An Overview of File Subsystem

Inode – Information Node

- In **Unix based** operating system each file is indexed by an **Inode**.
- Inode are special disk blocks they are created when the file system is created.
- The number of Inode limits the total number of files/directories that can be stored in the file system.

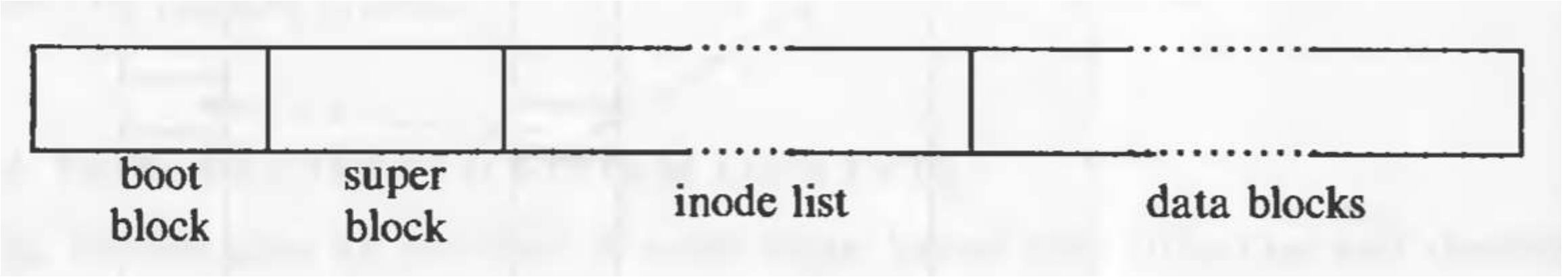
Inode – Information Node

- Internal representation of a file is given by an *inode*
  - Numeric UID of the owner.
  - Numeric GUID of the owner.
  - Size of the file.
  - File type: regular, directory, device etc…
  - Date and Time of Last modification of the file data.
  - Date and Time of Last access of file data.
  - Date and Time of Last change of the I-node.
  - Access Permissions
  - Pointer to the file in the disk

$ stat <filename> - Displays the details in inode

```
[nima@localhost ~]$ stat f6
  File: ‘f6’
  Size: 27              Blocks: 8          IO Block: 4096   regular file
Device: fd00h/64768d    Inode: 722678      Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/    nima)   Gid: ( 1000/    nima)
Context: unconfined_u:object_r:user_home_t:s0
Access: 2022-01-17 12:18:14.776000000 -0500
Modify: 2021-11-16 01:08:58.541000000 -0500
Change: 2021-11-16 01:08:58.541000000 -0500
```

File System Structure



File System Structure

- The *boot block:*
  - Occupies the beginning of a file system, typically the first sector.
  - Contain the *bootloader*

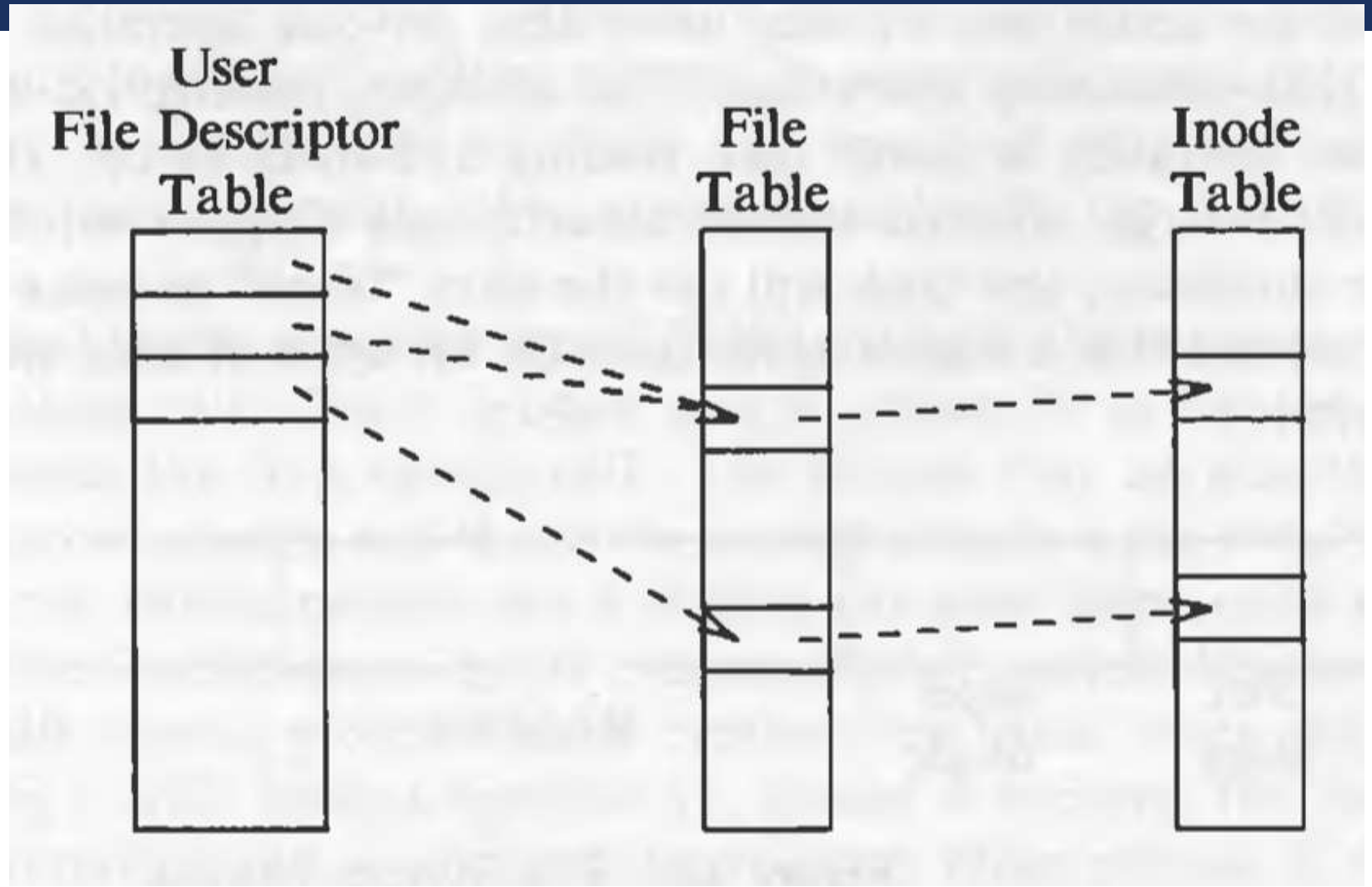
File System Structure

- The *super block*
  - Describes the state of a file system
    - Total size of the partition
    - List of free space
    - Name of partition
    - Time modified

Inode & Data Block

- Inode Block
  - Contains list of Inodes
- Data Block
  - Contains users files.

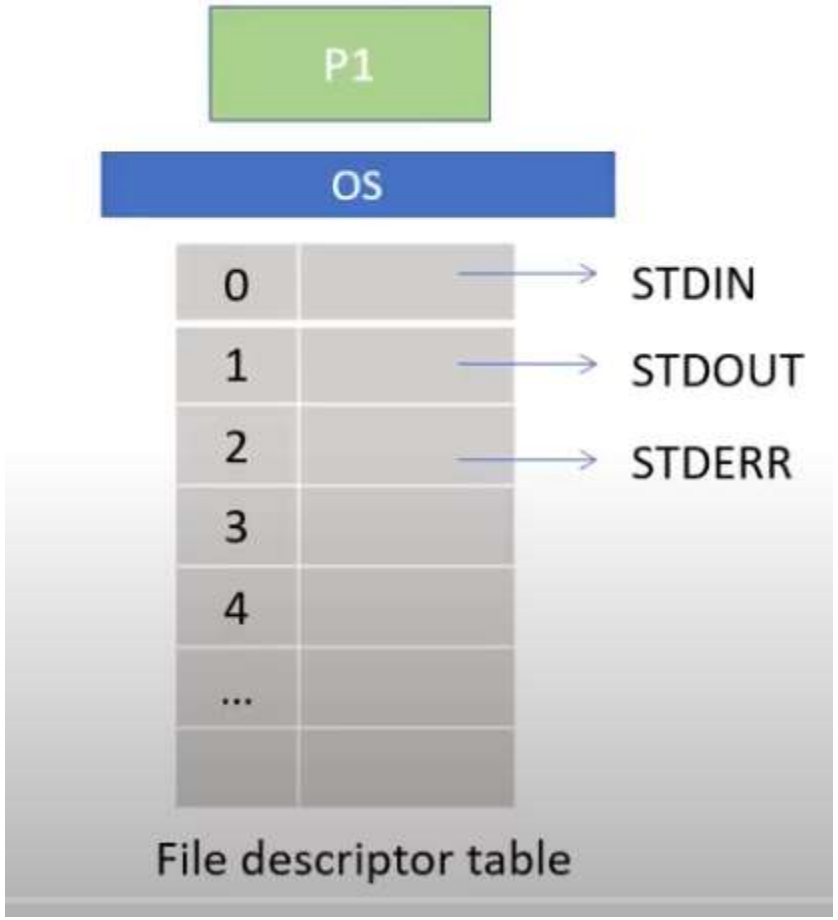
File Data Structures



User File Descriptor Table

fd = open(pathname, flags, mode);

- Flags controlling the operation of the file descriptor.
- A pointer/reference to the open file table.
- It is local to every process and contains information like the identifiers of the files opened by the process.
- Whenever, a process creates a file, it gets an index from this table primarily known as File Descriptor.



File Table

- The status flags specified when the file was opened.
- The file access mode.
- The current file offset (in bytes, from the start of the file)
- A pointer/reference to the i-node table

Inode Table

- File type (regular file, FIFO, socket, . . .)
- File permissions
- Other file properties (size, timestamps, . . .)

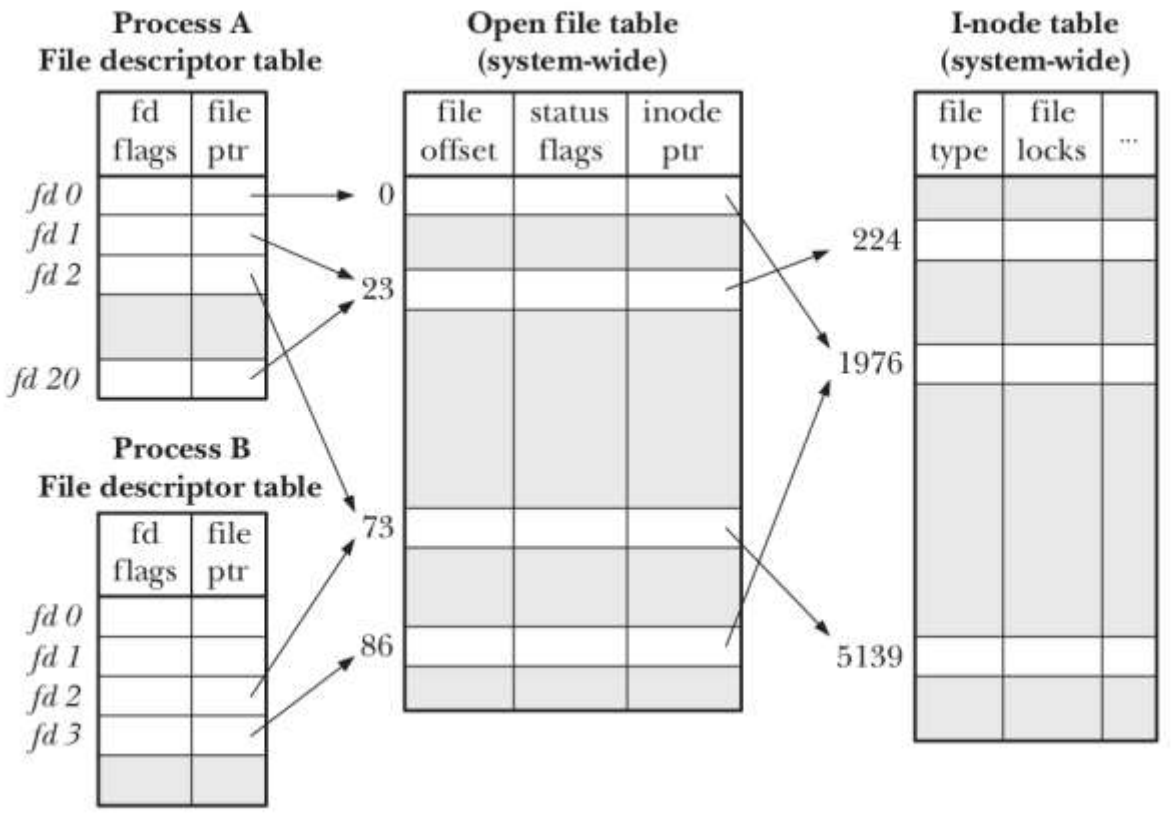


Figure 5-2: Relationship between file descriptors, open file descriptions, and i-nodes

Multiple processes can have FDs that points to same File Table Entry

Multiple File Table entry can point to the same i-node table entry

SYSTEM CALLS FOR FILE SYSTEM

- Open System Call
- Open is the first step to access data in a file.
- **fd = open (pathname, flags, mode);**
  - flags indicate the type of open (reading or writing) and mode gives the permissions if the file is being created.
  - It returns an integer called the user file descriptor.
  - Other file operations use the file descriptor returned by open.

Flags

`O_RDONLY` : Open for read-only

`O_WRONLY` : Open for write-only

`O_RDWR` : Open for read and write.

`O_CREAT` : Create the file if it doesn't exist.

`O_TRUNC` : Truncate the file to zero length if it exists.

`O_APPEND` : Append data to the end of the file.

`O_EXCL` : Used with `O_CREAT` , returns an error if the file already exists.

Open () system call

```
int fd = open(filename, O_WRONLY | O_CREAT |
O_TRUNC, 0644);
```

- Returns a non-negative integer, which is the file descriptor.
- On failure, it returns -1 , and you can use the errno variable to determine the specific error.

Read() system call

Number = read (fd, buffer, count)

- where fd is the descriptor returned by open.
- A pointer to the buffer where the read data will be stored.
- Count is the maximum number of bytes to be read.
- And it returns how many bytes were successfully read.

Write() System Call

- Write

number = write (fd, buffer, count);

The function signature is exactly same to that of read

```
#include<unistd.h>
int main()
{
write(1,"hello\n",6);
}
```

Close() System Call

A process closes an open file when it no longer wants to access it.

Close (fd);

CREATE() SYSTEM CALL

- The creat system call creates a new file in the file system.
-
- fd = creat (pathname, modes);
-
- If pathname of an existing file is passed to creat, it will truncate the file and set its size to 0