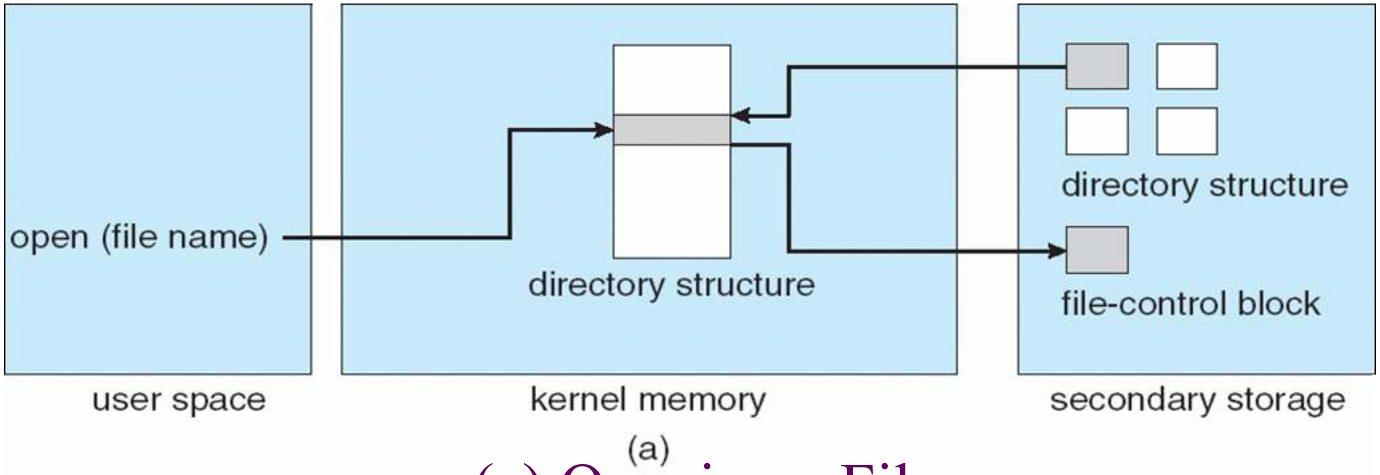
DATA STORAGE TECHNOLOGIES & NETWORKS (CS C446, CS F446 & IS C446)

LECTURE 26– STORAGE

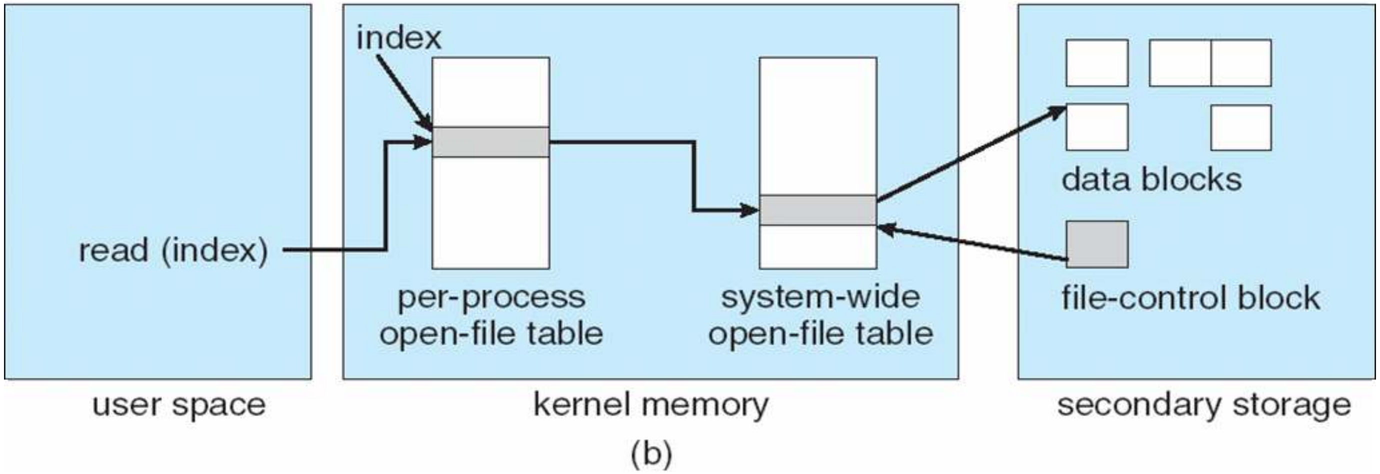
vnode

- Information stored in vnode
  - flags (used for locking vnode & identifying generic attributes)
  - Various reference counts
    - Number of file entries that are open for read and/or write that references vnode
    - Number of file entries that are open for writing that reference the vnode
    - Number of pages and buffers that are associated with the vnode
  - A pointer to the mount structure describes the file system that contains the object represents by the vnode
  - Various information is used to do file read ahead
  - A reference to an NFS lease
  - A reference to state about special devices, sockets, and FIFOs
  - Pointer to the set of vnode operations defined for the object
  - Pointer to private information needed for the underlying object (for local file system, this pointer will reference an inode and for NFS it is an nfsnode)
  - The type of the underlying object (regular file, character / block device, directory etc..)
  - Clean and dirty buffers associated with vnode
  - A count (number of buffer write operation in progress)

In-Memory File System Structures

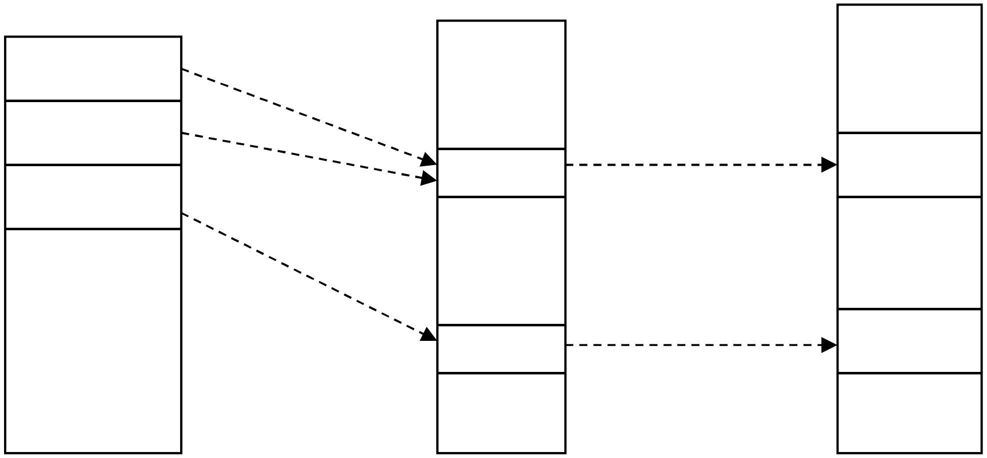


(a) Opening a File



(b)Reading a File

File descriptors, file table and inode table

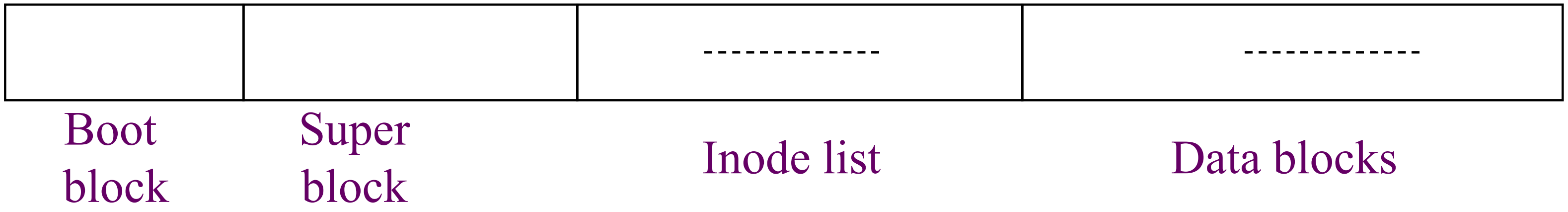


User
File Descriptor
Table
(per process)

File Table
(system wide)

Inode table
(system wide)

File system layout



Boot block

Super block

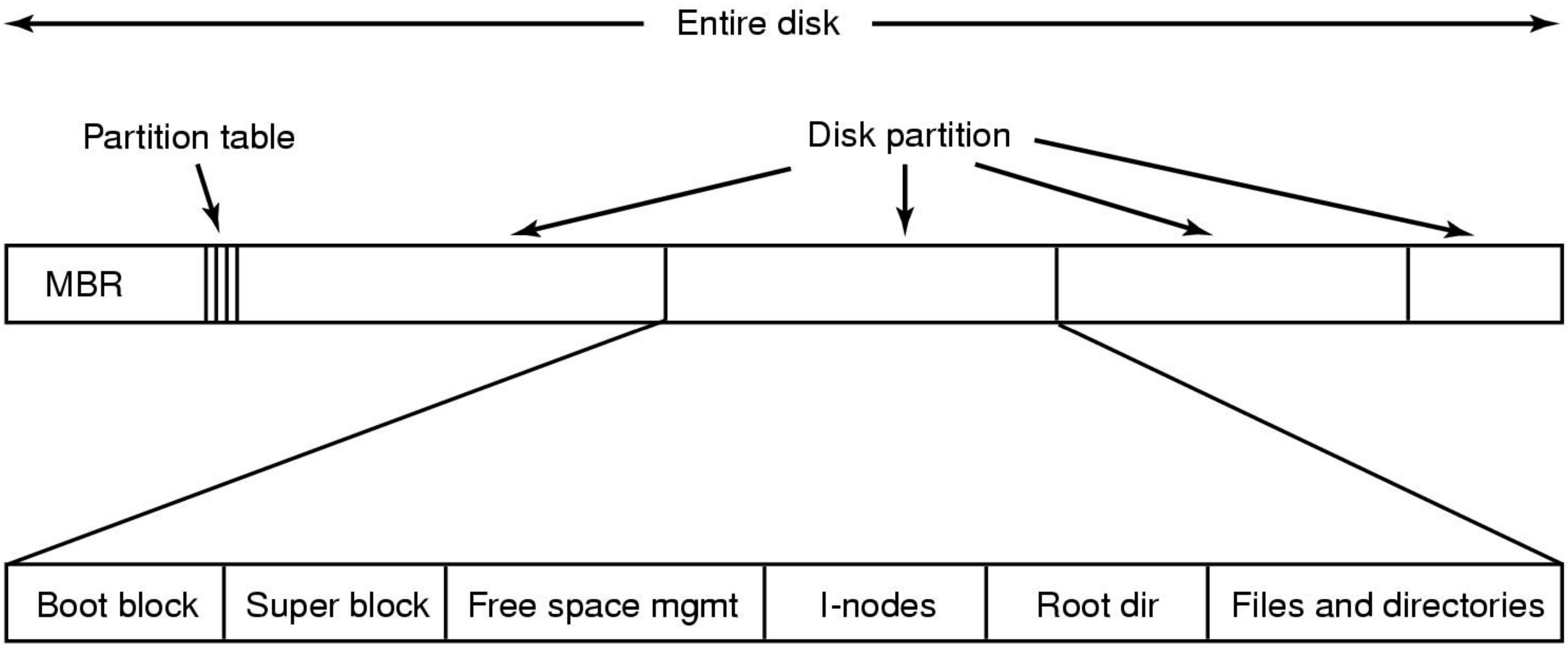
Inode list

Data blocks

- Boot block occupies the beginning of a file system
  - Typically first block and may contain boot strap code.
  - Only one boot block is needed to boot the system, but every file system has a (possibly empty) boot block.
- Super block describes the state of a file system.
  - How large it is
  - How many files it can store
  - Where to find free space on the file system and
  - Other information.

- Inode list is the list of inodes that follow the super block in the file system
  - Administrators specify the size of the inode list when configuring a file system.
  - Kernel reference inode by index into the inode list.
  - One inode is root inode (by which the directory structure of file system is accessible after execution of mount system call)
- Data blocks starts at the end of inode list and contain file data and administrative data
  - An allocated data block can belong to one and only one file in the file system.

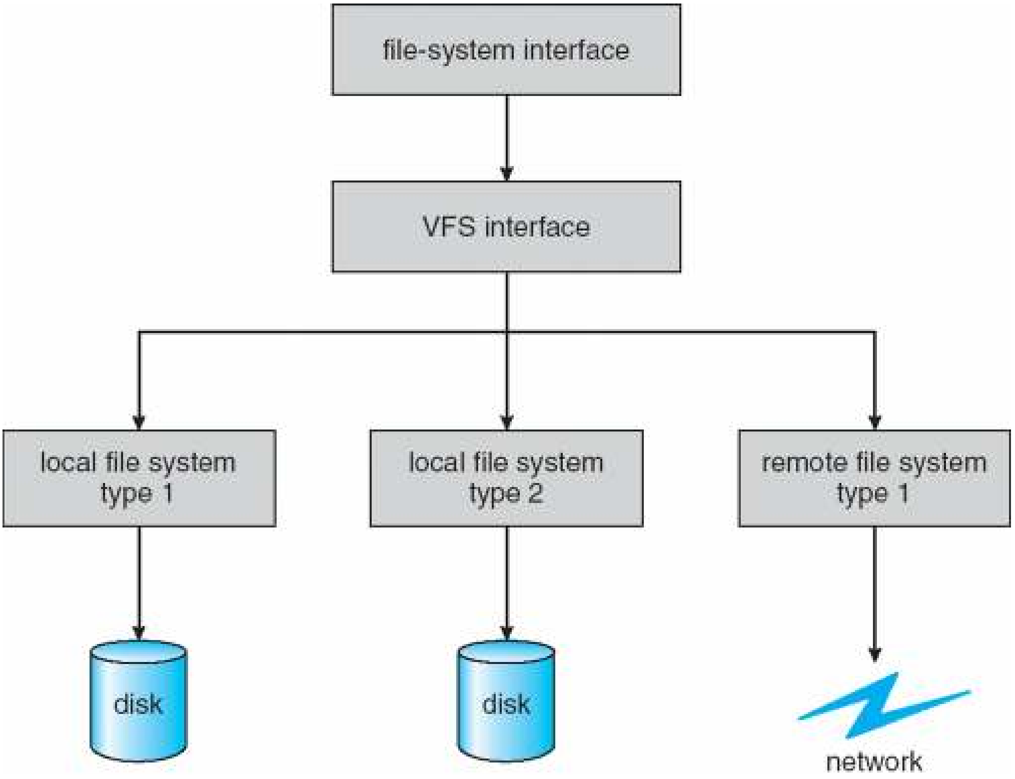
File System Implementation



A possible file system layout

Virtual File Systems

- Virtual File Systems (VFS) provide an object-oriented way of implementing file systems.
- VFS allows the same system call interface (the API) to be used for different types of file systems.
- The API is to the VFS interface, rather than any specific type of file system.



Buffering

- Buffer Cache
  - Memory buffer for data being transferred to and from disks
  - Cache for recently used blocks
    - 85% hit rate is typical
    - Typical buffer size = 64KB virtual memory
    - Buffer pool – hundreds of buffers
  - Consistency issue
    - Each disk block mapped to at most one buffer
    - Buffers have dirty bits associated
    - When a new buffer is allocated and if its disk blocks overlap with that of an existing buffer, then the old buffer must be purged.

Buffer Pool Management

- Buffer pool is maintained as a (separately chained) hash table indexed by a buffer id
- The buffers in the pool are also in one of four lists:
  - Locked list:
    - buffers that are currently used for I/O and therefore locked and cannot be released until operation is complete
  - LRU list:
    - A queue of buffers – a recently used item is added at the rear of the queue and when a buffer is needed one at the front of the queue is replaced.
    - Buffers staying in this queue long enough are migrated to an Aged list
  - Aged List:
    - Maintained as a list and any element may be used for replacement.
  - Empty List
- When a new buffer is needed check in the following order:
  - Empty List, Aged List, LRU list

The buffer cache

- Data access from a file
  - Process request to the kernel
  - Kernel brings data and auxiliary information (super block, inode etc)it into the memory
  - Process examines it
  - Process can alter / read / store it into sec memory back
- Kernel uses internal data buffers
  - Reduces frequency of disk access
  - System response time and throughput will be better
  - Contains the data in recently used disk blocks.
  - It is called buffer cache (a software structure, not hardware).
  - Places between file subsystem block and device driver block
  - Improves the write performance by accumulating multiple writes in the same block before writing it into disk.