File Systems

Dickinson College
Computer Science 354

slides courtesy of Professor Grant Braught

Road Map

- Past:
  - ✓ What an OS is, why we have them, what they do.
  - ✓ Base hardware and support for operating systems
  - ✓ Process Management
  - ✓ Process Scheduling
  - ✓ Multi-Threading
  - ✓ Thread Synchronization
- Present:
  - ✓ File Systems
- Future:
  - ✓ Memory management
  - ✓ Protection and Security

File Systems Outline

- Hard disk structure
  - ✓ Perspective gap
- User perspective
- Programmer perspective
- Operating system perspective

Hard Disk Structure

- Hard Disks:
  - ✓ Basic hard disk controller can:
    - ❖ Read a sector (or block)
    - ❖ Write a sector (or block)
  - ✓ Sector to read/write is specified by a cylinder:head:sector (CHS) address.
  - ✓ Most disk controllers use linear block addressing (LBA).

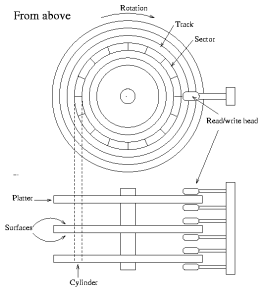


Image from: Linux System Administrators Guide
http://www.tldp.org/LDP/sag/html/hard-disk.html

Perspective Gap

- User Perspective:
  - ✓ A disk is a collection of files and directories that can be manipulated using commands.
- OS Perspective:
  - ✓ A disk is a collection of data blocks that can be manipulated via block numbers.
  - ✓ It is the job of the OS to bridge the gap between these two perspectives.

User's Perspective

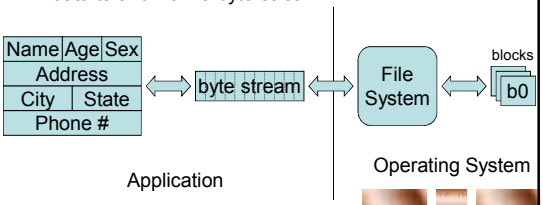
- A disk is a collection of files and directories that can be manipulated using commands.
- We'll look at:
  - ✓ Types of files supported
  - ✓ Directory structures supported
  - ✓ Information maintained
  - ✓ Protection mechanisms
  - ✓ Unix Commands

File Types

- From the user's perspective operating systems provide two basic types of files:
  - Unstructured Files
  - Structured Files

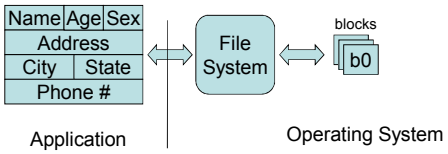
Unstructured Files

- With unstructured files, the OS provides system calls for *marshalling* streams of bytes into and out of blocks.
  - The OS is ignorant of the internal structure of files.
  - Each application must contain code to translate its data to and from a byte stream.



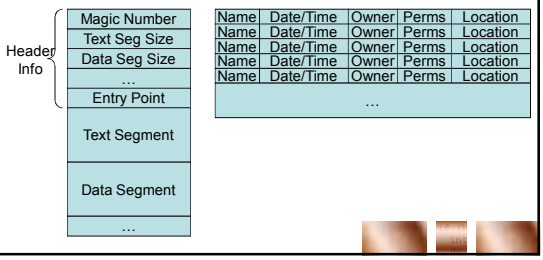
Structured Files

- With structured files the OS provides system calls for marshalling records into and out of blocks.
  - The OS may provide direct support for records or it may provide a way to associate user developed marshalling routines with a file.



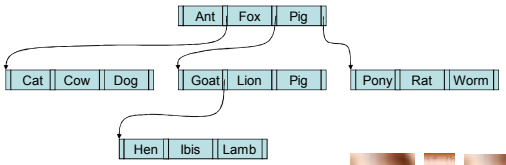
Common Structured Files

- Every OS provides at least two types of structured files:
  - Executable files
  - Directories



Tree Structured Files

- Some OS's provide structured files based on trees (a.k.a. indexed).
  - Records are requested based on key value instead of location within the file.
  - The tree structure provides rapid access to a record based on its key value.



Structured Files in the Mac OS

- In the Mac OS file system, all files have two parts:
  - Data Fork:
    - Unstructured byte stream
    - Program instructions and data are stored in the data fork
  - Resource Fork:
    - Structured collection of records recognized by the OS
    - User interface components, icons, file type and creator information is stored in the resource fork.

Binary and Text

- Files can also be classified by the way in which their data is stored:
  - Text Files
    - ASCII
      - Data is stored character by character using ASCII codes
        - 'A' is stored as 01000001 (65)
        - '1' is stored as 00110001 (49)
      - One byte per character.
        - E.g. Storing 1234567 takes 7 bytes.
    - Unicode
      - 2 bytes per character
  - Binary Files
    - Character data is stored using ASCII codes.
    - Numeric data is stored in binary representation.
      - Storing 1234567 as an int takes 4 bytes.

Random OS Humor

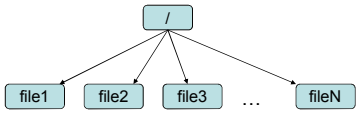
- "The box said, Win95 or better required... so I used a Mac !"
  Tim Scoff.

Directory Structures

- Different OS's support different types of directory (i.e. folder) structures:
  - Single Level
  - Hierarchical
  - Acyclic Graph
  - Graph

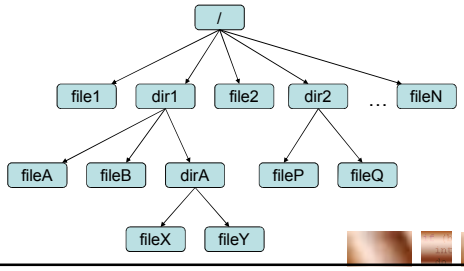
Single Level (flat) Directories

- Some early OS's supported only a flat directory structure.



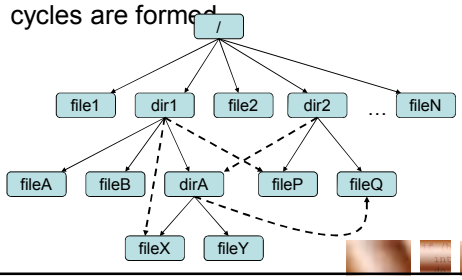
Hierarchical Directories

- Hierarchical (tree) directory structure allows for directories to be nested inside of other directories.



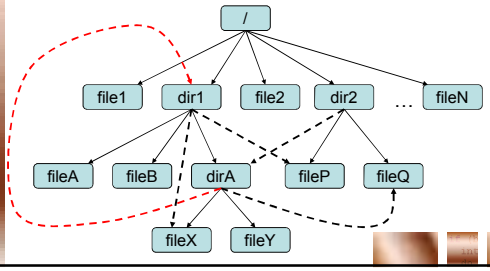
Acyclic Graph Directories

- Acyclic graph directories allow for *links* that make a directory or file appear in more than one location - so long as no cycles are formed.



Graph Directories

- A graph directory structure, removes the restriction that links may not create a cycle.



File and Directory Information

- For each file and directory, the operating system typically maintains the following information that can be seen by users:
  - ✓name, size
  - ✓creation date, creation time
  - ✓last modified date, last modified time
  - ✓owner, permissions
    - ❖Use ls -l in Unix to see some of this information.

Permissions

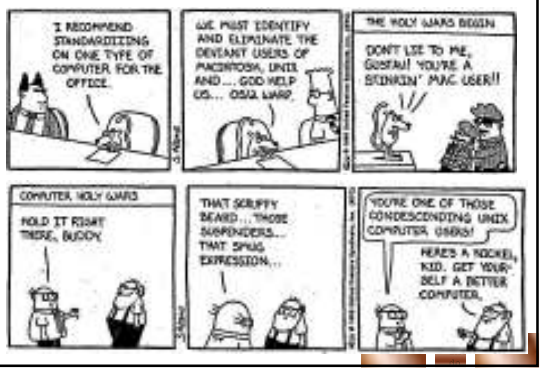
- Permissions determine which users are allowed to access a given file or directory and in what ways they may access it.
  - ✓In Unix permissions are expressed using 3 sets of 3 bits each.

	r	w	x	
❖Owner	1	1	0	(6)
❖Group	1	0	0	(4)
❖World	0	0	0	(0)

Unix File Manipulation

- In Unix users manipulate files and directories using a variety of commands:
  - ✓ls
  - ✓rm mv cp
  - ✓mkdir rmdir cd
  - ✓chmod chown
  - ✓ln symlink

Random OS Cartoon



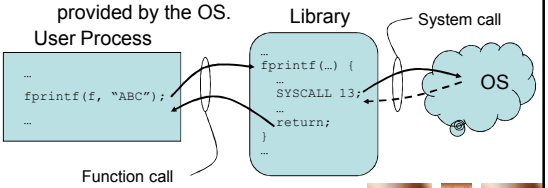
The Programmer's Perspective

Programmers Perspective

- From the programmer's perspective a disk contains files and directories that can be manipulated via library functions (or via system calls).
- We'll look at:
  - Review libraries and system calls
  - File access
  - System calls for file manipulation
  - Read/Write buffering
  - File locking
  - System calls for directory manipulation

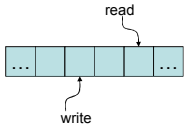
Programmer's Perspective

- The programmer typically accesses and manipulates files and directories using library functions provided by a particular programming language.
  - Those library functions are wrappers for system calls provided by the OS.



Read/Write Pointers

- When a program opens a file, two pointers are established (read and write).
  - Pointers indicate the positions in the file at which the next read or write operation will occur.



File Access Modes

- A file may be accessed in two modes:
  - Sequential Access: Read and write pointers are advanced only by file read and file write operations.
  - Random Access: Read and write pointers may be repositioned within the file at any time through the use of a special system call.

File Manipulation System Calls

- At a minimum every OS must provide system calls for the following file operations:
  - create, delete
  - open, close
  - read, write
  - seek
  - get attributes, set attributes

Read/Write Buffering

- The operating system can only read and write complete blocks of data at a time while programs read and write bytes of data via the system calls.
  - To improve performance the OS will buffer a block the first time it is accessed.

File Locking

- Having multiple processes accessing shared files can result in critical sections and race conditions.
  - Operating systems provide file locks to protect critical sections and prevent race conditions related to file access.
    - Shared vs. exclusive locks
      - Readers lock vs. Writers lock
    - Advisory vs. mandatory locks

Directory System Calls

- At a minimum every OS must provide system calls for the following directory operations:
  - open directory file, close directory file
  - read directory file
  - create entry, delete entry
  - rename entry

A Directory File

Name	Date/Time	Owner	Perms	Location
Name	Date/Time	Owner	Perms	Location
Name	Date/Time	Owner	Perms	Location
Name	Date/Time	Owner	Perms	Location
Name	Date/Time	Owner	Perms	Location
...				

The Operating System's Perspective

The OS Perspective

- The OS must provide the data structures and algorithms necessary to implement each of the system calls.
- We'll look at:
  - Internal data structures
  - Block Allocation and File Descriptors
  - Directory file records
  - Free space management
  - Disk formatting / defragmenting

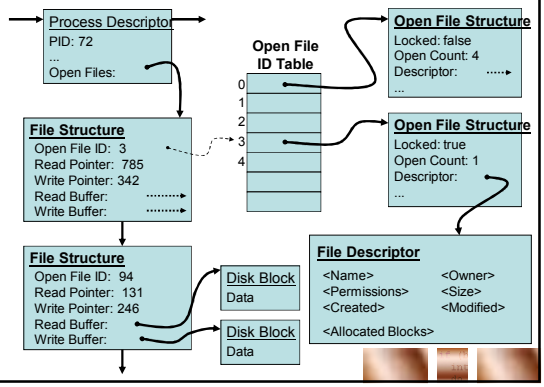
File / Dir Information

- The OS must maintain information about every file and directory.
  - For files that *are not open* the following information is sufficient:
    - Name, owner
    - Permissions, times(creation/modified)
    - Location, size
  - For files that *are open* the following additional information is necessary:
    - Which process(es) have the file open.
    - Locks
    - Read/Write pointers
    - Buffered blocks

File System Data Structures

- A file system uses four main data structures:
  - File Structure: One for each file or directory that is opened by a process.
  - Open File ID Table: An array of pointers to all of the Open File Structures. One for the entire system.
  - Open File Structure: One for each file or directory that is open.
  - File Descriptor: One for every file or directory on the disk is maintained on the disk. For each open file or directory, the file descriptor is copied into memory.

File System Data Structures



Block Allocation Schemes

- When files are created or when they grow, the OS must allocate unused block from the disk to the file.
  - Alternative schemes:
    - Contiguous Allocation
    - Linked List Allocation
    - File Allocation Tables (FAT)
    - Indexed Allocation
    - Multilevel Indexed Allocation

Evaluating Allocation Schemes

- Several factors should be considered when evaluating block allocation schemes.
  - Access Mode
    - Sequential access performance
    - Random access performance
  - Fragmentation
    - Internal fragmentation: Space that is allocated to the file but is unused.
    - External fragmentation: Space that is free but cannot be allocated to a file.

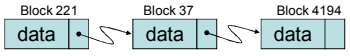
Contiguous Allocation

- With contiguous allocation, the space for a file is allocated using consecutively numbered blocks.
  - File descriptor only needs to store the starting block and the number of blocks in the file.
    - Drawbacks?
    - Benefits?
  - Modified contiguous allocation using extents.

File Descriptor:
<Name> <Owner>
etc…
<First Block>
<Number of Blocks>

Linked List Allocation

- Files are created with a single block. As files grow, the last word in each block stores the address of the next block.

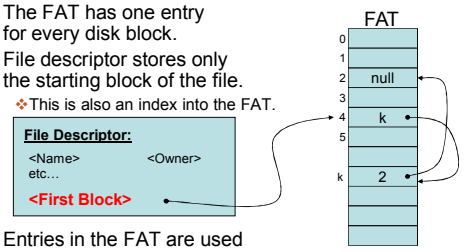


  - Benefits?
  - Drawbacks?

File Descriptor:
<Name> <Owner>
etc…
<First Block>

File Allocation Tables

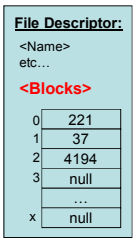
- The OS maintains a File Allocation Table (FAT) for each disk.
  - The FAT has one entry for every disk block.
  - File descriptor stores only the starting block of the file.
    - This is also an index into the FAT.



  - Entries in the FAT are used as a linked list to find the remaining blocks of the file.

Indexed Allocation

- With indexed allocation each file descriptor contains a list of the blocks making up the file.
  - ✓Drawbacks?
  - ✓Benefits?

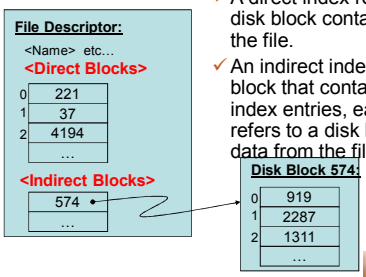


Indexed Allocation

- If we have a 16GB disk with a block size of 1KB and we want our maximum file size to be 1GB in size, how much space is required for the index entries in each file descriptor?

Multilevel Indexed Allocation

- With a multi-level index, the file descriptor contains direct and indirect indices.
  - ✓A direct index refers directly to a disk block containing data from the file.
  - ✓An indirect index refers to a disk block that contains additional index entries, each of which refers to a disk block containing data from the file.



Random Computer Humor

- "The computer allows you to make mistakes faster than any other invention, with the possible exception of handguns and tequila."

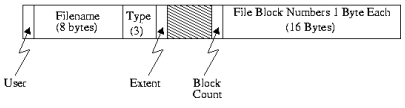
— Mitch Ratcliffe.

Directory File Records

- Each operating system defines its own structure for the records that appear within the directory files.
  - ✓Examples:
    - ❖CPM
    - ❖DOS
    - ❖Unix

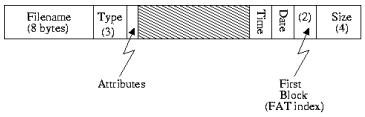
CPM Directory Records

- CPM is an old mainframe OS that used a flat directory structure.
  - ✓The first several blocks on the disk are set aside to hold the directory file.
  - ✓Every file on the disk has a record in the directory file.
  - ✓This record is also the File Descriptor in CPM.



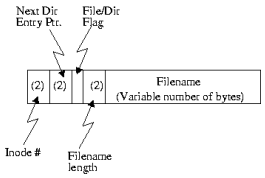
DOS Directory Records

- DOS supported a hierarchical directory structure.
  - ✓ One sector at the start of the disk is set aside to hold the directory file for the root directory.
  - ✓ DOS Directory records are also the file descriptor.
  - ✓ Each sub-directory has a block of directory entries elsewhere on the disk.



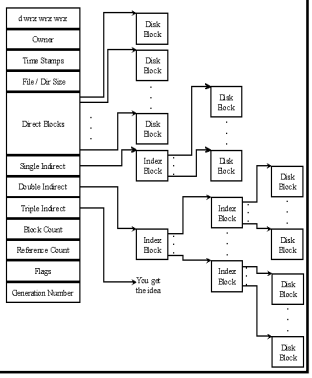
Unix Directory Records

- Unix directory entries contain only part of the information for the file descriptor.
  - ❖ The remaining information is contained in an *inode*.



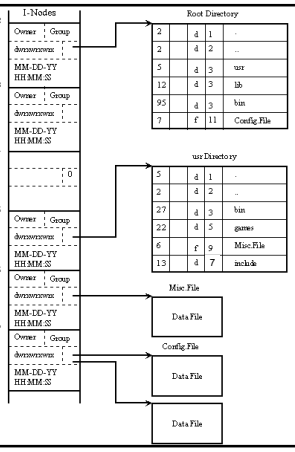
The Unix Inode

- Unix inodes use a multilevel indexed allocation scheme to track blocks allocated to a file.



Unix File System

- Inode #2 always holds information about the root directory of the disk.



Free Space Management

- The OS must also keep track of which blocks on the disk are not yet allocated to files or used for directory files (i.e. *free blocks*).
  - ✓ Several approaches:
    - ❖ Bit vector
    - ❖ Linked list
    - ❖ Indexed

Bit Vectors

- Free disk blocks can be tracked using a bit vector.
  - ✓ 00110100 00100011 11100011 1000…
    - ❖ Each 0 indicates an allocated block.
    - ❖ Each 1 indicates a free block.
  - ✓ First free block:
    - ❖ (# of 0 words) * (bits/word) + (offset of 1 in 1st non-zero word)
      - Finding offset of first 1in a word is often supported by a ML instruction.

Linked Lists

- Free space can also be managed using a linked list scheme.

  - Can be modified to be a linked list of *holes*.

Indexed

- Free space can also be tracked using an indexed approach.
  - Unix has used inodes 0 and 1 to track the free blocks on a disk.

File System Efficiency

- What are several ways that the efficiency of the original Unix file system could be improved?

Crashes and Recovery

- File system operations must update the in-memory data structures as well as the information stored on disk.
  - Allocated blocks, access times, free space list…
  - Problems can occur if the system crashes when in-memory data is out of synch with on-disk data or when separate pieces of on-disk data are inconsistent.
    - Repair tools:
      - fsck
      - Chkdsk

Random OS Humor

- When one is told "Go fsck yourself!" the meaning implied is to "go away, analyze yourself, and fix your problems"

  Wikipedia

Journaling File Systems

- In a journaling file system, all changes to the file system are written to a *journal* before applying them to the actual file system.
  - Physical journal
  - Logical journal

Miscellaneous Other Topics

- Some other topics:
  - File deleting vs. file erasing
    - File recovery
  - Disk formatting
    - Full format vs. quick format
  - Disk defragmenting
  - Others??