# Announcements

- Office hours Sunday at home 5p-7p (829 5639).
- Assignment 4 is online, officially handed out.
- Assignment 2 answer key is online.

# Chapter 11: File-System Interface

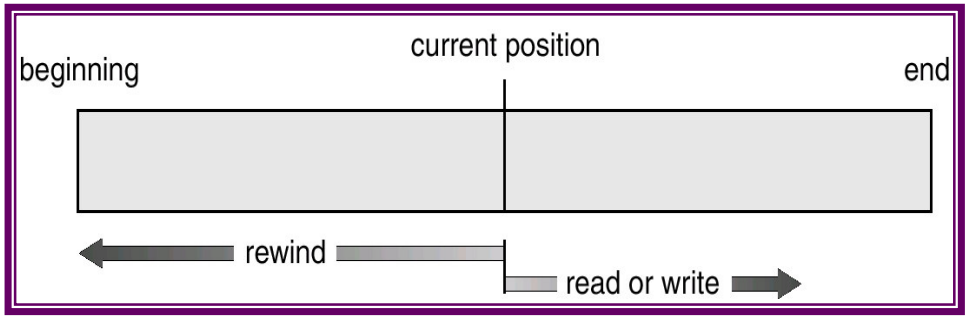
- File Concept.
- Access Methods.
- Directory Structure.
- File System Mounting.

# File Concept

- Types:
  - Data: binary, numeric, character (Unicode tags: e.g. FF FE; see Notepad).
  - Program (executable).
- Structure:
  - None: sequence of bytes.
  - Records:
    - Lines.
    - Fixed/variable length.
    - Formatted document: XML.
    - Relocatable load file (executable): ELF.
- Can implement structure via unstructured file by inserting control characters (e.g. "`\n`" "`\r\n`" mean end-of-line).
- Who offers the abstraction of structure?
  - Operating system: IBM, older mainframes to optimize storage/access (e.g. one disk block per record).
  - Applications: most modern systems to enable extensibility.

# Access Methods

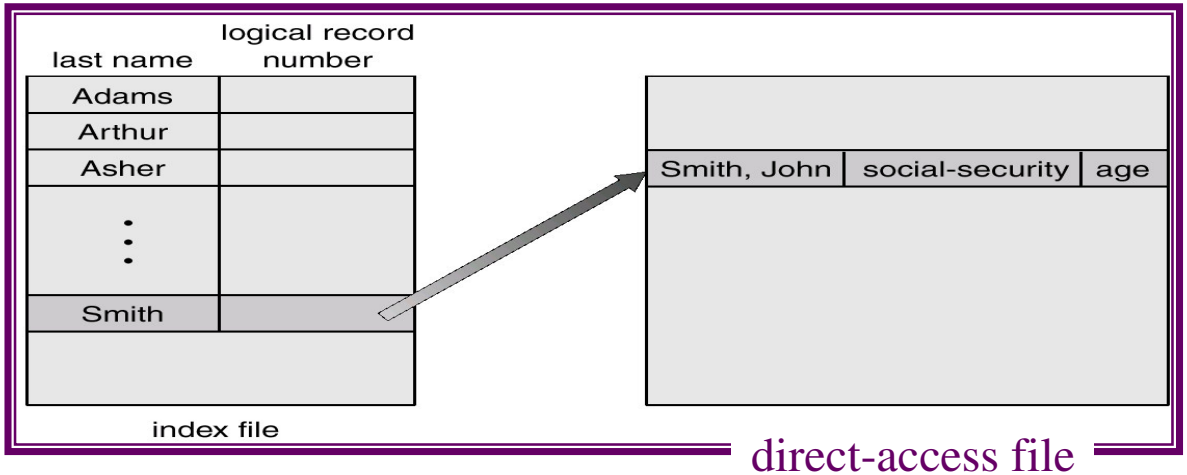
1. *Sequential* access: current position marks location of next read/write. Advances automatically after each operation. Cannot set to arbitary value.




2. *Direct* (or *relative*) access: current position can be set to arbitary value.

| sequential access | implementation for direct access |
|---|---|
| *rewind* | *cp* = 0; |
| *read next* | *read cp*;<br>*cp* = *cp*+1; |
| *write next* | *write cp*;<br>*cp* = *cp*+1; |

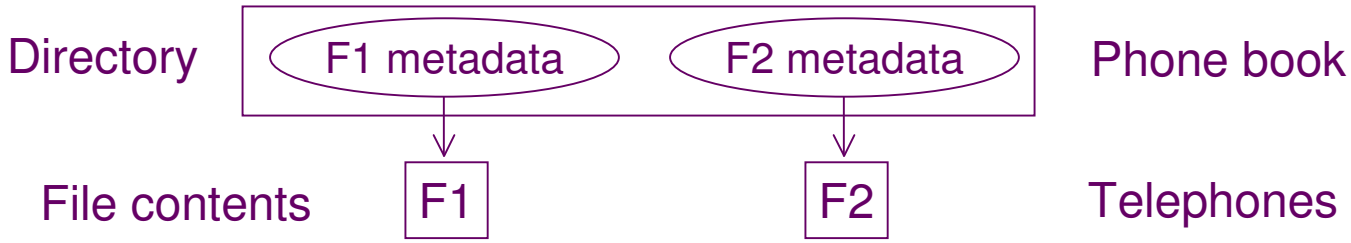
3. Other access: e.g. index + direct-access. Fast lookup via binary search on last name. OS manages these two physical files as a *single* abstract file.



General term; different than Windows folder. →

# Directory Structure

- Directory contains all information (meta-data) about files except for the file contents. Also stored on disk.
- Meta-data comprises:
  - Name: in which character set? See next slide.
  - Type: see earlier slide.
  - Location: pointer to file data on device (blocks).
  - Size: in bytes and/or blocks (fragmentation, index block meta-data). See Windows properties dialog box.
  - Protection: controls who (user, user group) can do what (read, write, execute).
  - Timestamps (access, modification, creation) and other data for protection, security, and usage monitoring.



# Unicode File Names

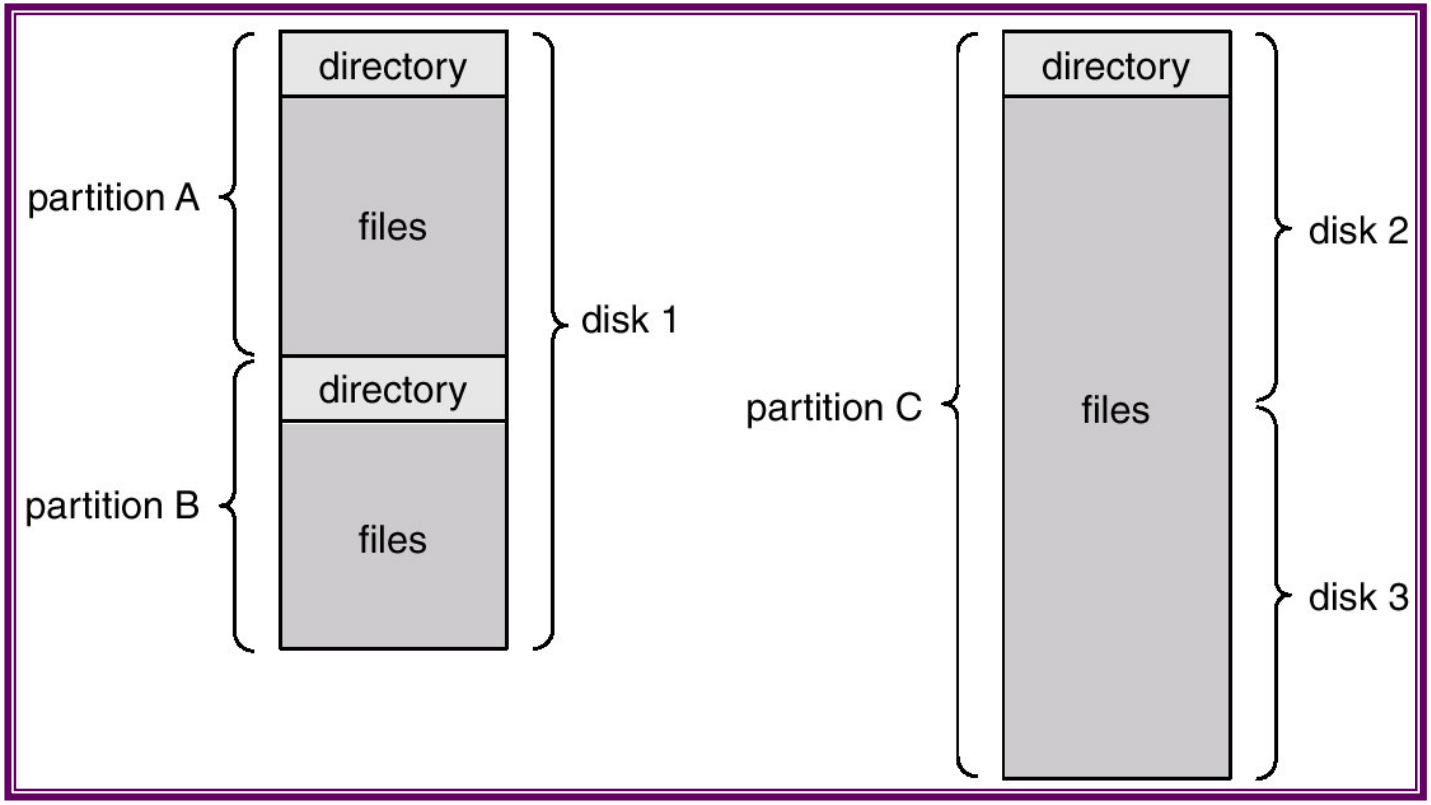
- Map from characters of all languages to integers.
- Unicode Transformation Format (UTF): how to store those integers into bytes:
  - UTF-16: use 2 bytes per integer. Easy.
  - UTF-8:
    - If integer *i*<128 (ASCII character), store one byte.
    - If *i*>=128, then store as two or more (up to six) bytes.
    - Very compact for ASCII strings, backwards-compatible.
- How are file names stored?
  - Windows NTFS: UTF-16.
  - Unix: doesn't care how to interpret file name: it is just a sequence of null-terminated non-zero bytes (`char *`).
    - So, for Unicode, UTF-8 because UTF-16 contains 0 bytes.
    - Any character set is fine by Unix. To translate to characters, applications (e.g. `ls`) use `LANG` environment variable.

# Disks/Partitions/Directories

Always: one directory per partition (a.k.a. logical disk, file system).
Typical: one partition per physical disk.

Two logical disks per physical disk:
e.g. isolates program from data storage
to protect programs from spillover in data.

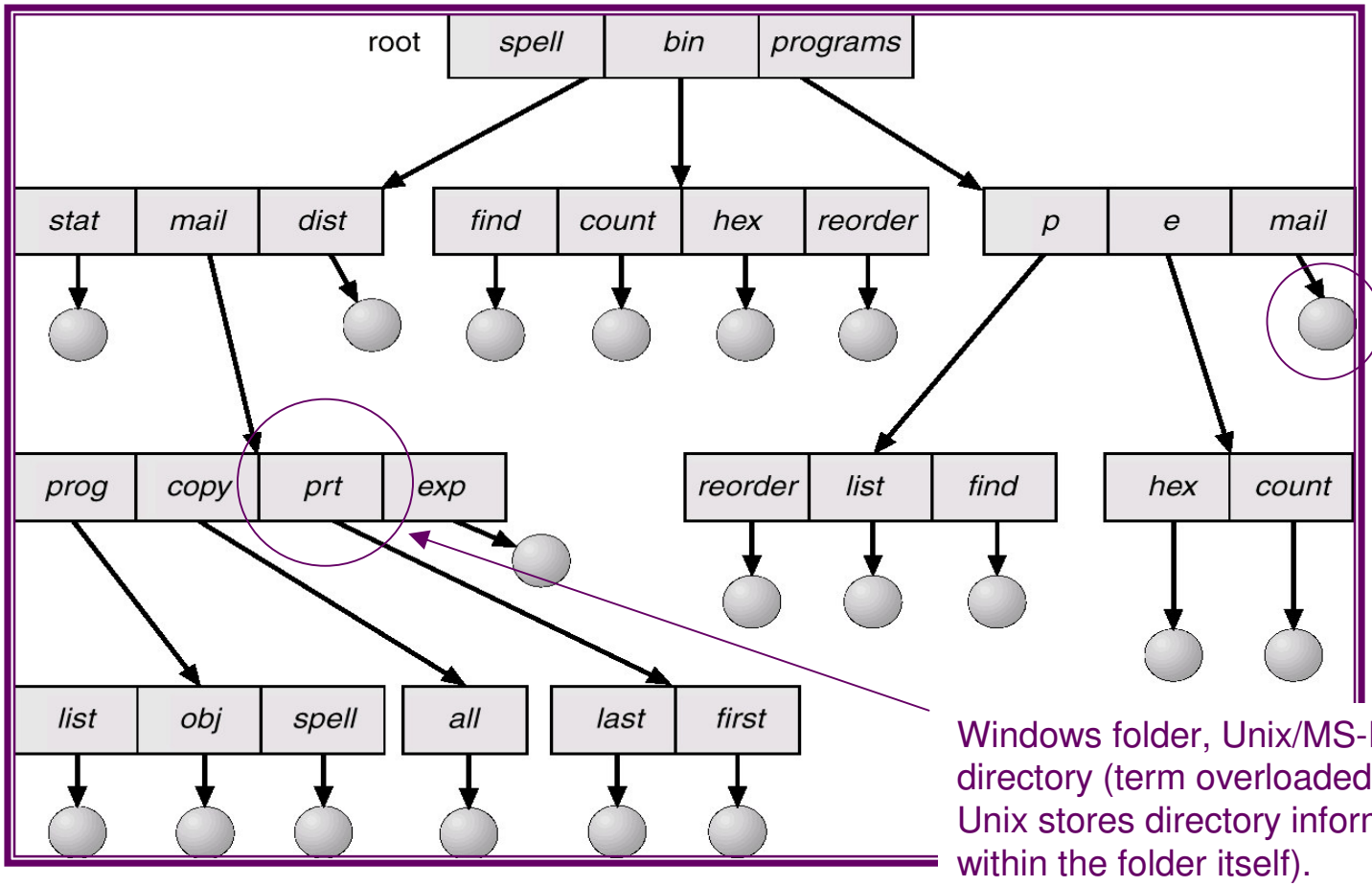
One logical disk combining two
physical disks: offers abstration of
single larger logical space.



# Tree-Structured Directories

Logical organization of directory into hierarchical groups for:
- Convenience: different users, same file names.
- Quick access: not too many files in one place.
- Grouping: logically group files by properties, e.g. all JDK 1.4 files.

**MS-DOS**
**MP3 Plrs**

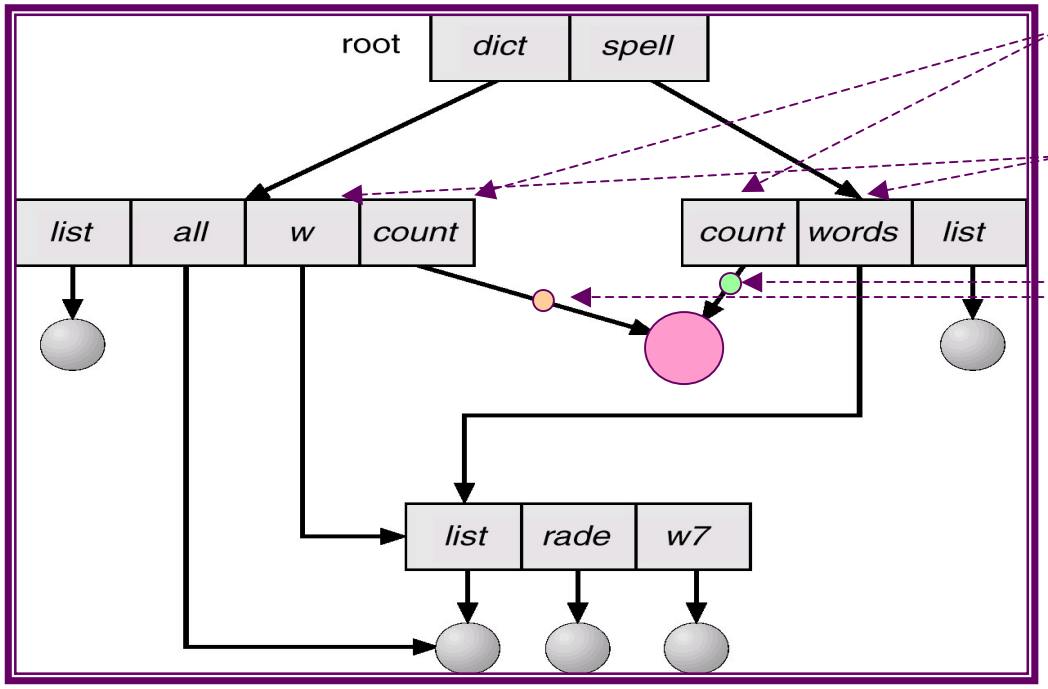



File contents.

Windows folder, Unix/MS-DOS directory (term overloaded because Unix stores directory information within the folder itself).

# Tree-Structured Directories

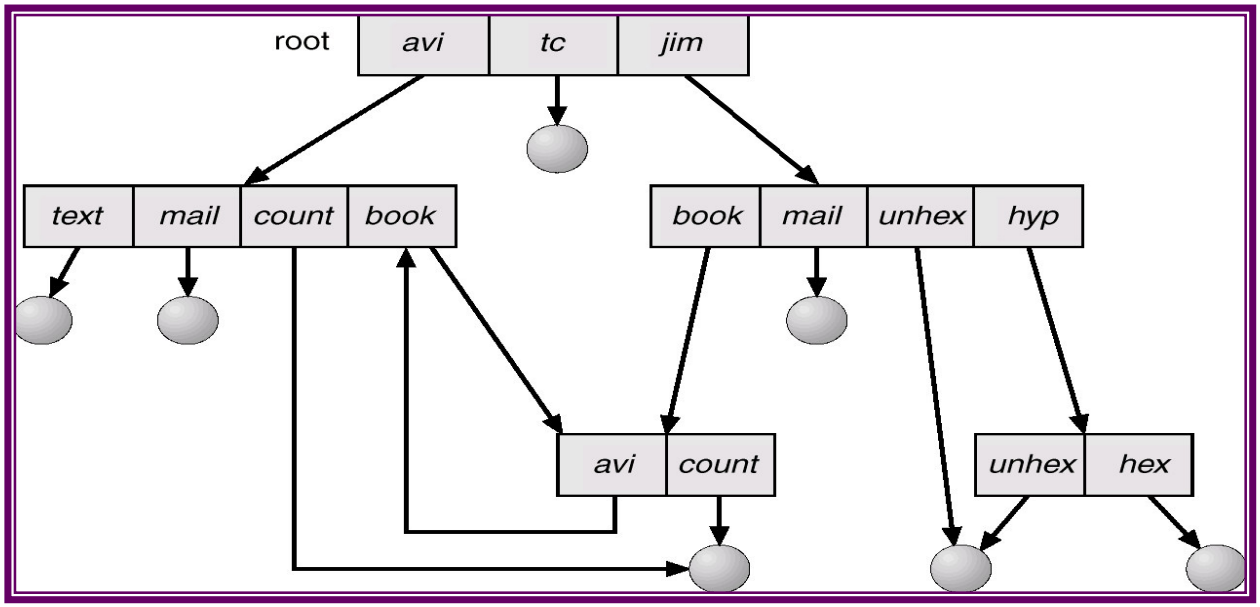
- *Absolute* file *path* or *relative* to *current* (working) *folder*. One current folder per process (in PCB).
- When folder is deleted, what happens to its contents?
  - Delete them too (Unix: `rm -r` command; *no* such system call).
  - Refuse to delete non-empty directory (Unix: `rmdir()` sys call).
- Adding shared subfolders and files gives acyclic graph:




Same name *count*, but can be different: same file contents or folder, different name (*alias*) as with *words*, *w*.

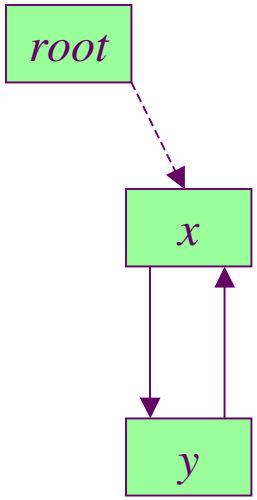
What happens if ○ goes away (i.e. an alias is deleted)?
• Count aliases, delete ● when none left (Java-like reference counting).
• Delete ● immediately. Use linked-list of backpoints to find and delete ○ . Or leave it *dangling* (like pointer to freed memory).

# General Graph Directory




**UNIX Windows**

- How do we guarantee no cycles?
  - Allow only aliases for (*links to*) files, not folders.
  - Every time user attempts to create a new link, run cycle detection and disallow addition if it would create cycle.
  - Let cycles occur:
    - When asked to perform recursive operation, complain if we go too deep: `/x/y/x/y/x/y`...
    - Need garbage collection to remove unreachable files: after running `rm /x`, `x` and `y` are both unreachable, but they still have one alias each (`x` has `y`, `y` has `x`) so they are not deleted.



# Unix Aliases

- Hard links (`ln` command):
  - Hard link must point to a file: no cycles.
  - Affects reference count: file contents remain until no hard links to file.
  - Even for non-shared files, directory entry is a hard link.
- Soft/symbolic links (`ln -s` command) (a.k.a. symlinks):
  - Small file `l` whose contents are (absolute or relative) *name* of another file `f`.
  - When asked to operate on `l`, OS operates on `f` instead.
  - If `f` is deleted, `l` is left dangling (contains invalid name) so operations on `l` fail with "File does not exist." `l` does not affect reference count of `f`.
  - Soft links can be aliases for folders too...
    - ... but garbage collection not needed because symlinks don't affect reference count so never prevent deletion.
    - OS complains "Path too long" if cycle leads to 256 or more elements in file path while trying to *resolve* it.
  - Canonical file path: alternate path to file without symbolic links in the path. May have one or more due to hard links.

# File System Mounting

- If two file systems, does name `/x` refer to one or the other?
  - Create one global file namespace: `/f1/x` means `/x` under `f1` file system, `/f2/x` means `/x` under `f2` file system.
  - Generalize: *mount* file system anywhere in global namespace:
    - Create empty folder: `mkdir /users/f1users`.

      File system name: device/partition.
    - Mount: `mount` `dsk1/p1` `/users/f1users`.
- Unix: put non-file entities in same space, e.g. processes under `/proc`.