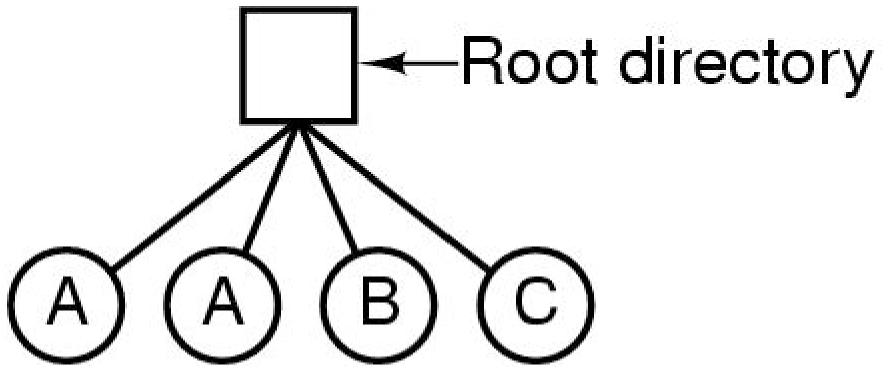
Memory-Mapped Files

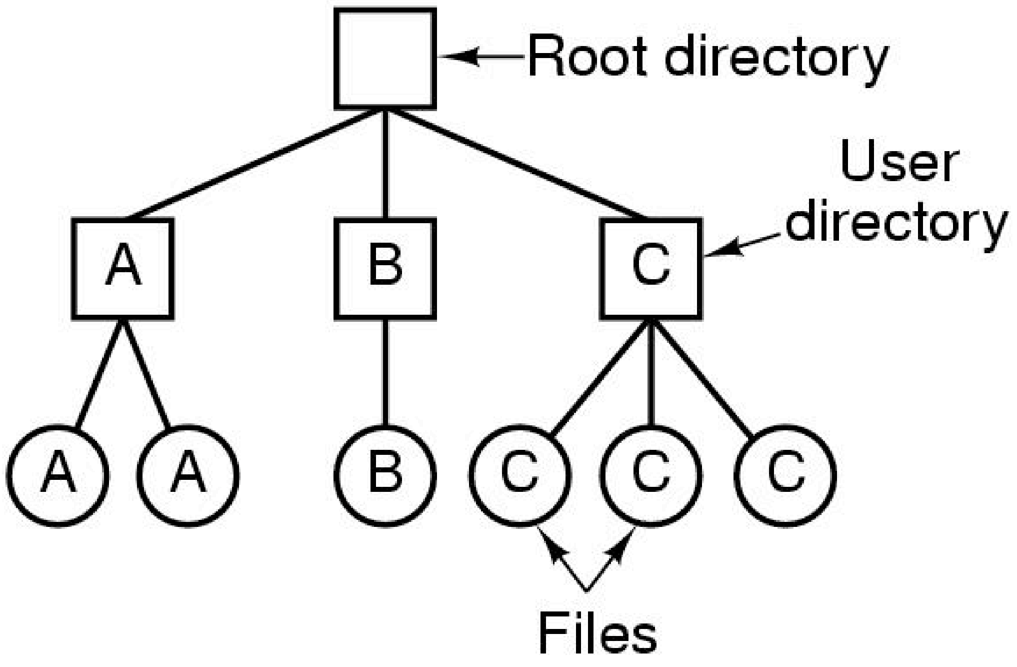
- Many operating systems provide a service whereby a whole file is mapped into memory
- A read/write operation to the mapped memory region is equivalent to read/write to the corresponding file.
- In fact, the loader uses this service to load programs and libraries into memory.

Directories
Single-Level Directory Systems



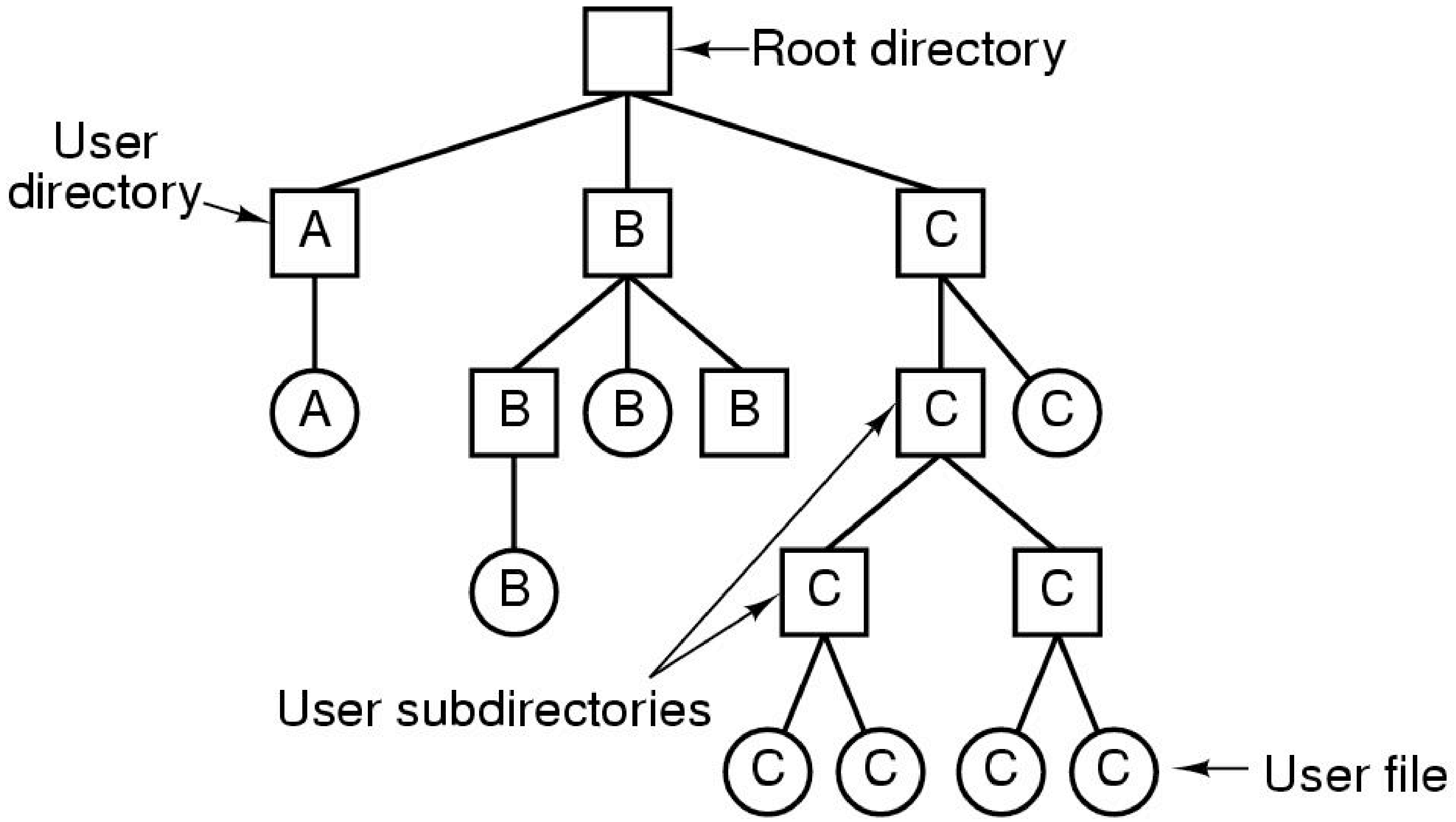
- A single level directory system
  - contains 4 files
  - owned by 3 different people, A, B, and C

Two-level Directory Systems



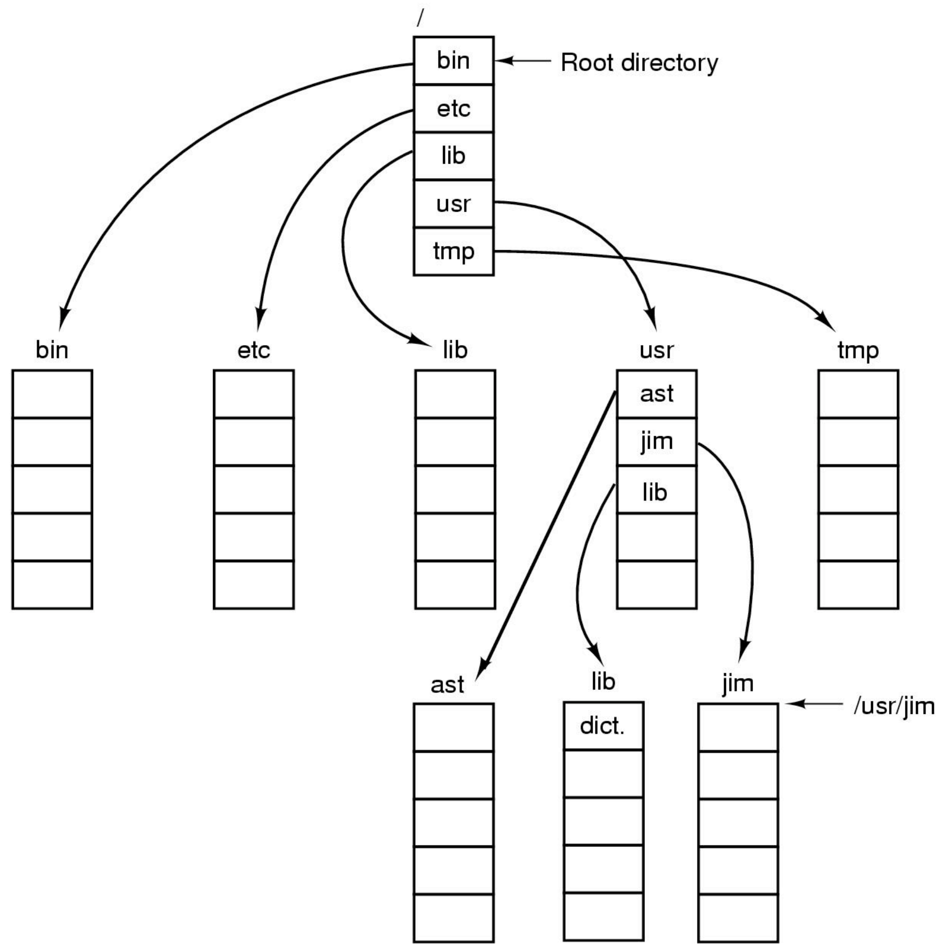
Letters indicate *owners* of the directories and files

Hierarchical Directory Systems



A hierarchical directory system

Path Names

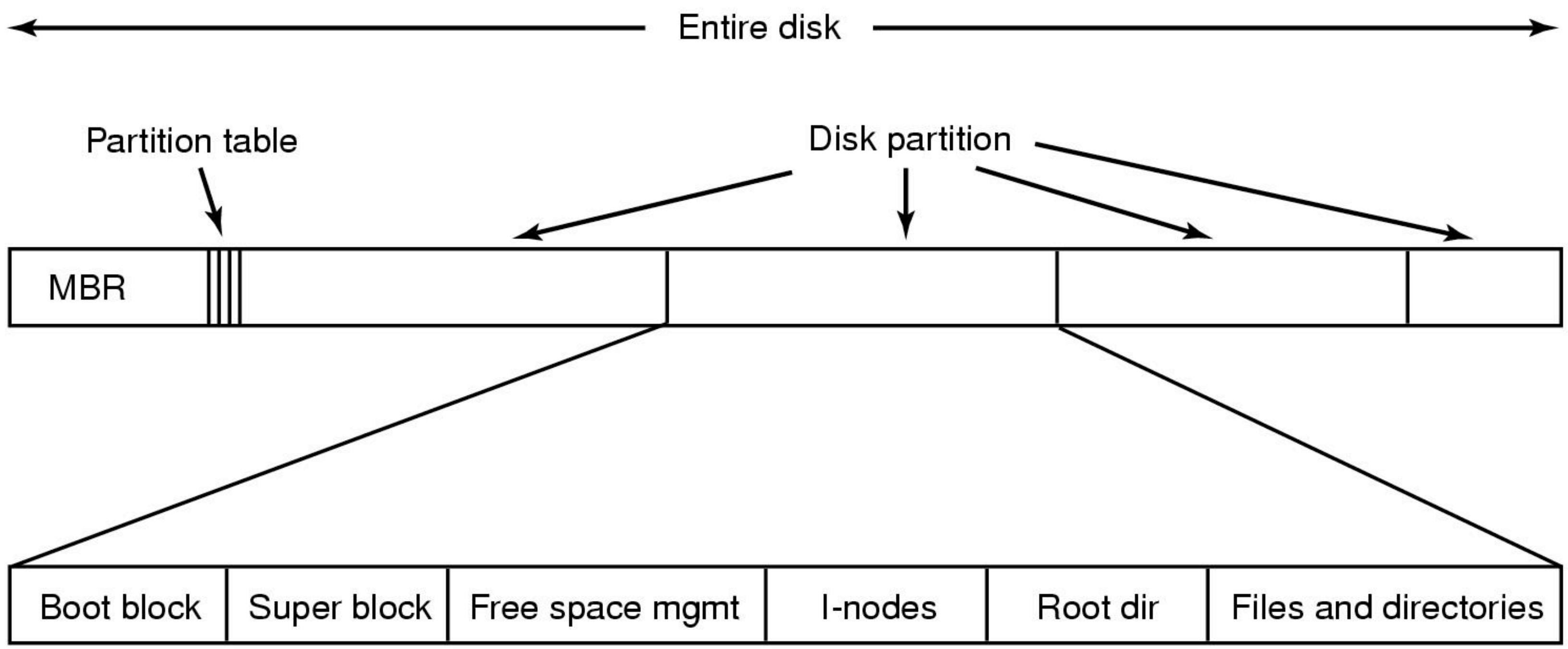


A UNIX directory tree

Directory Operations

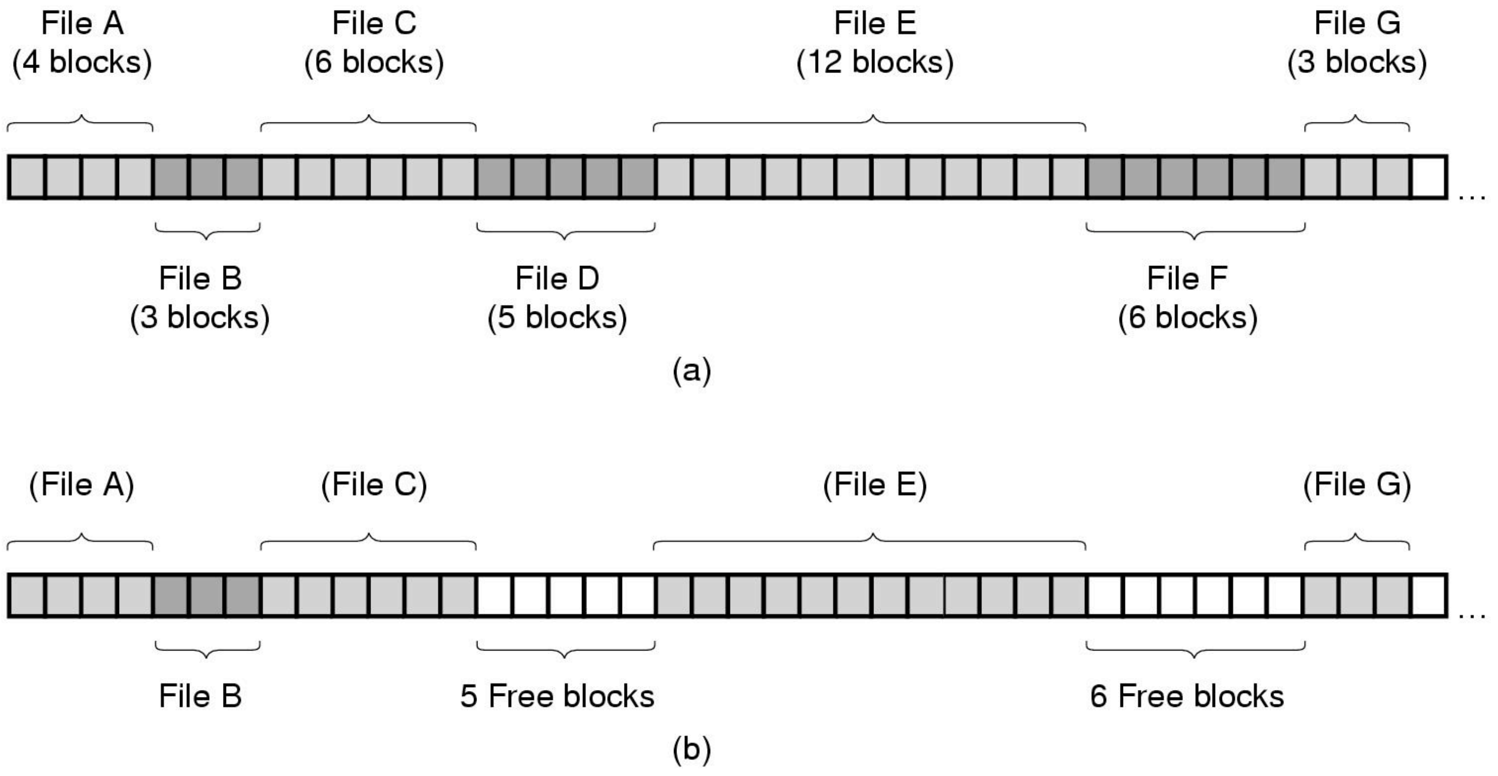
1. Create
2. Delete
3. Opendir
4. Closedir
5. Readdir
6. Rename
7. Link
8. Unlink

File System Implementation



A possible file system layout

Implementing Files
Contiguous Allocation



(a) Contiguous allocation of disk space for 7 files

(b) State of the disk after files *D* and *E* have been removed

Contiguous Allocation Advantages

- Simple to implement.
  - For every file we need two numbers:
    - Address of the first block
    - Number of blocks
- High read performance
  - Since blocks are contiguous, the whole file can be read in a single operation

Contiguous Allocation Disadvantages

- High disk fragmentation
  - When files are deleted they leave holes
- Need to specify the size of the file at creation time
  - Specify small size=>cannot add to the file later
  - Specify large size=>large unused space on disk