

File System

CIT 595
Spring 2008

Disk Data Storage

- To access data in low-level format is cumbersome
 - Need to know about tracks and sectors
- OS provides a uniform logical view of information storage via the *File-System*
 - Known as high-level or logical formatting
 - Logical storage unit a.k.a *file*
 - File System provides method for storing and organizing files and the data they contain to make it easy to find and access them
- A file is nothing more than a sequence of bytes
- A file is stored in one or more *blocks* on disk
 - A block is unit of transfer between Memory and Disk

CIT 595

2

Block

- A block comprises of one or more sectors of the disk
 - Depends on block allocation method used by the OS
- OS maintains info on which blocks belong to a file
 - Data structures use logical *block numbers*
 - Starting at 0 and running through consecutive integers up to some maximum
- Logical block number is converted into *physical disk address*
 - i.e. Physical Disk Address -> track/cylinder, surface, sector
 - Conversion is performed by the device driver which outputs low-level hardware specific instructions to disk controller

CIT 595

3

Disk Partitioning

- Is the creation of logical divisions (isolated sections) of a hard disk
 - Partitioned into one or more groups of cylinders
 - Gives appearance of more than one hard drive e.g. C drive, D drive
- Disk can be prepared for use, or even dedicated to different uses
 - Allows one to apply OS-specific logical formatting per partition
 - Allow more than one OS per partition
- Encapsulate your data
 - Corruption is local to a partition, so stand to lose only some of your data if an accident occurs
- Also improve disk efficiency
 - E.g. Windows depending on the File System, assigns block sizes based on partition size
 - Larger disk size implies larger block size that can result in wasted space

CIT 595

4

Disk Layout

- Master Boot Block (MBR)
 - A known sector on disk is used to boot the computer
- Partition Table
 - Starting and Ending Addressing of each partition
 - One of the partitions is marked as active
- Disk Partitions
 - Boot Block
 - Contains a program where the OS resides in the partition
 - Even if no OS for that partition, a boot block is set aside
 - Partition Control Block
 - Magic number to denote what kind of file system e.g. FAT, NTFS, UFS
 - Number of Blocks and the size of blocks
 - Data Structure for Free Blocks, Directory, File Allocation Info, and the files itself

CIT 595

5

Boot Block

- Upon boot, the BIOS (Basic Input/Output System) program is executed
 - Is stored in ROM that gets executed on startup
 - Conducts a basic hardware check, to determine whether all of the attachments are present and working
 - Then executes program located in the MBR
- MBR program locates active partition, reads in its first block i.e. boot block and execute it
- Boot block loads the OS and then the OS takes over

CIT 595

6

Directory

- Directory provides information needed to find the disk data blocks of a file
- A directory is nothing more than a *file itself*, except it is specially structured
- Each entry in the directory contains
 - Symbolic file name
 - Indicate where file is located on disk
 - Depends on the Block Allocation Scheme for files
 - File attributes: size, access-information, time and date
- Root directory contains information about where each sub-directory is located
 - Always contained at fixed within a partition

CIT 595

7

Block Allocation for files

- How do we allocate files on disk so that
 - Space is utilized effectively
 - Allow accesses that are
 - *Random/Direct*: each block can be accessed independently of other
 - Quick – try to minimize seek time
- Two kinds of allocation
 - Contiguous
 - Linked
 - File Allocation Table (FAT)
 - File Index

CIT 595

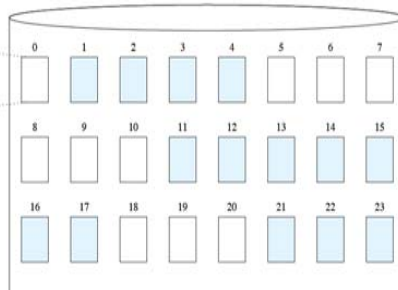
8

Contiguous Allocation

- Requires files to occupy a set of contiguous blocks

Directory

File	Start	Size
CIT593doc	5	6
..		
CIT595doc	18	3



CIT 595

9

Contiguous Allocation (contd..)

Advantage

- Random access is possible
- If all blocks of one file fit physically on track then
 - Disk seek is minimized after the starting block

Disadvantage

- Finding contiguous space for file
- Suffers from *external fragmentation* i.e. disk is broken into pieces
 - A new request may not be able to fit into any of the pieces
 - Need to compact all free space into one continuous space - disk defragment

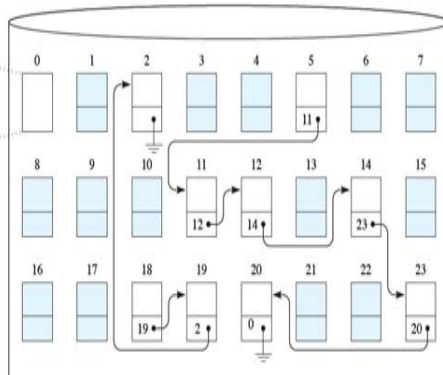
CIT 595

10

Linked Allocation

Directory

File	Start
CIT595doc	5
CIT595doc	18



CIT 595

11

Linked Allocation (contd..)

Advantage:

- No external fragmentation

Disadvantage:

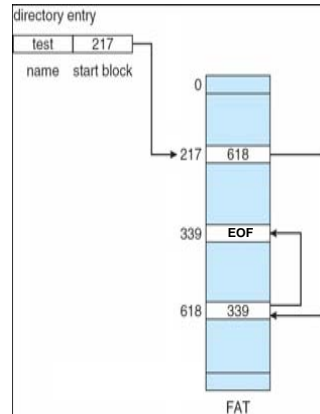
- Random Access is not possible
- Disk seek time is increased for a file as the blocks are now scattered all over
- Pointer storage overhead
 - Some of the storage in each block devoted to the pointer

CIT 595

12

File Allocation Table (FAT)

- One implementation of Linked Allocation scheme
 - Pointers to data blocks are stored in FAT
- FAT is an array – one entry for each disk block
 - Indexed by the block number
 - Each array entry is a pointer to the next block
- FAT is also kept on disk
 - One per partition
 - Tells OS parts of the disk are currently used by files, and which are free for use



CIT 595

13

FAT (contd..)

- Random Access is possible
- Does not solve the seek problem
 - Considerable arm movement to read the FAT and actual file blocks
- Seek time can be minimized by caching the FAT
 - Store in main memory
 - No disk I/O needed for FAT lookup – only block requests are sent to disk
- Caching FAT also has another overhead
 - E.g. 5 GB disk partition, 1K block size - 5M block entries with each entry size of 3 bytes, hence FAT occupies 15 MB of Memory

CIT 595

14

FAT (contd..)

- Disadvantage: If drives have big sizes, then the block/cluster sizes increase leading to wasted space per block
- FAT12 was used in floppy disk (1.44 MB)
 - Each entry stores 12 bits, so number of blocks restricted to 2^{12}
 - Good for fixed sizes
- FAT16
 - Advantage of FAT16 is that it is compatible across a wide variety of OS, including Windows 95/98/Me, Linux, and some versions of UNIX
 - Limitation, drives over 2GB require 64KB block sizes
- FAT32
 - Actually use only 28 bits
 - Over come FAT16 limitation, can support 1TB with block size of 4KB
 - Not recognized like FAT16

CIT 595

15

File Index Block

- Only a few files are open at any one time
 - So keep only necessary information about open files in memory to get good performance
- The *index block* groups together index information about each file individually
 - Called Inode in Unix
- The index block is an array of block numbers but it can only have a subset of block numbers
 - Less memory overhead when cached
- Updating the index is like writing to any block on disk

CIT 595

16

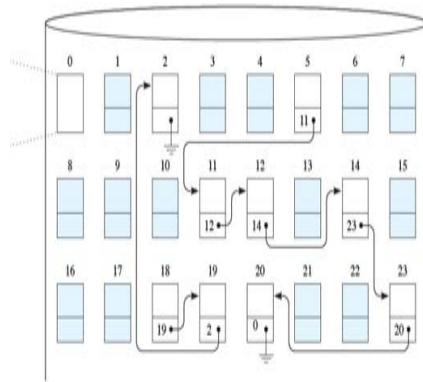
File Index

Directory

File	Index
CIT595doc	17

Block 17

18
19
2
-1
-1
-1



CIT 595

17

File Index (contd..)

- If the number of blocks need cannot fit in one index block, then apply levels of indirection

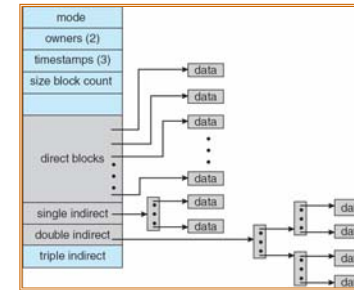


Figure 12.9
OS Concepts by
Silberschatz, Galvin &
Gagne

- Any disadvantages ?

CIT 595

18

Free-Space Management

- A free-space list is maintained that records all free blocks
- Linked-list Technique
 - Just like linked allocation – Pointer to first free block is maintained in fixed location on disk
 - Or can be accounted in part of the allocation data structure
 - E.g. in FAT, block entries that not occupied have special symbol written in them
- Grouping Technique
 - Store the address of n free blocks in the first free block
 - n – 1 entries are actually free and the last block contains the address of the next n free blocks

CIT 595

19