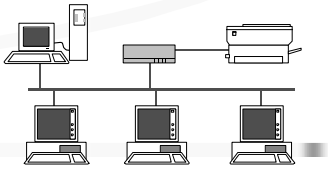


Operating Systems
600.418
The File System

Department of Computer Science
The Johns Hopkins University

The File System

Lecture 7

Reading:
Silberschatz & Galvin
chapter 10,11
Additional Reading:
Stallings
chapter 12

The File System

- Provides online storage and access to data (and programs).
- Two separate aspects:
  - **Interface to the user.**
  - **Implementation.**
- The interface exports two distinct notions:
  - **Files, each storing a collection of data.**
  - **A directory structure, organizing and providing information about all the files in the system.**

Files

- A named collection of related information that is recorded (usually) on secondary storage.
- A logical storage unit, abstracting out the physical properties of the storage device (e.g. disk, tapes, floppies).
- Can provide long term storage, persistent through:
  - **Activation of programs over time.**
  - **System reboots.**
  - **Power failures.**

File Attributes

- Name - a string of characters.
  - **Legal characters**
  - **Upper / Lower case.**
- Type - some operating systems support different types of files and handle them differently.
- Location - information on the device and location of the file data.
- Size - current size of the file (usually in bytes). Maybe also max allowed size.
- Protection - access control. Who can read, write, execute.

File Attributes (cont.)

- Various dates - creation, last modification, last access.
- Ownership - who owns this file.

- Where are these attributes maintained?
- Maintained in the directory structure.
- Each file has a record in the directory (record size can range from a few bytes to over 1KBytes).
- Where is the directory stored ?
- On the same device.
- Can consume considerable amount of memory.
- Should reflect consistent view of the files.

Operations on Files

Required Operations

- Create - finds a space in the file system and create an entry in the directory structure.
- Write - specifies the name of the file and the information to be written. The system also keeps a location pointer in the file.
- Read - specifies the name of the file and a memory buffer to store the read data. The system keeps a location pointer in the file.
- Seek - repositioning the location pointer for the file.
- Delete - releases the file space and erase the directory entry.

Additional Operations on Files

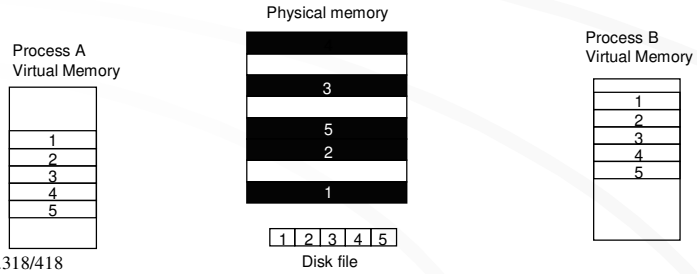
- Truncate - maintains file attributes but erases the content of the file.
- Append - writes information at the end of the file.

Additional Operations on Files

- Truncate - maintains file attributes but erases the content of the file.
- Append - writes information at the end of the file.
- Open, close - to avoid searching the directory for each of the previous operations, expediting the overall performance of file operations, the operating system provides the open and close operations that associate a handler with the file.

Memory Mapped Files

- Some operating systems allow mapping a file into memory on virtual-memory systems.
  - **Part of the virtual address space is associated with the file.**
  - **Reads and writes to that memory space are treated as reads and writes to the file.**
- Memory mapped files can be used also to share the file data among several processes.

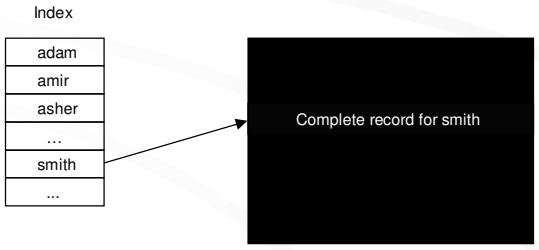


Access Methods

- Sequential Access
  - **Information in the file is processed in order, one record after the other.**
- Direct Access
  - **A file is considered as an ordered array of fixed-length logical records.**
  - **Read and write operations may access random records with no particular order.**
  - **The operation provides a record (or block) number, relative to the beginning of the file.**
  - **Can implement sequential access.**

Access Methods (cont.)

- Access using an index:
  - **Based on direct access.**
  - **Several variations exist.**
  - **Builds some sort of index file that allows fast access to the data in the file.**

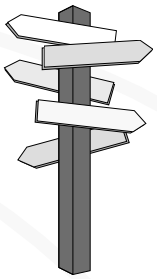


Directory Organization

- Goals:
  - **Efficiency: quickly locating the file.**
  - **Convenience: naming files in a convenient way to users.**
  - **Grouping: allowing users to group files based on users classifications.**
- Organization of the file system
  - **Virtual disks (called volumes, or minidisks, or partitions). May span a physical disk, part of a physical disk, or several disks.**
  - **Virtual disk directory.**

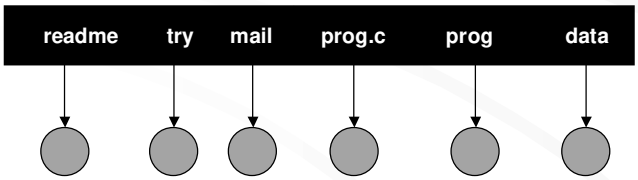
Directory Organization (cont.)

- Directory Operations:
  - **Create a file.**
  - **Delete a file.**
  - **Search for a file.**
  - **List a directory.**
  - **Rename a file.**
  - **Traverse the file system.**



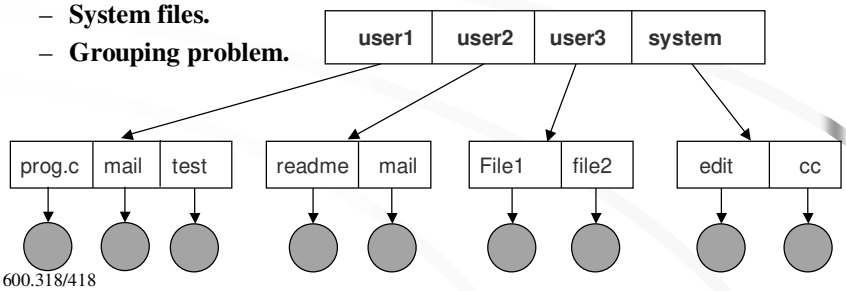
Single Level Directory

- All files are contained in the same directory.
- Easy to support and understand.
- Has significant limitations in terms of:
  - **Large number of files (naming).**
  - **Ability to support different users / topics (grouping).**



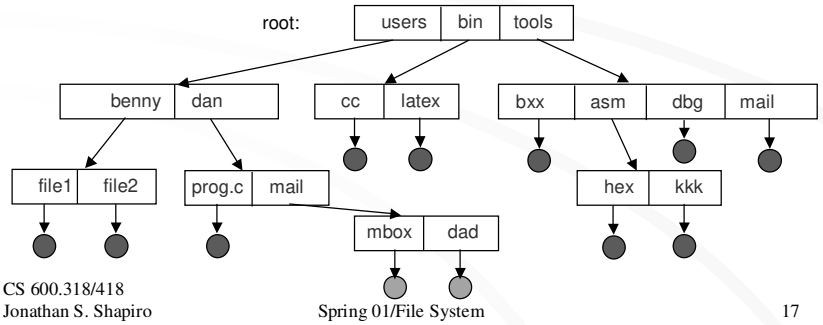
Two-Level Directory

- One master file directory.
- Each user has their own user file directory.
- Each entry in the master file directory points to a user file directory.
- Issues:
  - **Sharing - accessing other users files.**
  - **System files.**
  - **Grouping problem.**



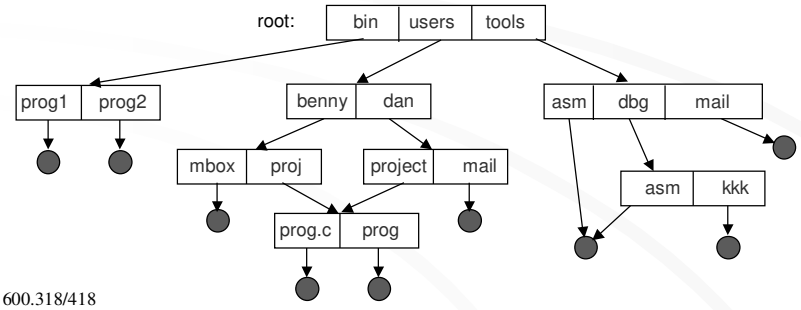
Tree-Structured Directory

- The directory structure is a tree with arbitrary height.
- Users may create their own subdirectories.
- Issues
  - **Efficient Searching, grouping.**
  - **Current directory notion / change directory (absolute/relative naming).**
  - **Directory semantics (e.g. deletion).**



Acyclic-Graph Directory

- Allow sharing of directories and files several times on the tree structure.
- Issues
  - **Only one actual copy of the file or directory is stored.**
  - **Can be accessed through one or more paths.**
  - **May have different names.**



Acyclic-Graph Directory (Cont.)

- There are several ways to implement shared files and directories.
- Example - Unix:
  - **Symbolic links:**
    - A different type of a directory entry (other than a file or a directory).
    - Specifies the name of the file that this link is pointing to.
    - Can be relative or absolute.
    - What happens when one deletes the original file?
  - **Duplicate directory entries (also called hard links):**
    - The original and copy entries are the same.
    - What happens when one deletes the original or copy entries?

General-Graph Directory

- A problem with acyclic graphs - how to ensure that there are no cycles.
  - **Can happen only when linking a directory.**
  - **Every time a link is added to a directory - use a cycle detection algorithm...**
- General graph directories allow cycles.
- Issues:
  - **How to avoid traversing a component in a cycle while searching.**
  - **How to release orphan cycles (garbage collection).**

Protection

- File owner/creator should be able to control:
  - **What file access operations are allowed.**
  - **By whom.**
- Types of access:
  - **Files:**
    - Read, Write, Execute, Append.
  - **Directories:**
    - Create, Delete, List, Search.

Access Control Methods

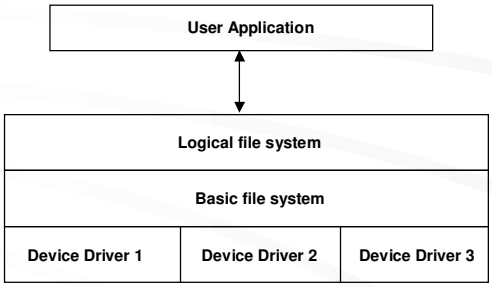
- Access lists:
  - **Associating an access list with each file.**
  - **Maintenance problem as system users are dynamic.**
  - **Where do we store this? (variable size for directory entries?).**
- Groups: (example - Unix)
  - **Condensing access lists to three classifications for users: owner, group, universe.**
  - **Maintenance of groups in the system is not connected with specific files or directories.**
  - **Each file has an owner and a group associated with it. Permissions are specified to each of them** (rwxrwxrwx)**.**
- Passwords:
  - **Associating a password with files or directories.**
  - **Maintenance problem for passwords in the system.**

Consistency Semantics for File Sharing

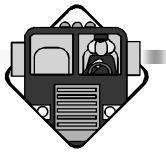
- Determines how modification of data is observable to other users.

- Unix Semantics:
  - **Writes to a file by a user are immediately visible to other users that have this file open.**
  - **Supports a mode of sharing where users share even the current location pointer into the file.**
- Session Semantics (Andrew File System):
  - **Writes to a file by a user is not visible to other users.**
  - **Once the file is closed, the changes are visible only to new sessions.**

File System Architecture



File system components



- Device Drivers
  - **Communicates directly with peripheral devices (disks, tapes, etc.).**
  - **Responsible for starting physical I/O operations on the device.**
  - **Processes the completion of an I/O request.**
  - **Schedule access to the device in order to optimize performance.**

File system components

- Basic File System
  - **Uses the specific device driver.**
  - **Deals with blocks of data that are exchanged with the physical device.**
  - **Concerned with the placement of blocks on the disk.**
  - **Concerned with buffering blocks in main memory.**
- Logical File System
  - **Responsible for providing the previously discussed interface to the user including:**
    - File access.
    - Directory operations.
    - Security and protection.

File Allocation Methods

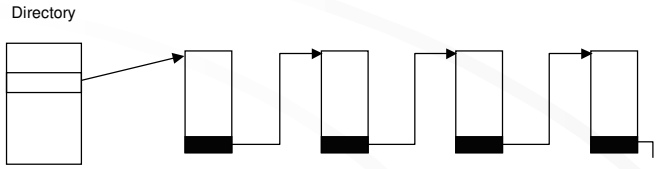
- Contiguous allocation.
- Linked allocation.
- Indexed allocation.
- Multilevel indexed allocation.
- Combined scheme.

Contiguous File Allocation

- Each file occupies a set of contiguous blocks on the disk.
- Allocation using first fit / best fit.
- A Need for compaction.
- Only starting block and length of file in blocks are needed to work with the file.
- Allows random access.
- Problems with files that grow.

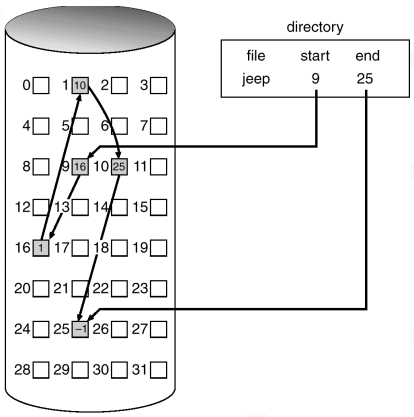
Linked File Allocation

- Each file is a linked list of blocks.
- Tradeoffs?
- No external fragmentation.
- Effective for sequential access.
- Problematic for direct access.



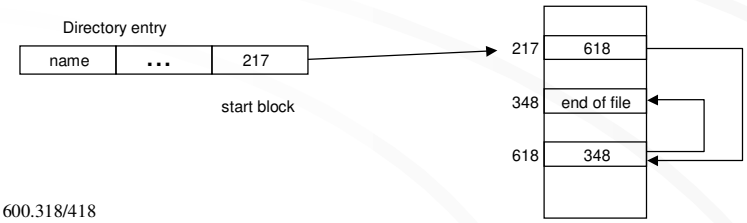
Linked File Allocation (cont).

- How it looks on the disk:



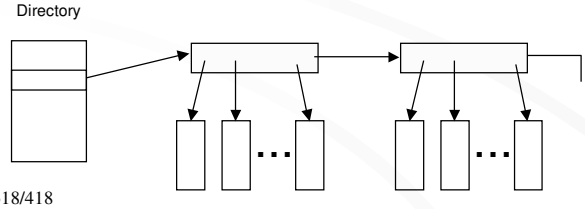
File Allocation Table (FAT).

- Variation of the link list (MS/DOS and OS/2).
  - **A section of the disk at the beginning of each partition (Volume) is set aside to contain a FAT.**
  - **FAT has one entry for each disk block, pointing to the next block in the file.**
  - **The link list is implemented in that section.**
  - **Actual file blocks contain no links.**



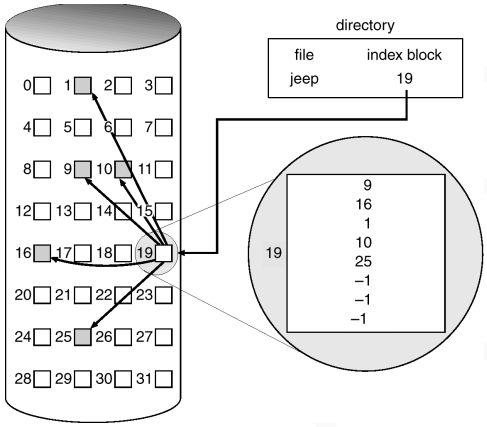
Indexed File Allocation

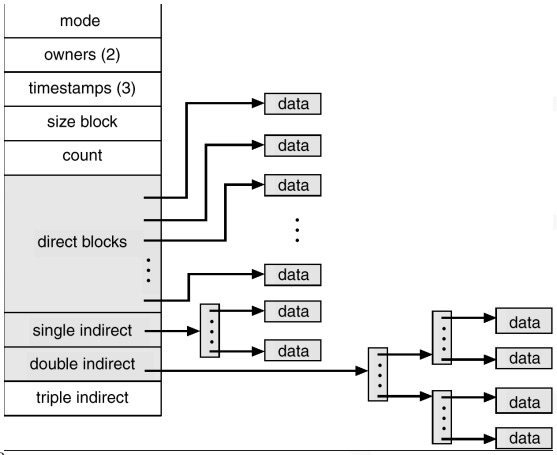
- Indexed allocation is bringing all the pointers together.
- Tradeoffs?
- Much more effective for direct access.
- Inefficient for small files
  - **both access and space**.



Indexed File Allocation (cont).

- How it looks on the disk:



A Combined Scheme

- Unix