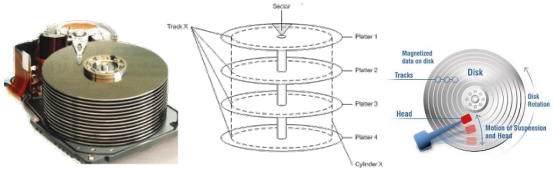
Persistent Storage

- **Persistent Storage vs. non-persistent storage**
  - We've discussed "memory" in a computer (regFile/cache/RAM)
    - This is considered non-persistent storage
    - Turn off computer, information is lost
  - Persistent Storage – long term storage
    - Information/data stored across on/off cycles of the machine
    - Power goes off, data doesn't disappear!
- **Persistent Storage Technology**
  - Many different kinds:
    - Flash drives/SSD drives/Floppy disk/Hard drive/compact discs/ etc.

8

Hard Drives / Hard Disks (HD)

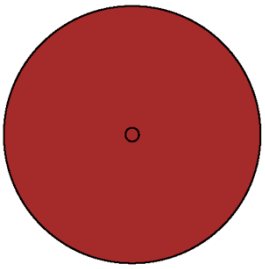
- **A collection of metallic "platters"**
- **Each platter covered with magnetic material**
- **Magnetic charge 'stores' 1 bit of information**
- **Can vary charge across platter!**



9

Hard Disk

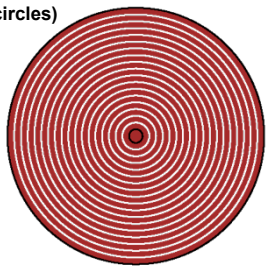
- **Disk with magnetic material on its surface**



10

Hard Disk

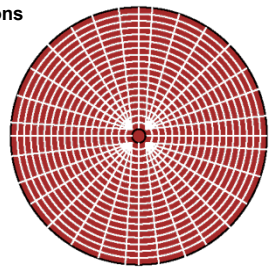
- **Disk with magnetic material on its surface**
- **Divided into tracks (circles)**
- **Modern disks**
  - 300,000 TPI
  - TPI = Tracks Per Inch



11

Hard Disk

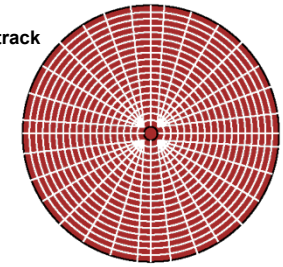
- **Disk with magnetic material on its surface**
- **Divided into bit regions**
- **Modern disks**
  - 1.5M BPI
  - BPI= bits per inch



12

HARD DISK

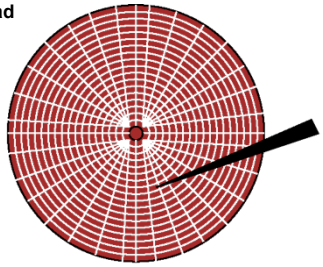
- Each bit located at a position (R,Θ)
- R = select track
- Θ = select bit from track
- disk spins
  - Traces through Θ



13

HARD DISK

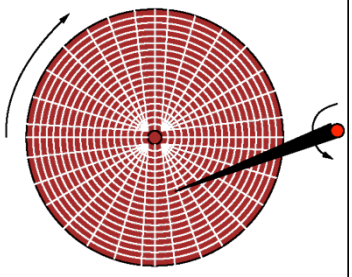
- Each bit located at a position (R,Θ)
- Add arm to move head



14

HARD DISK

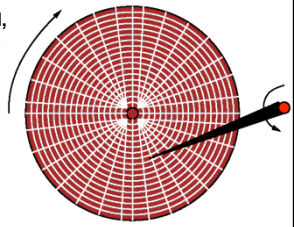
- Each bit located at a position (R,Θ)
- Head arm moves
  - Varies R



15

DISK BANDWIDTH (PRECLASS 1, 2)

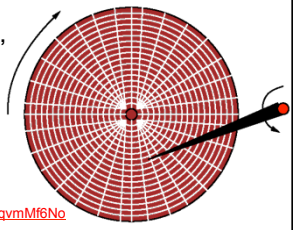
- Typical Disk speed
  - 15,000 RPM
  - Time for revolution in seconds?
- At R=1 inch and 1.5M BPI,
  - How many bits in revolution?
  - How many bits/second?



16

DISK BANDWIDTH

- Typical Disk speed?
  - 15,000 RPM
  - One rotation every
    - 60s/15,000=4ms
- At R=1 inch and 1.5M BPI,
- How many bits/second?
  - 2 π × 1 in × 1.5MB/in / 4ms
  - 9Mbits/4ms = 2.25Gb/s
    - ≈ 280 MB/s

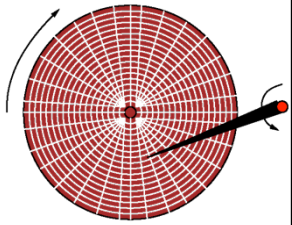


https://www.youtube.com/watch?v=3owqvmMf6No

17

DISK ACCESS TIME

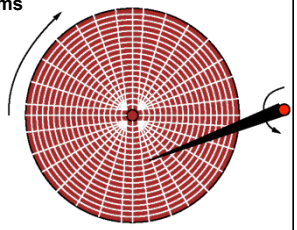
- **Move head in R?**
  - Also a few milliseconds: 6
- **How long to access a random bit?**



18

Disk Access Time

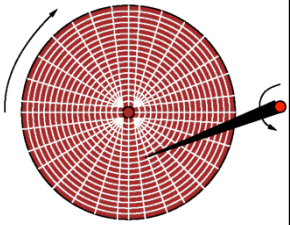
- **Move head in R?**
  - Also a few milliseconds: 6
- **Typical Data access: ~10ms**
  - *E.g.* 4ms rotate +6ms seek
- **Random bit throughput (b/s) ?**
- ***Note: CD-ROMs similar mechanism***



19

Disk Read/Write Performance

- **Typical Data access: ~10ms**
  - *E.g.* 4ms rotate +6ms seek
- **Throughput reading**
  - 4KB sequential chunk
  - 1MB sequential chunk



20

Throughput and Implications

- **Disk throughput and access time**
  - 10ms latency
  - 280MB/s throughput (~1B/4ns)
- **Observations?**
  - Throughput faster than access time
  - 10ms seek → Random bit access 100b/s
  - Sequential access 280MB/s
- **Conclude:**
  - **Want to exploit sequential access!**
    - **Read blocks of data**

21

Seagate 2.5" Disk Drive

Specifications	160GB[1]
Model Number	ST91608220AS
Interface Options	SATA 1.5Gb/s
Performance	
Transfer Rate	
Sustained Internal (MB/s)	44
Maximum External (MB/s)	150
Flash Memory (MB)	256
Cache, Multisegmented (MB)	8
Average Seek (ms)	12.5
Average Latency (ms)	5.6
Performance Level	5400

5400 RPM

http://www.seagate.com/docs/pdf/datasheet/disk/ds_momentus_5400_psd.pdf

22

Persistent Storage Technology

Flash-drives / Hard-drives

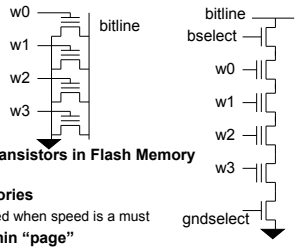
23

Flash Memory

- **A little like memory circuits we have learned about…**
  - Except it is non-volatile or simply…persistent storage
  - Data won't go away when power is turned off
  - Based on the "floating gate" transistor
- **Today's Examples**
  - Persistent storage in your MP3 player, cell phone
    - FYI: first iPod had a hard disk…
  - USB Flash drive
  - Solid-State Disk (SSD)

24

FLASH MEMORY



- **Two ways to configure FG transistors in Flash Memory**
  - NOR/NAND
- **NOR -- Read like other memories**
  - Fast, but not very dense…used when speed is a must
- **NAND – Sequential read within "page"**
  - Denser than NOR, but slower, use when area is a must
- **Can only "erase" in blocks**
  - 4KB, 64KB→256KB
- **Once erased can write byte (page) at a time**
  - Write time variable
  - Typically need feedback to sense when written

25

SAMSUNG 256MX8 NAND FLASH

Parameter		Symbol	Min	Typ	Max	Unit
Program Time		tPROG(1)	-	200	500	µs
Dummy Busy Time for Multi Plane Program		tDBSY		1	10	µs
Number of Partial Program Cycles in the Same Page	Main Array	Nop	-	-	1	cycle
	Spare Array		-	-	2	cycle
Block Erase Time		tBERS	-	2	3	ms
$\overline{RE}$ Pulse Width		tRP	25		-	ns
$\overline{WE}$ High to Busy		tWB	-		100	ns
Read Cycle Time		tRC	50		-	ns
$\overline{CE}$ Access Time		tCEA	-		45	ns
$\overline{RE}$ Access Time		tREA	-		30	ns
$\overline{RE}$ High to Output Hi-Z		tRHZ	-		30	ns
$\overline{CE}$ High to Output Hi-Z		tCHZ	-		20	ns
$\overline{RE}$ or $\overline{CE}$ High to Output hold		tOH	15		-	ns
$\overline{RE}$ High Hold Time		tREH	15		-	ns
Output Hi-Z to $\overline{RE}$ Low		tIR	0		-	ns
$\overline{WE}$ High to $\overline{RE}$ Low		tWHR	60		-	ns
Device Resetting Time(Read/Program/Erase)		tRST	-		5/10/500(1)	µs
Last $\overline{RE}$ High to Busy(at sequential read)		tRB	-		100	ns
$\overline{CE}$ High to Ready(in case of interception by $\overline{CE}$ at read)		tCRY	-		50 + tr(R/$\overline{B}$)(2)	ns
$\overline{CE}$ High Hold Time(at the last serial read)(2)		tCEH	100			ns

http://www.datasheetcatalog.com/datasheets_pdf/K/9/F/2/K9F2G08U0M.shtr

26

INTEL SOLID-STATE DRIVE (SSD)

Table 3. Maximum Sustained Read and Write Bandwidth

Access Type	MB/s
Sequential Read	up to 250
Sequential Write	up to 170

Table 4. Random Read and Write Input/Output Operations per Second (IOPS)

Access Type	IOPS
4K Read	35,000
4K Write	3,300

35,000/s x 4KB = 140MB/s

Table 5. Latency Specifications

Type	Average Latency
Read	75 µs (TYP)
Write	85 µs (TYP)
Power On to Ready	1 s

SSDhttp://download.intel.com/design/flash/nand/extreme/extreme-sata-ssd-datasheet.p

27

FLASH MEMORY

- **Similar phenomenon**
  - T ~= A + B×N
  - Large fixed expense A
    - Move in R and Θ for disk ~ 10ms
    - Erase block for flash ~ 3ms
  - High bandwidth B for sequential data
    - Both … ~100s of MB/s
- **Conclude**
  - More efficient to operate on large chunks of data

28

FILE

29

HOW ORGANIZE DATA

- **Have technology to store bits**
  - GB, TB of data
- **How do we find our data?**
  - Remembering a (R,Θ) pair could be hard
  - Even remembering a 40b address
- **How do we distinguish used/unused storage?**
  - Make sure someone doesn't overwrite our data

30

FILE

Example File: my_file.txt (14 characters)

```
I LOVE ESE150!
```

```
X49 x20 x4C x4F x56 x45
x20 x45 x53 x45 x31 x35
x30 x21
```

```
01001001 00100000
01001100 01001111
01010110 01000101 …
```

- **A file is simply a collection of bits**
  - Whether it's an ASCII file or a binary file (.docx, pdf, etc)
  - A program gives the bits meaning
  - Sequential access to bits efficient → useful to group together so can read all at once

31

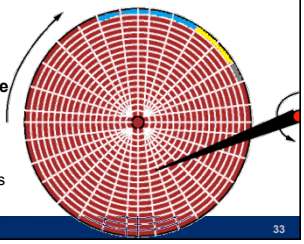
FILES → BLOCKS → PHYSICAL DISK

Mapping physical storage media to "bit/bytes/blocks"

32

HOW IS A FILE STORED?

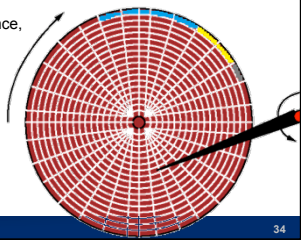
- **A file is an abstraction of the physical storage media**
  - Media "independent"
  - Don't care how technology is implemented, just read/write file!
- **Recall a file consists of:**
  - A bunch of bits
  - Logically related
  - Ex: my_report.docx, etc
- **A file system is responsible for providing this abstraction of the disk**
  - On storage media, at lowest level, file is in "blocks" of bits



33

HOW WE STORE FILES ON PHYSICAL MEDIA

- **As we write data to a disk, we do it sequentially…**
  - Data in File 1 = blue, Data in File 2 = yellow, Data in file 3 = grey…
- **Sequentially written files have huge advantage in terms of seek time…**
  - Head/needle only moves once, then disk spins and we read many bits at a time

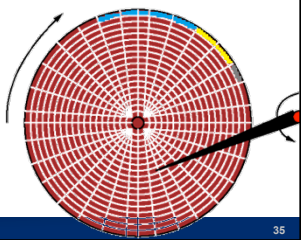


34

HOW WE STORE FILES ON PHYSICAL MEDIA

First Model

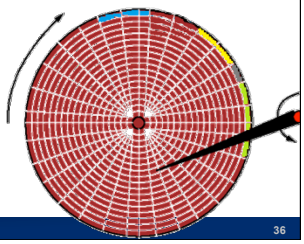
- File
  - start address
  - Length
- Allocate space sequentially
  - Keep track of first free address
- **But what happens when…**
  - we delete file?
  - Append to a file?



35

FRAGMENTATION

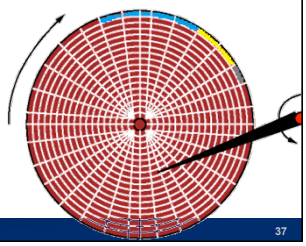
- **Let's say we shorten file1 (blue file)…**
  - We now have 2 open spaces (blocks) on disk
  - What happens when write a 4th file of 4 blocks?
  - The freespace is now "fragmented"
    - Still usable, just fragmented



36

How we store files on physical media

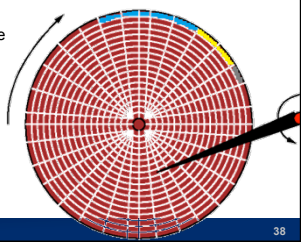
- **We could also lengthen file1 (blue file)…**
  - Ex. Need 2 **more** open spaces (blocks) to store file
  - How accommodate?



37

Fragmentation

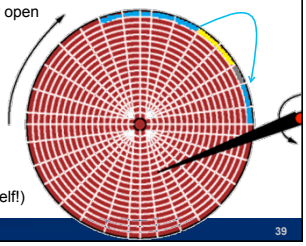
- **We could also lengthen file1 (blue file)…**
  - Ex. Need 2 **more** open spaces (blocks) to store file
  - Should we actually move it on the disk?
    - Expensive
    - Now the address change
      - How programs and people find the file when it needs to move?



38

Fragmentation

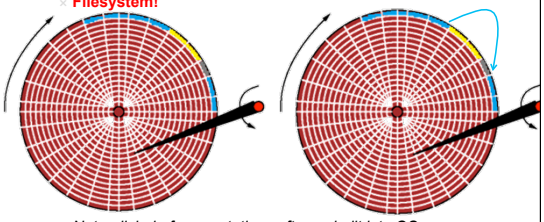
- **We could also lengthen file1 (blue file)…**
  - What if we need 2 **more** open spaces (blocks) to store file
  - Should we actually move it on the disk? *expensive…*
    - Or instead…find 2 other open spaces and create a link
  - The file is no longer sequential
    - Take longer to access but functional; still one "logical" file (abstraction of media itself!)



39

Fragmentation

- **Fragmentation is a fact of life**
  - But who is keeping track of all this fragmentation?
    - **Filesystem!**



- *Note: disk de-fragmentation software built into OSs now*

40

What is a filesystem?

41

Formation of a File / Filesystem

- **Represent File**
  - Not always just a sequence of bits on the media
- **When files become fragmented…**
  - How account for and manage fragmentation?
- **How do we know where freespace is?**
- **For that matter…**
  - How are all the files kept track of, or even found on the disk?
- **The file system**
  - Part of the operating system that organizes/keeps track of the disk
  - To understand what its keeping track of, we need to see what a file is…

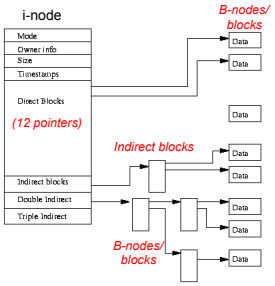
42

File Representation

- **File is not just a sequence of bits**
- **Contains some data about it**
  - Length
  - Type
  - Timestamp, ….
- **Set of pointers to the data (when large)**
  - Allowing the data to be non-sequential

43

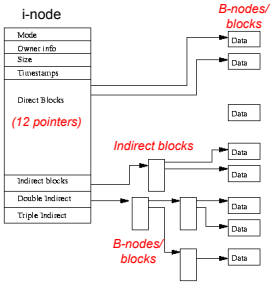
Files → i-nodes → b-nodes/blocks



- **In the Linux OS**
  - Popular filesystem named: ext3
  - Derived from original Unix File System (UFS)
  - Uses i-nodes
- **What is an i-node?**
  - A data-structure that represents a file on the storage media
  - Consists of file information:
    - Owner, size, timestamp, etc.
  - Also 15 pointers
    - 12 pointers that point directly to "blocks"
    - 3 additional pointer that point to other pointers! (indirect blocks)

44

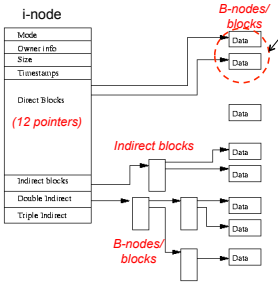
Files → i-nodes → b-nodes/blocks



- **How does it work?**
  - If a file only needs 12 blocks on disk…
    - 12 direct block pointers are used
    - Each of them point to a "bnode" or "block" of data on the disk
  - If a file requires 13 blocks…
    - 12 direct blocks are set/allocated
    - 1 indirect block is also needed

45

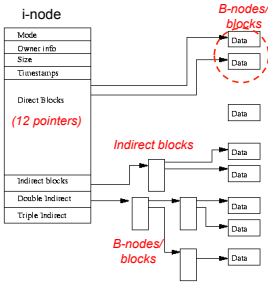
Files → i-nodes → b-nodes/blocks



- **What exactly is a b-node?**
  - Smallest unit of storage allocation
  - Fixed-size of data on the disk itself
  - Typical sizes:
    - 512 bytes for hard drives
    - 2048 bytes for CDs/DVDs
    - 4096 bytes (4kB) for todays drives
  - Smallest "addressable" unit on disk
    - Example: bnode address 76
    - *Would get 76 x 4096 = 311,296th byte on disk*
  - Address actually maps to physical address on disk
    - Example: bnode address 76
    - *R=1.012in, θ=32.07 degrees*
    - *Actual location on disk*
  - One level above media itself

46

Files → i-nodes → b-nodes/blocks

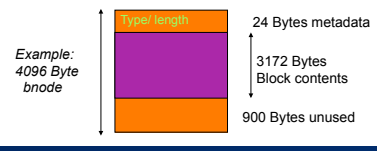


- At 4KB / Bnode
- How large a file can represent using only Direct Blocks?
  - (assuming get full 4KB block for data)

47

Files → i-nodes → b-nodes/blocks

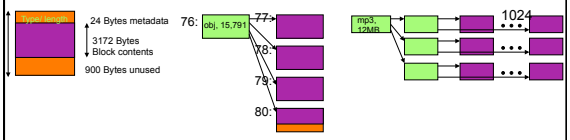
- **Bnode's structure:**
  - Bnode contains metadata (like a header)
    - Description of data to follow
    - Block type, File type, length
  - Bnode contains data itself (contents)
    - This is actual data for the file in question
    - Example shows only a 3172 block file (could use all 4072 bytes)



48

FILES → I-NODES → B-NODES/BLOCKS

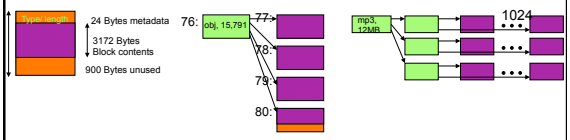
- **Bnode's structure:**
  - bnode contains metadata (like a header)
    - Description of data to follow
    - Block type, File type, length
  - Alternatively…can be table of pointers (indirect block)
    - Can be a multi-level tree if necessary (doubly/triply indirect)



49

FILES → I-NODES → B-NODES/BLOCKS

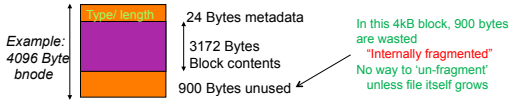
- **How help with shortening or appending to file?**



50

SIZE OF B-NODE/BLOCK REPRESENTATION

- **Why not make block sizes smaller?** ***(say 1-bit)***
  - How about 1 bit at a time? Addressing each bit wouldn't be easy…
    - 1 TB disk ~ 9 trillion bits…9 trillion addresses!
  - Most times files are larger than just a few bits
- **Why not make block sizes larger?** ***(512, 2048, 4096 bytes)***
  - Usually, we transfer "blocks" of data to/from media at a time
  - Why not go larger?
  - Remember, block size is, smallest unit of addressable space



51

B-NODE/BLOCK SIZE

- **Performance: Balance**
  - Number of blocks need to read
  - Efficiency of reading large blocks
- **Space efficiency: Balance**
  - Internal fragmentation
  - Overhead for metadata and links

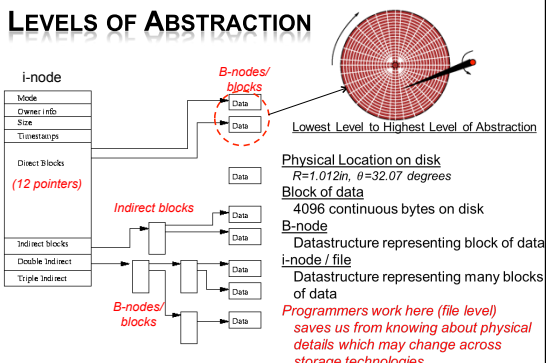
52

EXT3 FILE SIZE LIMITATIONS

- **Limits?**
  - i-node is an array of (15) 32-bit pointers
  - If block = 4 KB, first 12 pointers represent 48 KB file
  - 13th pointer – single Indirect bnode, 1024 pointers
    - Why 1024 pointers?
    - file size?
  - 14th pointer – double indirect: file size?
  - 15th pointer – triple indirect: file size?
- **Will that become a limit?**
  - (Hint, welcome ext4 file system!)

53

LEVELS OF ABSTRACTION



Physical Location on disk
R=1.012in, θ=32.07 degrees
Block of data
4096 continuous bytes on disk
B-node
Datastructure representing block of data
i-node / file
Datastructure representing many blocks of data
Programmers work here (file level) saves us from knowing about physical details which may change across storage technologies

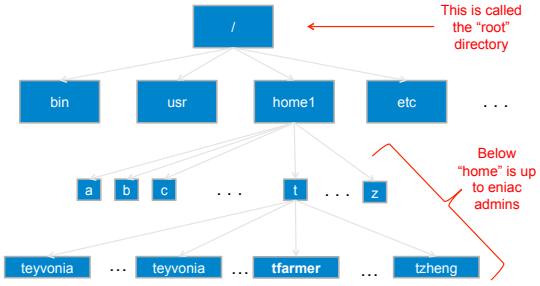
54

How keep track of files?

- **Add another file that tells us where the files are**
  - Directory
- **On ext3 filesystem…**
  - Directories are just files themselves!
  - Pairs of (Name, i-node)
  - Contain pointers to other nodes
- **…and since files can be directories**
  - Directories can contain directory files
    …which can contain directory files…
- **Leading to a directory hierarchy**

55

Unix System File Structure



ESE150 "Home directory" is **/home1/e/ese150** or simply: ~
Because the structure under "home" is typically different on all unix systems,
~ is a convention, and is always points to *your* home directory

Preclass 4

- **10^9 data items**
- **Assume at bottom of balanced tree**
- **Each tree-node has c leaves**
- **Directory i-node to hold c leaves needs**
  - 32×c Bytes
- **How many tree nodes must visit?**
- **How long to read a tree node?**
- **Time to lookup item (traverse tree)?**
  - c=2, c=10^3, c=10^9

57

Looking at inodes

- **All directories start from "root" or / directory**

```
ese150@plus:/> cd /
ees150@plus:/> ls -al
total 164
drwxr-xr-x  25 root root  4096 Mar 31 05:44 .
drwxr-xr-x  25 root root  4096 Mar 31 05:44 ..
drwxr-xr-x   2 root root  4096 Mar 31 05:44 bin
drwxr-xr-x   4 root root  4096 Mar 31 05:40 boot
drwxr-xr-x  18 root root  4020 Feb 19 13:37 dev
```

```
ese150@plus:/> ls -ali
total 164
     2 drwxr-xr-x  25 root root  4096 Mar 31 05:44 .
     2 drwxr-xr-x  25 root root  4096 Mar 31 05:44 ..
393222 drwxr-xr-x   2 root root  4096 Mar 31 05:44 bin
     2 drwxr-xr-x   4 root root  4096 Mar 31 05:40 boot
  1025 drwxr-xr-x  18 root root  4020 Feb 19 13:37 dev
```

First block of disk is inode 1, called "masterblock" / "superblock"

58

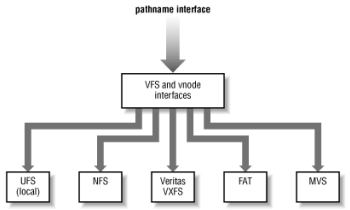
Superblock

- **For bootstrapping and file system management**
  - Each file system has a master block in a canonical location (first block on device)
  - Describes file-system type
  - Root bnode
  - Keeps track of free lists …at least the head pointers to (bnodes, blocks)
- **Corruption on superblock makes file system unreadable**
  - ➔Store backup copies on disk

59

Higher Level ex: Virtual File System (VFS)

- **Provides common interface to many different underlying filesystems!**



cd /home/d/data may live on NFS (on a server) or on local disk/CD-rom!

60

Format Disk

- **Identify all non-defective bnodes**
  - Defective blocks skipped
  - ➔ those addresses not assigned to bnodes
- **Create free bnode data structure**
- **Create superblock**

61

Disk Data Security

- **How is security enforced?**
  - OS demands credentials for login
  - User doesn't get direct access to hardware
  - OS intermediates

62

Security Caveats

- **On standard Unix/Windows setups**
  - Without the OS to provide protection, all the data is accessible
    - Sometimes good for recovery
  - On standard Unix/Windows setups
    - rm/del doesn't make the data go away
      - Also sometimes useful for recovery
  - Even format does not guarantee data overwritten
- **See:** Remembrance of Data Passed: A Study of Disk Sanitization Practices
- **What about iPhone?**

63

Security Example: File Permissions in Unix

- **Best to show with an example:**
  - Assuming you been typing in examples thus far, type:

`cd ~/ese150`

    - Puts you into the "ese150" subdirectory of your "~" home directory

`ls -al`

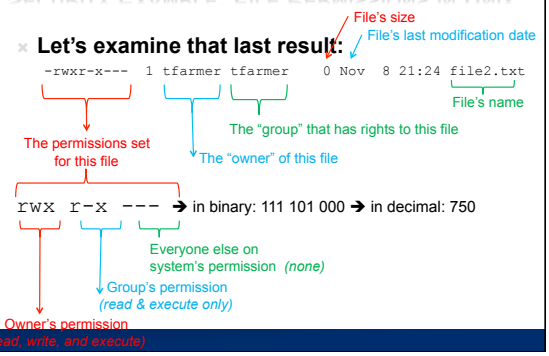
    - Passes two "arguments" to the ls command:
    - "a" (shows all files/even hidden)
    - "l" (stands for long formatted listing, you will see)

`ls -al` will return the following directory listing:

```
student@minus:~/ese250> ls -al
total 12
drwxr-x---  2 tfarmer tfarmer 4096 Nov  8 21:25 .
drwxr-x--x 36 tfarmer tfarmer 8192 Nov  8 21:24 ..
-rw-r-----  1 tfarmer tfarmer    0 Nov  8 21:25 file1.txt
-rw-r-----  1 tfarmer tfarmer    0 Nov  8 21:24 file2.txt
```

Security Example: File Permissions in Unix

- **Let's examine that last result:**



Security Example: File Permissions in Unix

- **To change permission on a file (or directory) you own:**

```
-rwxr-x---  1 tfarmer tfarmer    0 Nov  8 21:24 file2.txt
```

Type:
chmod 700 file2.txt

ls –al shows the change:

```
-rwx------  1 tfarmer tfarmer    0 Nov  8 21:24 file2.txt
```

Notice, the group lost its permission to see this file! (700, convert to binary)

- **To change ownership on a file you own:**

Type:
chown edin:ese-facu file2.txt *(changes owner & group)*

Careful, now your TA Edin, owns the file (and you don't!!):

```
-rwx------  1 edin ese-facu    0 Nov  8 21:24 file2.txt
```

OS's and their Filesystems

- **Typically an OS has a native filesystem it supports:**
  - Early PC OS: DOS (Disk Operating System)
    - File system: FAT (File Allocation Table)
  - Early Apple OS: Macintosh's "System"
    - File system: MFS (Machintosh File System)
  - Today's PC OS: Windows
    - File System: NTFS (New Technology File System)
  - Today's Apple OS: Mac OSx
    - File System: HFS+ (Hierarchical File System)
  - Today's Linux OS: Ubutnu, Debian, Fedora, Red Hat, SUSE, etc)
    - File System: EXT (Extended File System)
- **Now a days, many OS include support for more than one file system**
  - Example…CDs/DVDs have their own file system:
    - CDs: CDFS (Compact Disc File System): ISO 9660
    - DVDs: UDF (Universal Disk Format)

67

Big Ideas…Filesystem

- **Filesystem**
  - Responsible for governing/organizing the persistent storage media (harddrive/flash/etc)
  - Provides a logical/common abstraction of the media to the OS and eventually the programmer/user
  - Provides structures to keep data on disk logically organized
    - Files may be all over the place physically, but programmer would never know!
  - Provides mechanism for securing files on physical disk
    - Security not always respected without presence of OS

68

This Week In Lab

- **Lab 10: Design multi-view file system**
  - As might want for MP3 player
- **In Ketterer again**
- **Lab posted**

69

Learn More at Penn!

- **Online reading/pointers**
  - Unix File System Tutorial
  - Flash, SSD, Hard drive data sheets
  - Data found on hard drive articles
- **Courses**
  - CIS121 – efficient data structures
  - CIS380 – operating systems

70