ECE437/CS481

M06A: FILE SYSTEMS

CHAPTER 11.1-11.3 & 12.1-12.3

Xiang Sun

The University of New Mexico

File Concept

- Contiguous logical address space

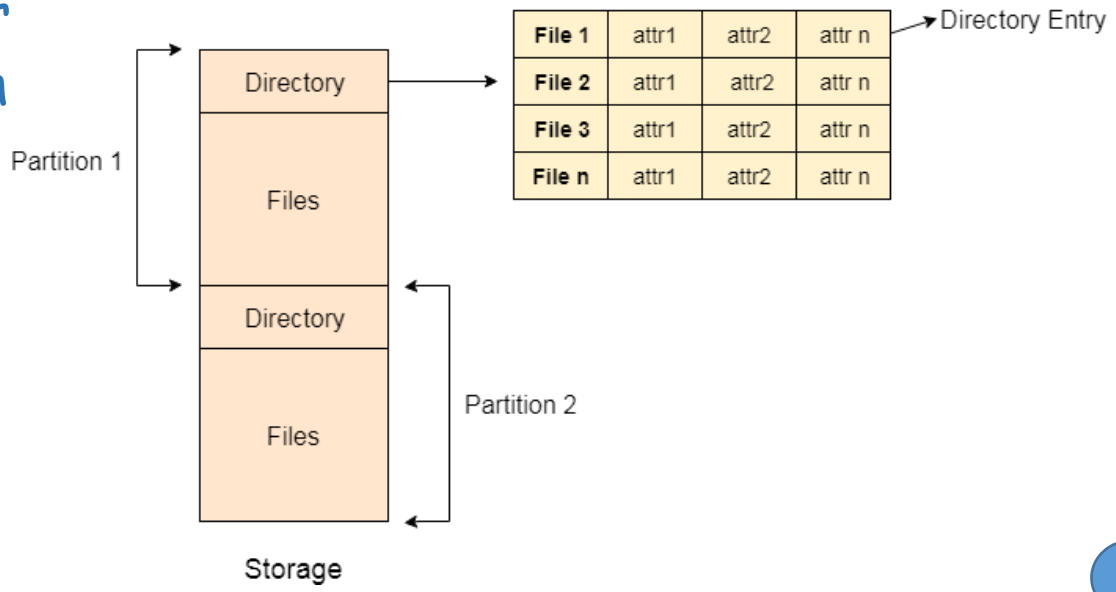
- Types:
  - Data
    - ✓ Numeric
    - ✓ Character
    - ✓ Binary
  - Program
    - ✓ source programs
    - ✓ executable programs

File Structure

- A file can have various kinds of structure, which depends on its type
  - A text file is a sequence of characters organized into lines.
  - A source file is a sequence of functions, each of which is further organized as declarations followed by executable statements.
  - An executable file is a series of code sections that the loader can bring into memory and execute.
  - Who interprets this structure?
    - ✓ Operating system
    - ✓ Program

File Attributes

- **Name** – only information kept in human-readable form
- **Identifier** – unique tag (number) identifies file within file system
- **Type** – needed for systems that support different types
- **Location** – pointer to file location on device
- **Size** – current file size
- **Protection** – controls who can do reading, writing, executing
- **Time, date, and user identification** – data for protection, security, and usage monitoring
- **Extended file attributes**: including character encoding of the file and security features such as a file checksum
- Information about files are kept in the directory table, which is maintained on the disk (nonvolatile device)
  - ✓ A directory table is a special type of file that represents a directory



File Operations

- Six Basic File Operations
  - Create a file
  - Write into a file: write()
  - Read from a file: read()
  - Deleting a file
  - Repositioning within a file
  - Truncating a file.
    - ✓ The user may want to erase the contents of a file but keep its attributes.
- Other File Operations
  - Open(Fi): search the directory structure on disk for entry Fi, and move the content of entry to memory (open-file table).
  - Close (Fi) – move the content of entry Fi in memory (open-file table) back to directory structure on disk
  - Copy a file---can be achieved by combining basic file operations.

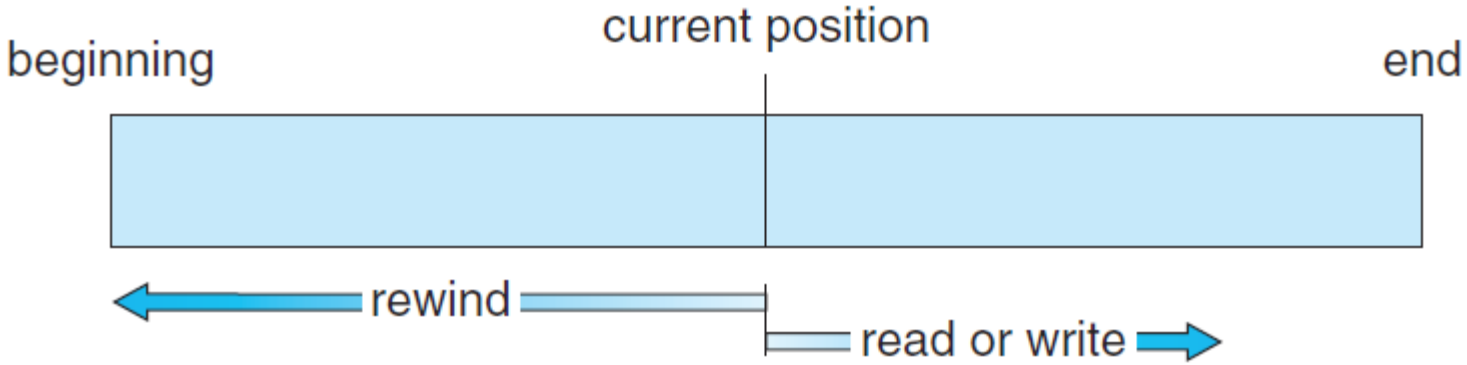
Opening File Information

- Information associating an opening file
  - File pointer: pointer to last read/write location. The file pointer is unique to each process operating on the file.
  - File-open count: multiple processes may have opened a file. File-open count tracks of the number of opens and closes of the file. If File-open count=0 (i.e., all the processes have closed the file), the file entry will be removed from the open-file table.
  - Disk location of the file
  - Access rights: per-process access mode information
  - File locks: locking a open lock to prevent other processes from gaining access to it.
    - Mandatory locks: if a process acquires an exclusive lock on a file, the OS will prevent other processes from accessing the file.
    - Advisory locks: if a process acquires an lock on a file, the OS will NOT stop other processes from accessing it. Instead, other process may check the status of the lock (hold/release) to determine whether to access it.

File Access Methods

File Access

- Define the ways to read/write data from/to a file.
- Two common access methods: Sequentially Access and Direct Access
- Sequentially Access
  - ✓ Reading/writing of data records/portions in sequential order, that is, one record after the other.
  - ✓ Reset to the beginning.



File Access Methods

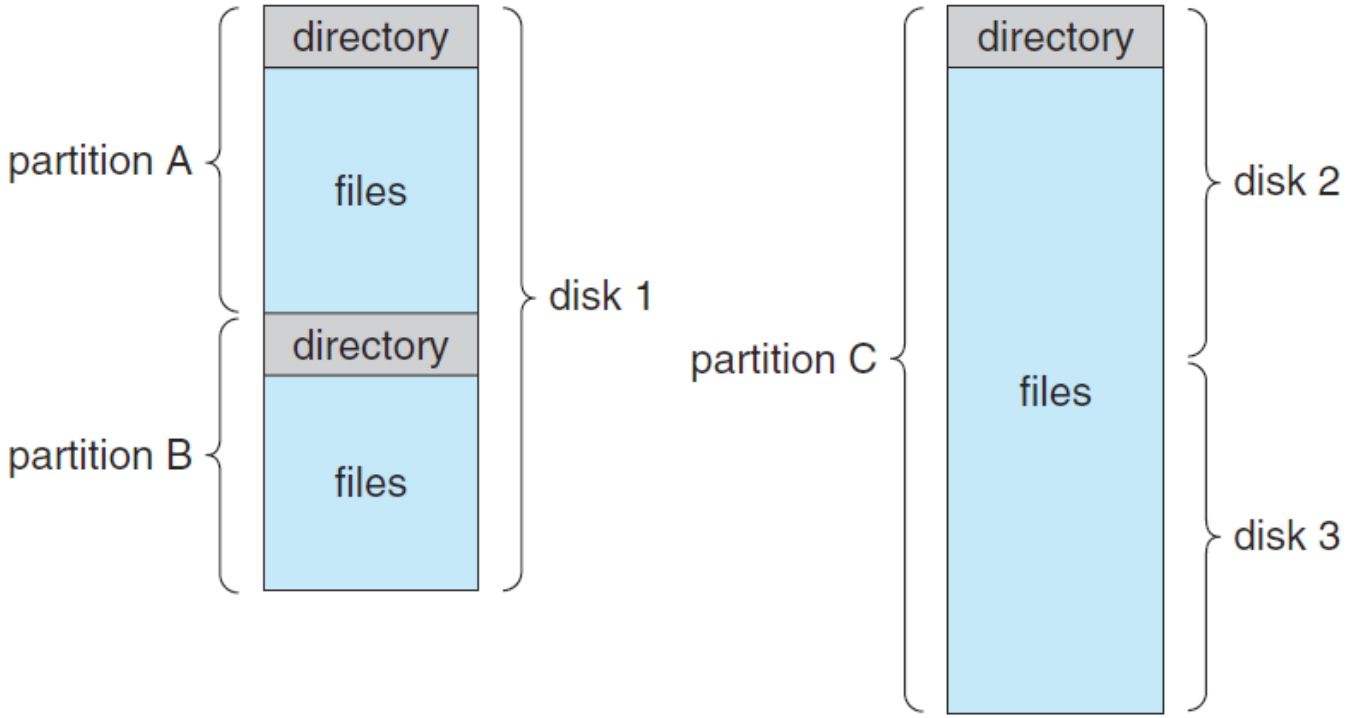
File Access

- Define the ways to read/write data from/to a file.
- Two common access methods: Sequentially Access and Direct Access
- Direct Access
  - ✓ Reading/writing of data records/portions in no particular order.
  - ✓ For example, the file is viewed as a numbered sequence of blocks or records. The process may read block 14, then read block 53, and then write block 7. There are no restrictions on the order of reading/writing for a direct-access file.
  - ✓ Very useful in searching and retrieving data records over a big database

Directory Structure

Directory

- Directory is a formatted record, containing various attributes of files/subdirectories.

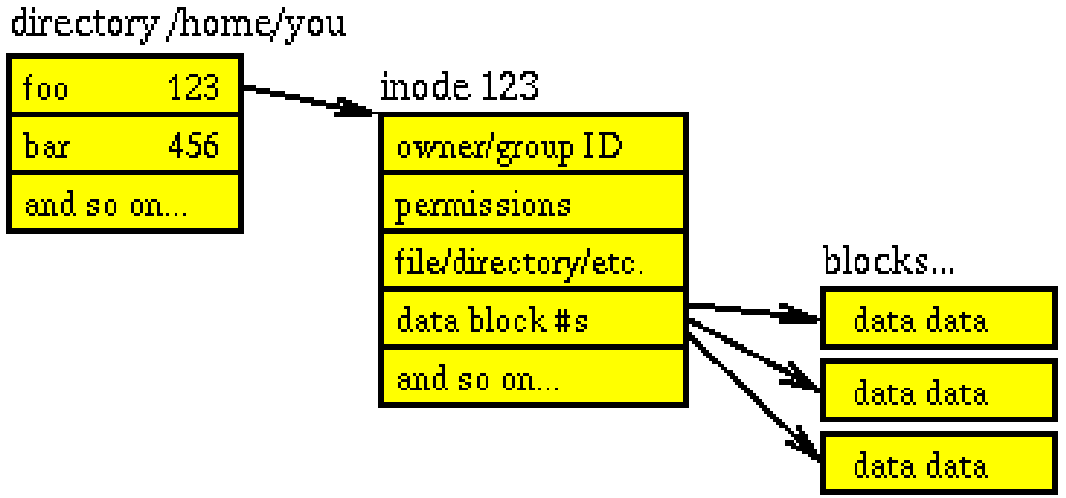


A typical file-system organization

Directory Structure

Directory in Unix

- In Unix, directory is different, i.e., the directory table only contains filenames of the files and the mapping to the related inode.
- inode: is a data structure that describes a file-system object such as a file or a directory. Each inode stores the attributes and disk block location of the object. Each inode is identified by an integer number (i.e., i-number), which is unique in a disk partition.



Directory Structure

Directory in Unix

- "." and ".." points to the current and parent directory
- inode number=2 ➔ root directory
  - ✓ why inode numbers of /run and /dev also equal 2?
  - ✓ /run and /dev are under different devices.

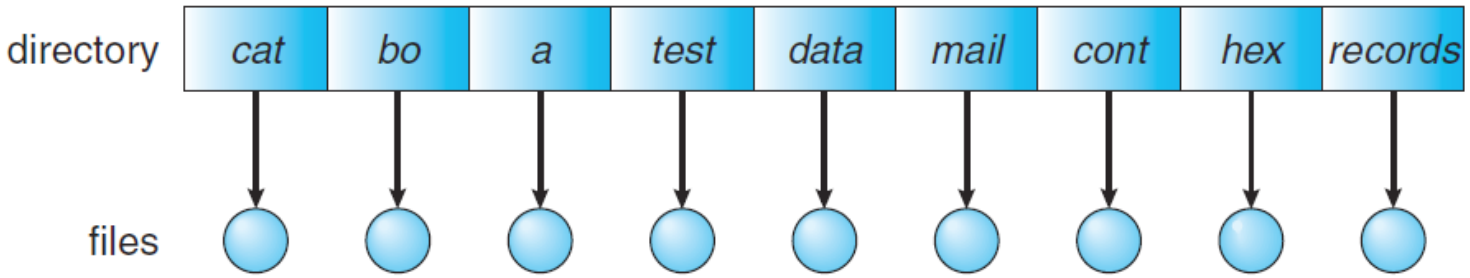
```
shaun@shaun-VirtualBox:/home$ cd /
shaun@shaun-VirtualBox:/$ ls -ila
total 2097264
      2 drwxr-xr-x  24 root root       4096 Oct  3 06:16 .
      2 drwxr-xr-x  24 root root       4096 Oct  3 06:16 ..
1310721 drwxr-xr-x   2 root root       4096 Sep 28 17:25 bin
2228225 drwxr-xr-x   3 root root       4096 Oct  4 06:41 boot
2883586 drwxrwxr-x   2 root root       4096 Aug 31 16:05 cdrom
      2 drwxr-xr-x  18 root root       4100 Oct 17 14:40 dev
 262145 drwxr-xr-x 122 root root      12288 Oct 18 06:04 etc
2359297 drwxr-xr-x   3 root root       4096 Aug 31 16:06 home
     14 lrwxrwxrwx   1 root root         33 Oct  3 06:16 initrd.img -> boot/initrd.img-4.15.0-36-generic
     13 lrwxrwxrwx   1 root root         33 Oct  3 06:16 initrd.img.old -> boot/initrd.img-4.15.0-34-generic
 655361 drwxr-xr-x  21 root root       4096 Aug 31 16:07 lib
1966081 drwxr-xr-x   2 root root       4096 Jul 24 21:04 lib64
     11 drwx------   2 root root      16384 Aug 31 16:03 lost+found
1441793 drwxr-xr-x   3 root root       4096 Sep  3 13:05 media
2883585 drwxr-xr-x   2 root root       4096 Jul 24 21:03 mnt
 393217 drwxr-xr-x   3 root root       4096 Oct  8 17:13 opt
      1 dr-xr-xr-x 224 root root          0 Oct 16 16:30 proc
3145729 drwx------   3 root root       4096 Sep 12 11:13 root
      2 drwxr-xr-x  26 root root        840 Oct 19 11:11 run
1179649 drwxr-xr-x   2 root root      12288 Oct  8 17:14 sbin
2752513 drwxr-xr-x  10 root root       4096 Aug 31 16:10 snap
 786433 drwxr-xr-x   2 root root       4096 Jul 24 21:03 srv
     12 -rw-------   1 root root 2147483648 Aug 31 16:04 swapfile
      1 dr-xr-xr-x  13 root root          0 Oct 19 16:43 sys
2490369 drwxrwxrwt  15 root root       4096 Oct 19 11:11 tmp
 917505 drwxr-xr-x  10 root root       4096 Jul 24 21:03 usr
 524289 drwxr-xr-x  14 root root       4096 Jul 24 21:14 var
     16 lrwxrwxrwx   1 root root         30 Oct  3 06:16 vmlinuz -> boot/vmlinuz-4.15.0-36-generic
     15 lrwxrwxrwx   1 root root         30 Oct  3 06:16 vmlinuz.old -> boot/vmlinuz-4.15.0-34-generic
shaun@shaun-VirtualBox:/$
```

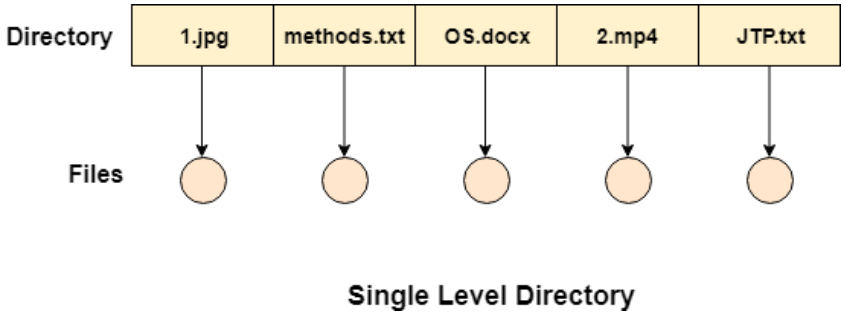
```
shaun@shaun-VirtualBox:/$ stat /
  File: /
  Size: 4096            Blocks: 8          IO Block: 4096   directory
Device: 801h/2049d      Inode: 2           Links: 24
Access: (0755/drwxr-xr-x)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2018-10-19 16:42:52.883665976 -0600
Modify: 2018-10-03 06:16:24.676776922 -0600
Change: 2018-10-03 06:16:24.676776922 -0600
```

```
shaun@shaun-VirtualBox:/$ stat /run
  File: /run
  Size: 840             Blocks: 0          IO Block: 4096   directory
Device: 16h/22d Inode: 2           Links: 26
Access: (0755/drwxr-xr-x)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2018-10-17 11:41:56.045435149 -0600
Modify: 2018-10-19 11:11:20.395016659 -0600
Change: 2018-10-19 11:11:20.395016659 -0600
```

Directory Structure

- Single-Level Directory --- All files are contained in the same directory





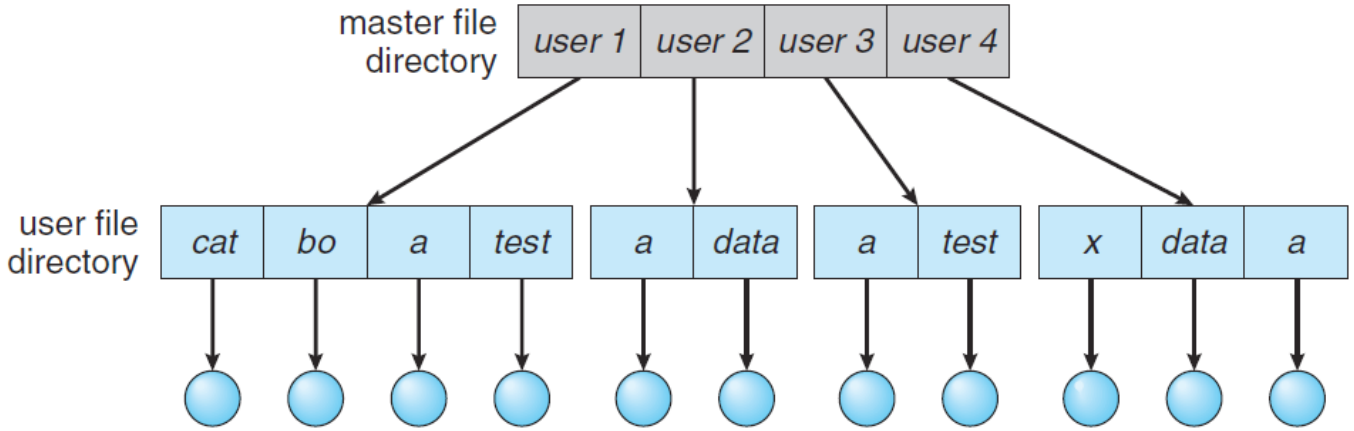
Single Level Directory

- All files are in the same directory, they must have unique names.
  - ✓ If two users call their data file as test.txt, then the unique name rule is violated.
  - ✓ Users have to check the uniqueness of the names when they try to name files.

- Searching files under single directory is not efficient.
  - ✓ If a user tries to search a file. It has to search over all the users' files.

Directory Structure

- Two-Level Directory --- Separate directory for each user

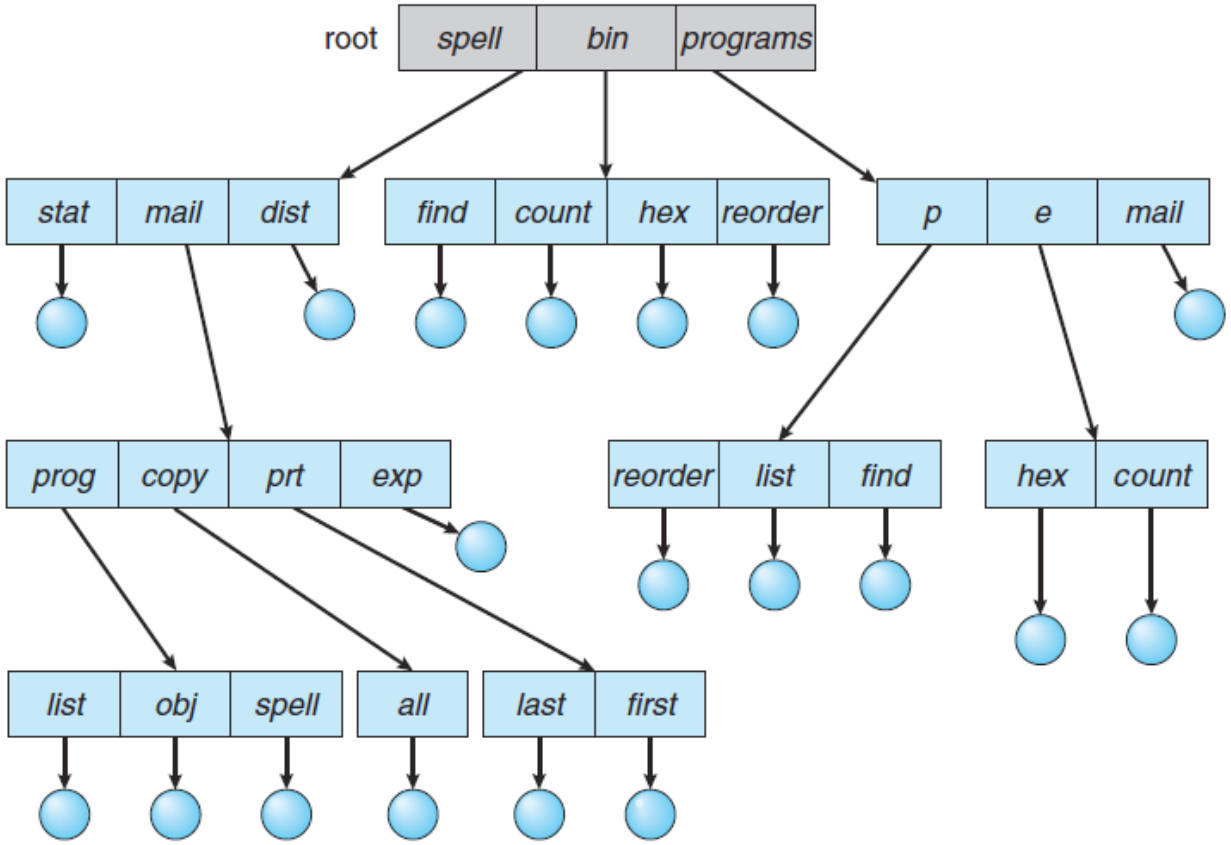


- Each user has its own user file directory (UFD).
  - ✓ Different users can choose the same file name, i.e., file names should be unique within each UFD.
- Each file can be identified by using username plus filename/UFD.
- A user searching a file is relative efficient.
- However, if the users want to cooperate on some task and to access one another's files, two-level directory is not efficient/possible to achieve it.

Directory Structure

Tree-Structure Directory

- Users are allowed to create their own subdirectories and to organize their files.
- The tree has a root directory, and every file in the system has a unique path name.
- Each directory (or subdirectory) contains a set of files/subdirectories.
- A directory itself is considered as a file of the system to provide information about files.
- A process has a current directory (working dir).
  - ✓ When the process refers to a file using a simple file name or relative path, the reference is interpreted relative to the current working directory of the process.



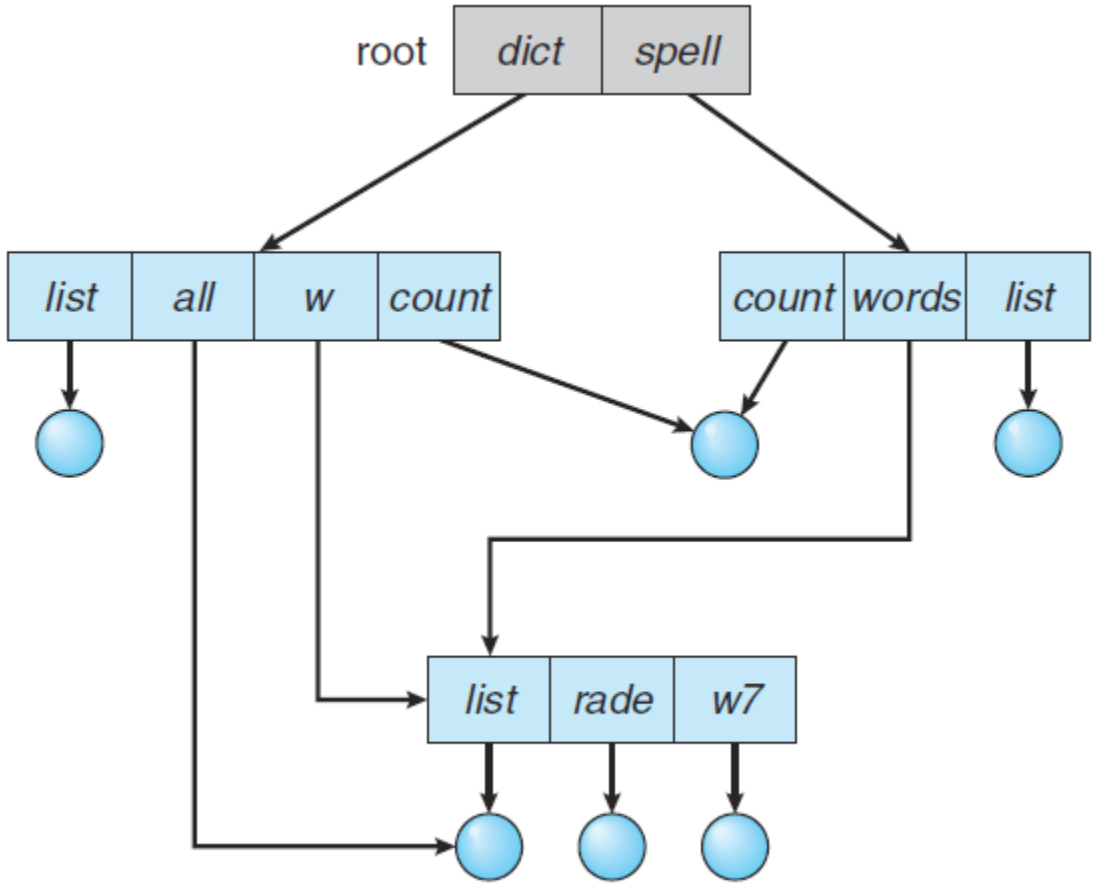
Directory Structure

Acyclic-Graph Directory—have shared subdir and files

- The same file or subdirectory may be in two different directories. That is, a file or subdirectory is shared by two directories
  - ✓ A "count" file is in two different directories.

- New directory entry "link" is designed to achieve shared file or subdirectory.
  1. Symbolic link
     - ✓ A symbolic link is considered as a pointer to another file or subdirectory.
     - ✓ A symbolic link could be implemented as an absolute or a relative path name, e.g., /spell/count.
     - ✓ Using the path name can resolve the symbolic link to locate the shared file or subdirectory.

  2. Non-symbolic link
     - ✓ Duplicate all information about the shared file or subdirectory.



Directory Structure

Symbolic link and non-symbolic link in Unix

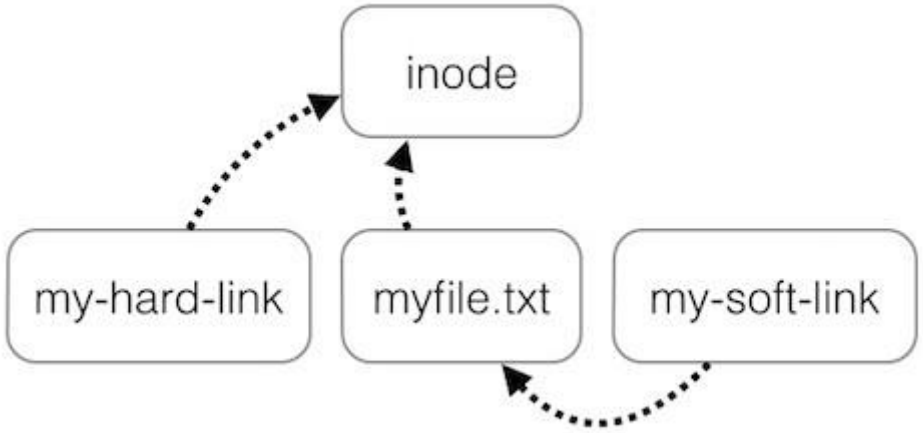
- In Unix, when a file is created in a directory, the file name and an inode with inode number are assigned to that file.

Non-symbolic link (hard link)

- ✓ share the same inode number with their original file. They point to the same inode of a hard drive.

Symbolic link (soft link)

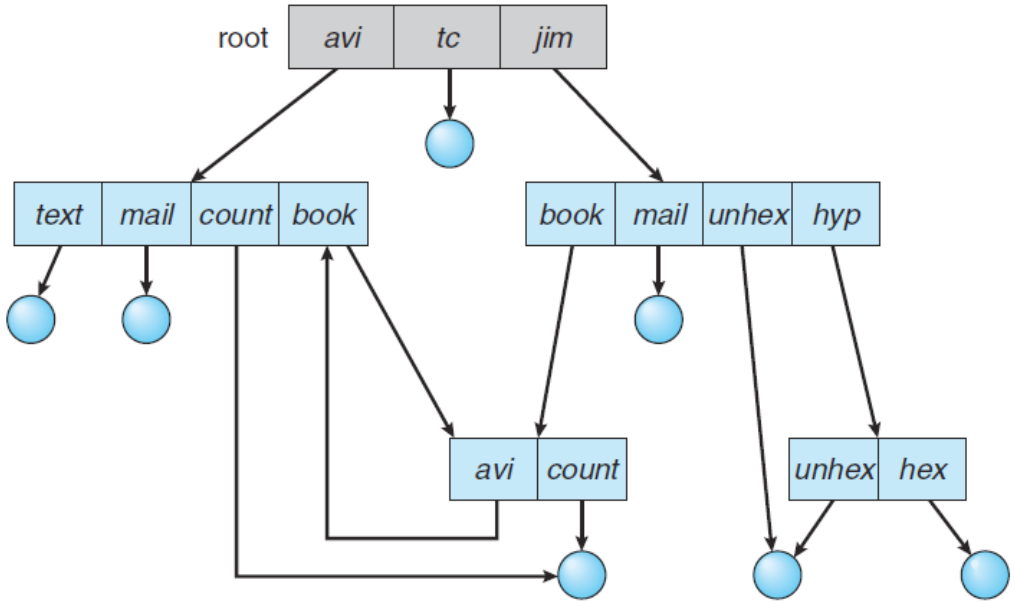
- ✓ points to the name of another file and does not contain any data of their own.



Directory Structure

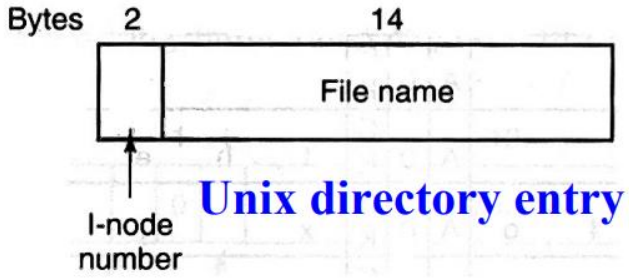
Acyclic-Graph Directory—have shared subdir and files

- Cycles in Acyclic-Graph Directory
  - ✓ Acyclic-Graph Directory is no long a tree structure. It may incur cycles.
  - ✓ Cycles may incur infinite loop once the system tries to search a file.
- How to guarantee no cycles?
  - ✓ Allow only links to file not subdirectories

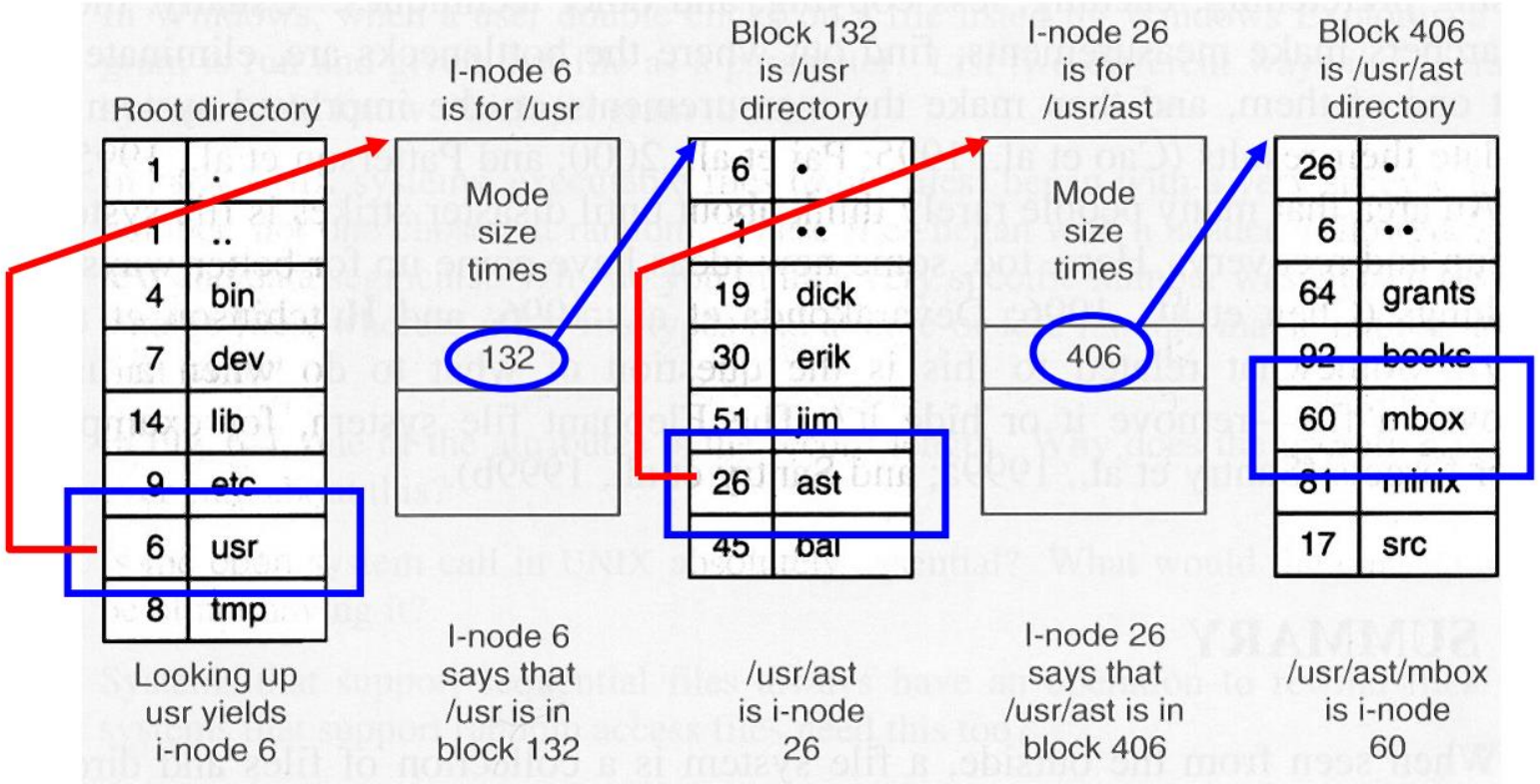


Directory Structure

- One example of directory in Unix



find /usr/ast/mbox

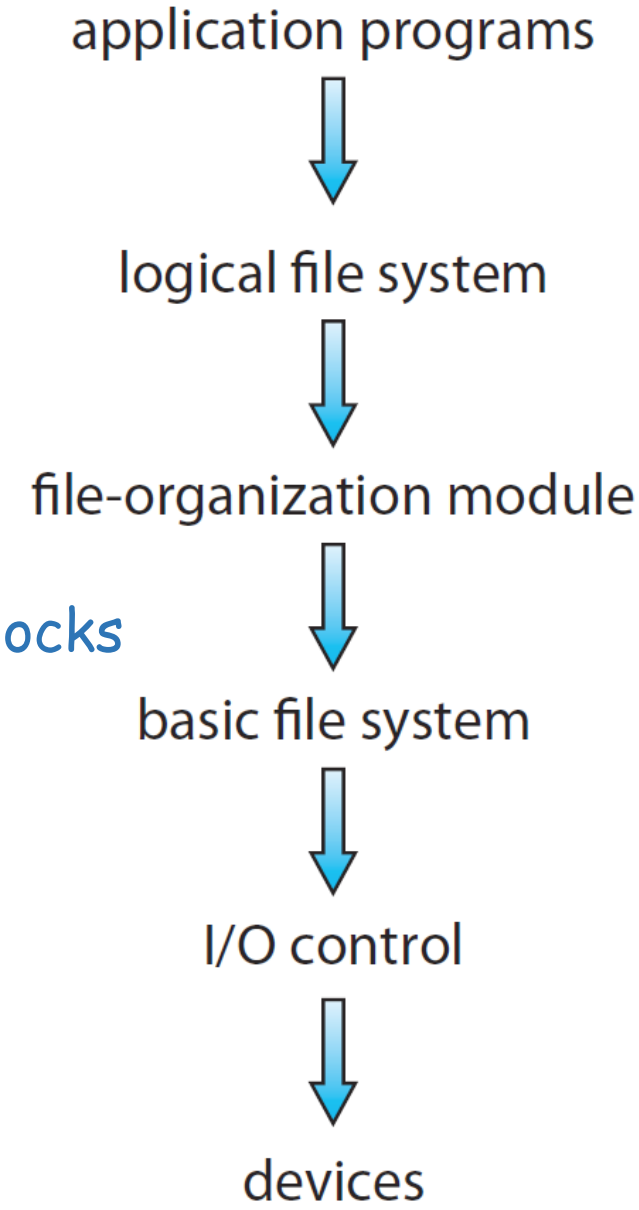


File System Structure

- File system resides on secondary storage (e.g., disks)
  - Provides user interface to storage, mapping logical file blocks to physical blocks (usually 512 B)
  - Provides efficient and convenient access to disk by allowing data to be stored, located retrieved easily.
- Some important data structure in a file system
  - Volume control block—contains volume (partition) details
    - ✓ Total # of blocks, # of free blocks, block size, free block pointers in a volume.
  - Directory structure—contains information on how to organizing files
    - ✓ For example, in Unix, directory table (containing file names and inode number) represents directory structure
  - Per-file File Control Block (FCB) —contains details (e.g., attributes) about the file
    - ✓ For example, in Unix, each inode is the file control block.

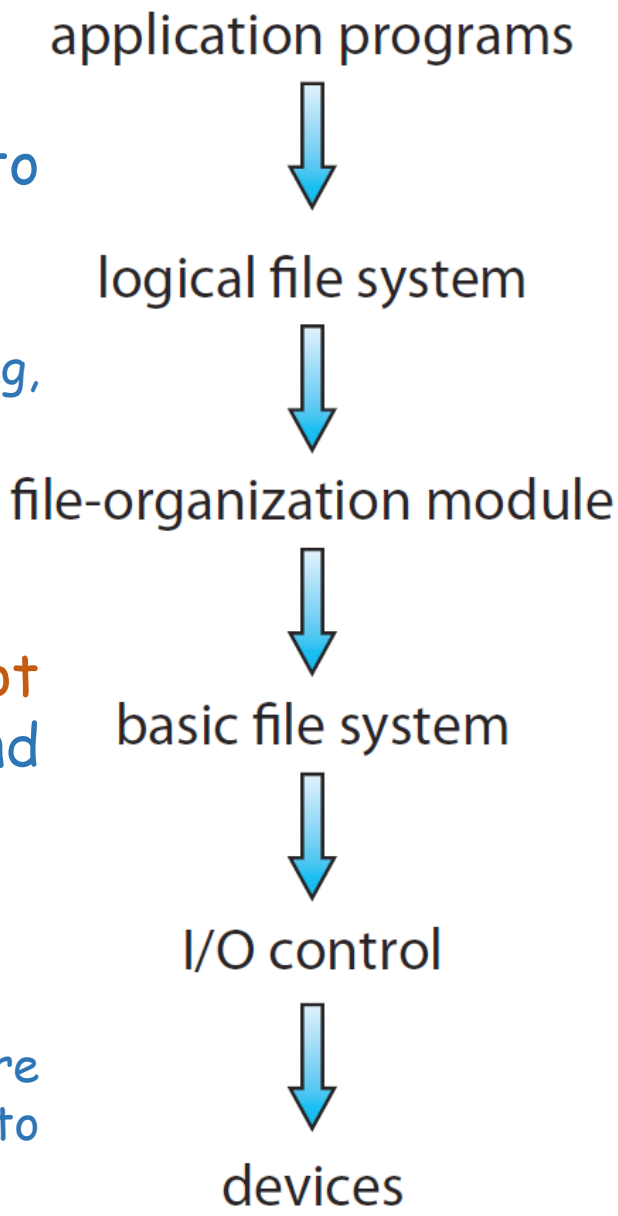
File System Structure

- File system organized into layers
  - Logical file system -manages metadata of files (directory structure)
    - ✓ Maps file names into file control blocks (e.g., inode).
    - ✓ Manages the directory of the file system.
    - ✓ Manages protection and security of the file system.
  - File-organization module – understand about file, logical blocks and physical blocks
    - ✓ Mapping logical block numbers/addresses into physical block numbers/addresses.
    - ✓ Manage free space (tracks unallocated blocks).



File System Structure

- File system organized into layers
  - Basic file system -- gives commands to the device driver to read/write physical block on the disk
    - ✓ Also manages memory buffers and caches (allocation, freeing, replacement of buffers)
      - ❖ Memory buffers hold data in transit.
      - ❖ Caches hold frequently used data.
  - I/O control layer--consists of device drivers and interrupt handlers to transfer information between main memory and the disk
    - ✓ A device driver can be thought of as a translator.
    - ✓ Its input consists of high-level commands such as "retrieve block 123"
    - ✓ Its output consists of low-level, hardware-specific instructions that are used by the hardware controller, which interfaces the I/O device to the rest of the system



File System Structure

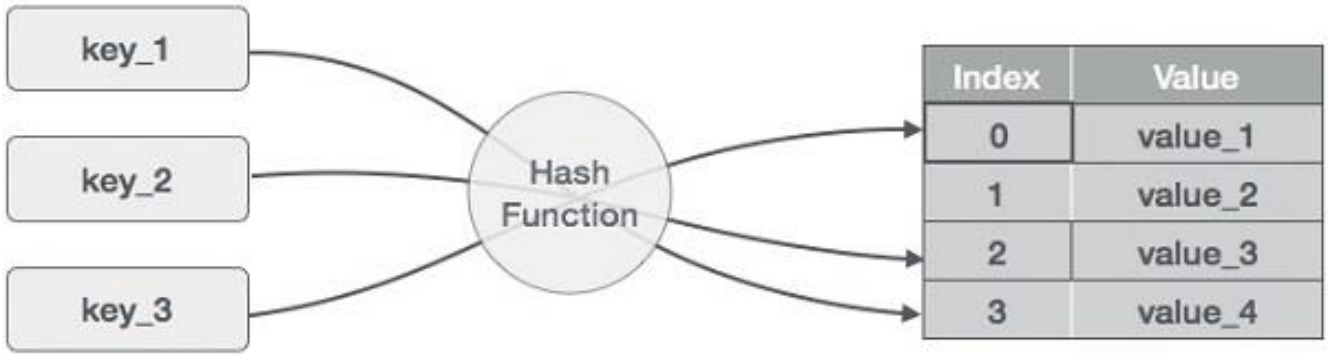
Directory Implementation

- If we create/delete/updating a file, how to reflect it into directory
- Two methods are used:
  - Linear List
  - Hash Table
- Linear list: all the files in a directory are maintained as single lined list.
  - ✓ When a new file is created, then the entire list is checked whether the new file name is matching to a existing file name or not. In case, it doesn't exist, the file can be created at the beginning or at the end.
  - ✓ Searching for a unique name is a big concern because traversing the whole list takes time.
  - ✓ The list needs to be traversed in case of every operation (creation, deletion, updating, etc.) on the files therefore the systems become inefficient.

File System Structure

❑ Directory Implementation

➢ Harsh table: A key-value pair for each file in the directory gets generated and stored in the hash table.



- key_1/ key_2/ key_3 are the file names
- value_1/value_2/value_3 are the inode numbers

✓ Decreasing the directory searching time.
✓ May incur collision: two different file names hash to the same value.
✓ Hash tables are generally fixed size, thus reducing its flexibility.

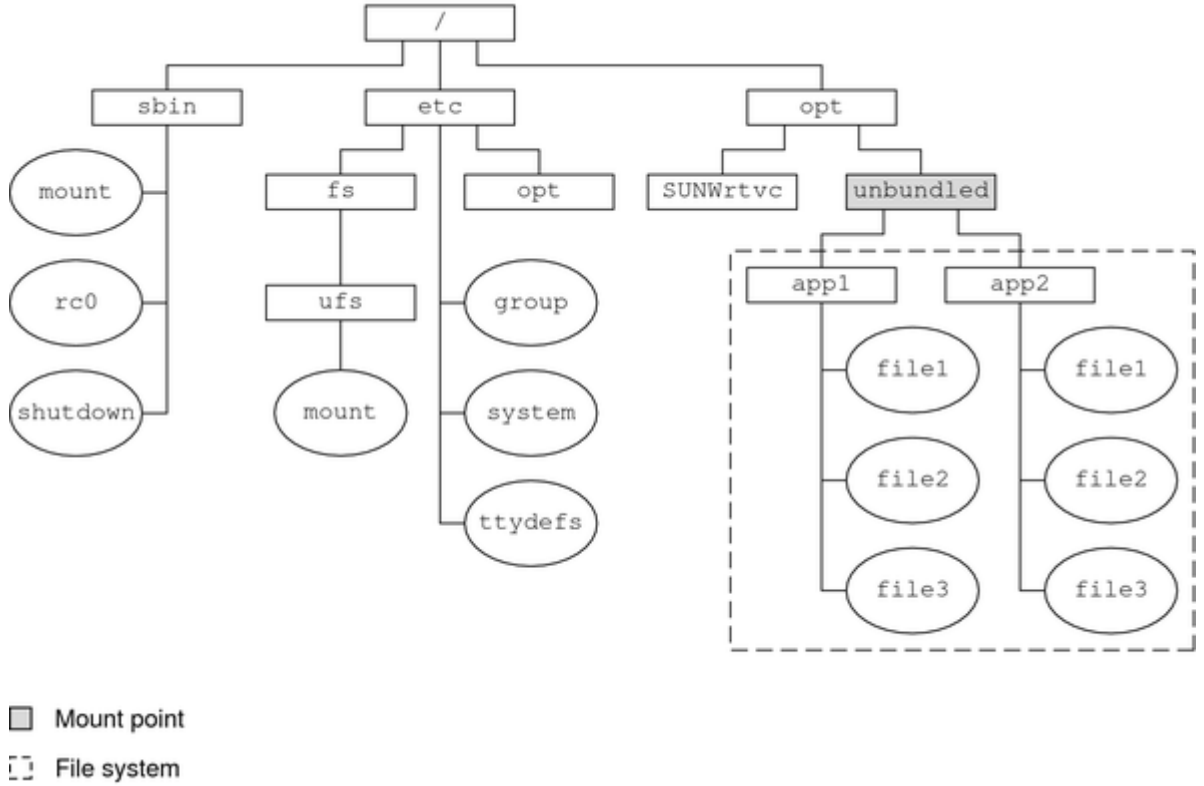
File System Mounting

- An OS may have more than one file systems (devices), and file system must be mounted before it can be available to processes on the system.

- Once a file system is mounted, the OS will give the name (e.g., device ID), mount point, and type (option) of the file system.

  - A mount point is a directory (typically an empty one) in the currently accessible filesystem on which an additional filesystem is logically attached.
  - To access a local file system, you have to

    - ✓ Create a mount point
    - ✓ Mount the local file system by using the mount command



In-Memory File System

- In addition to on-disk file structures, several file system structures are maintained in memory as well.
  - A system-wide open-file table that contains a copy of the FCB for each open file.
  - A per-process open-file table that contains a pointer to the appropriate entry in the system-wide open-file table.
  - A mount table that stores file system mounts, mount points, file system types.
  - Cached portions of the directory structure
    - For example, in Unix, in-memory inode table is applied, where recently accessed inode data are cached in in-memory inode table.
  - Buffers for assisting in the reading/writing of information from/to disk.

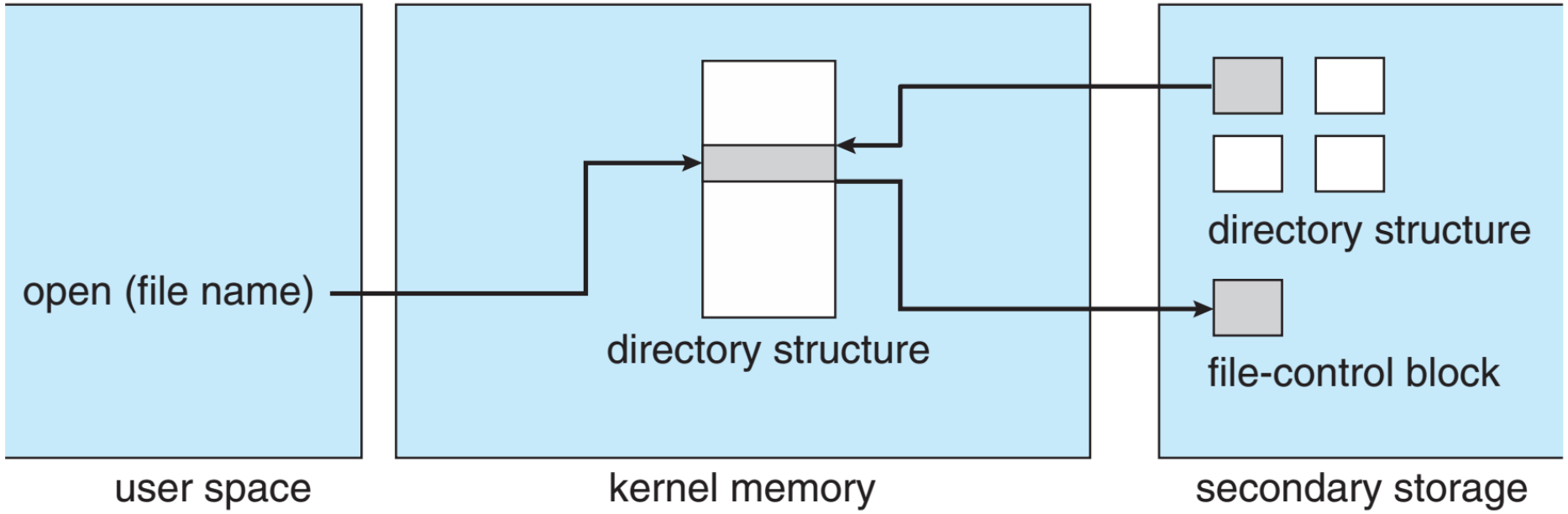
In-Memory File System

❑ Example of mount table in Unix

```
shaun@shaun-VirtualBox:/$ cat /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/sda1 during installation
UUID=3baf0097-3e93-41f2-b995-aa779281a991 /               ext4    errors=remount-ro 0       1
/swapfile                                 none            swap    sw                0       0
shaun@shaun-VirtualBox:/$
```

- Filesystem: indicate the UIUD (Universally Unique Identifier) of disk for the file system.
- Mount point: a mount point is a directory in the currently accessible filesystem on which an additional filesystem is mounted.
- Type: the type of filesystem
- Option: describe the mount options associated with the filesystem
- Dump: whether the filesystem should by dumped by the "dump" command. A value of "0" indicates that no dump is to take place.
- Pass: whether the filesystem checks are done at reboot time. The "/" root filesystem should have a value of "1" and other filesystems should have a value of "2". If this filed is not present or has a value of "0" then it is assumed that no check is to take place.

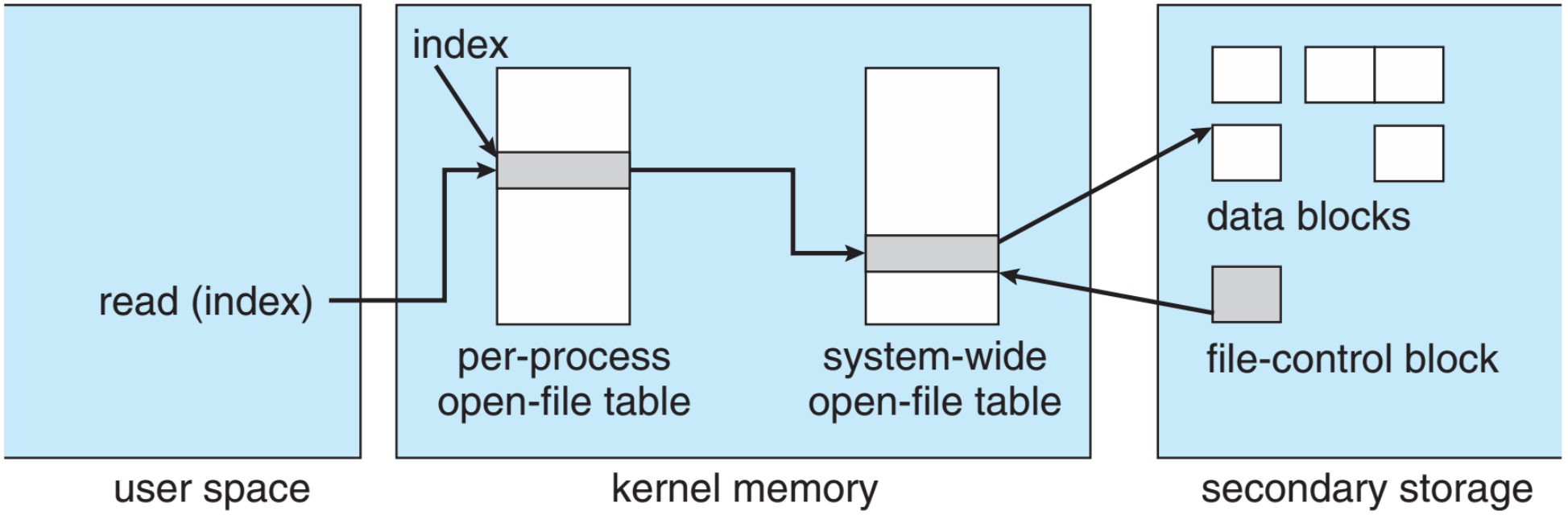
In-Memory File System – Open Files



- The open() call passes a file name to the logical file system, which first searches the system-wide open-file table to see if the file is already in use by another process.
  - ✓ If it is, a per-process open-file table entry is created pointing to the existing system-wide open-file table.
  - ✓ If not, the directory structure is searched for the given file name. Parts of the directory structure are usually cached in memory to speed directory operations

- Once the file is found, the FCB is copied into a system-wide open-file table in memory.
  - ✓ System-wide open-file table not only stores FCBs but also tracks the number of the processes that have the file open.

In-Memory File System – Read/Write/Close Files



- Once a file is opened, an entry is made in the per-process open-file table, with a pointer to the entry in the system-wide open-file table and some other fields (e.g., a pointer to the current location in the file), which can be used for the read() and write() operation.

- When a process closes the file, the per-process table entry is removed, and the system-wide entry's open count is decremented.

- When all users that have opened the file close it, any updated metadata is copied back to the disk-based directory structure, and the system-wide open-file table entry is removed.

In-Memory File System – Read/Write/Close Files

- File descriptor
  - A file open operation gets a file descriptor, assigned by kernel.
  - A file descriptor is a non-negative integer.
  - Be used to identify the opened file with following file operations.
  - Three file descriptors i.e., stdin, stdout, stderr, (with values of 0, 1, 2) are automatically generated when a process is created.