POSIX File Systems

Peter Chapin
Vermont State University

What is an (Old-Style) Disk Partition?

- Holds raw data in *sectors*
  - Multiple *platters* (usually two sided)
  - On each platter are multiple tracks
    - Corresponding tracks on each platter form a *cylinder*
  - One head for each surface (typically two per platter)
- Three dimensions
  - (head, cylinder, sector)
- Sectors are small
  - Typically 512 bytes

Disk Driver

- Driver presents simple model
  - Partition appears as a large array of blocks.
    - Each block typically multiple sectors (say, 8 sectors for a total of 4096 bytes)
  - Driver translates a block number into disk coordinates.
    - e.g., block 37892 => starts at (head 5, cylinder 249, sector 16)
  - Driver accepts request for a particular block...
    - Controls the disk hardware to access appropriate sectors.

Driver Concerns

- The disk driver wants to optimize disk access
  - Requests for blocks sit in a queue
  - Driver chooses requests in some optimal way
    - Not necessarily FIFO order
    - May choose requests that are "close" to the heads' present position
  - Much can be said about this... not an issue for right now
- *Higher levels of the OS request blocks from the driver!*

SSD vs HDD

- Many modern systems use solid-state drives (SSDs)
  - Underlying electronics is quite different
  - More like an array of blocks in actuality
- The driver still presents the "array of blocks" abstraction to higher-level software
- More general to refer to *storage devices* rather than drives
  - Many people still say "drive" or "disk"

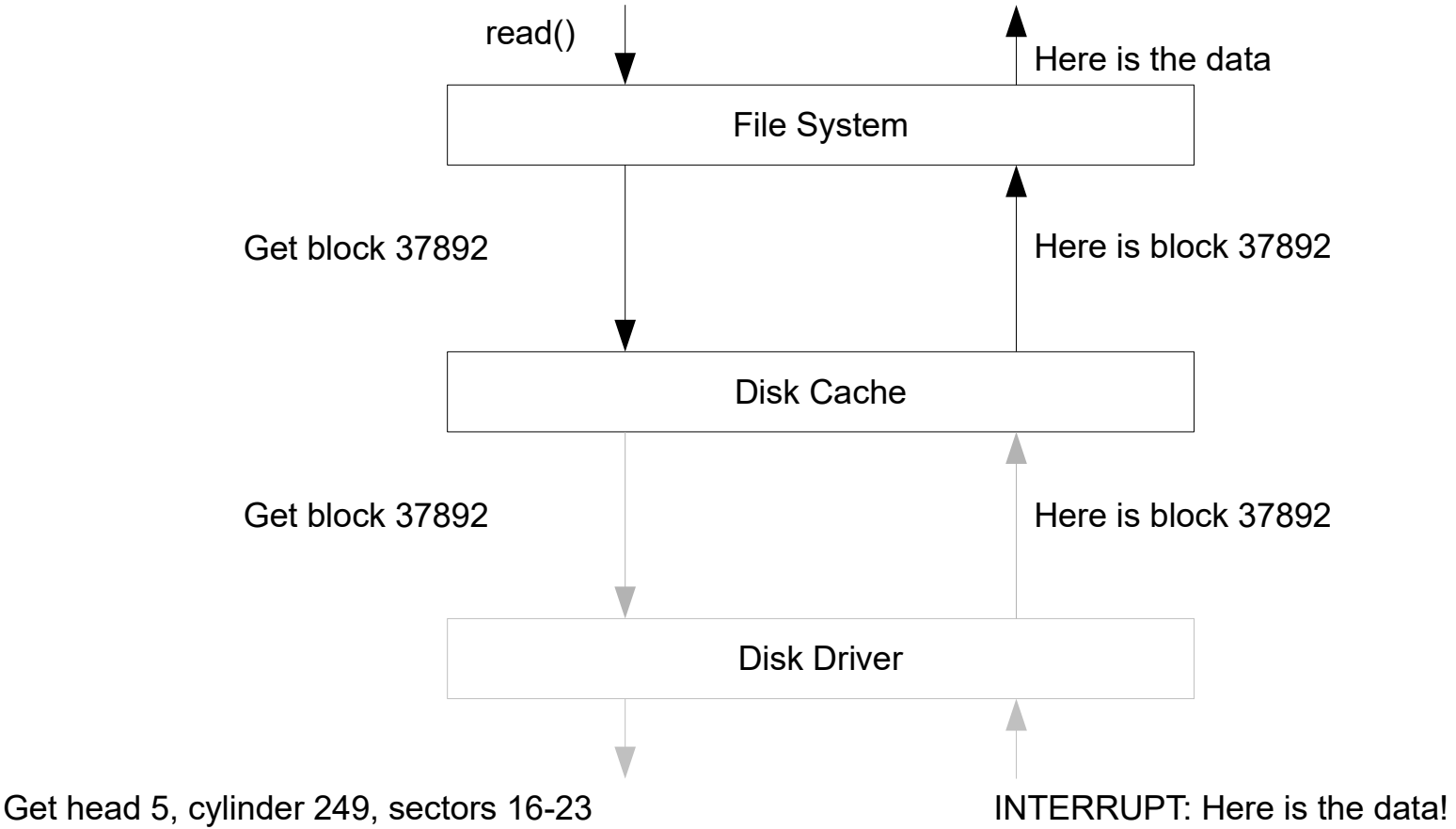
Disk or Partition?

- Disk is sliced into partitions
  - A table on the disk describes their size and extent
  - Driver understands (disk, partition, block#) coordinates
    - Specifies a disk block uniquely on the system
  - Driver changes the block number into (head, cylinder, sector) coordinates.
- Disk drivers number blocks on each partition.
  - Block 37892 on partition 0 is different than block 37892 on partition 1.

What About Files?

- Applications deal with files
  - Descriptive names: `/home/pchapin/afile.txt`
  - Orgainzed in a tree hierarchy
  - Variable size
  - How does a command like "read 1024 bytes from `/home/pchapin/afile.txt`" get converted into "read block #37892?"
- The file system does it!
  - The file system is the part of the OS that understands files and talks to the disk driver.

File System Organization



File System Layout

- File systems control the layout of files on disk
  - Different file systems do it differently
  - Many interesting issues come up
    - Optimizations
      - ... for a large number of small files
      - ... for very large files
      - ... for random access
      - ... for sequential access
    - File system limits
      - How big is the largest file?
      - How large a partition can be used?
      - How long can file names be?

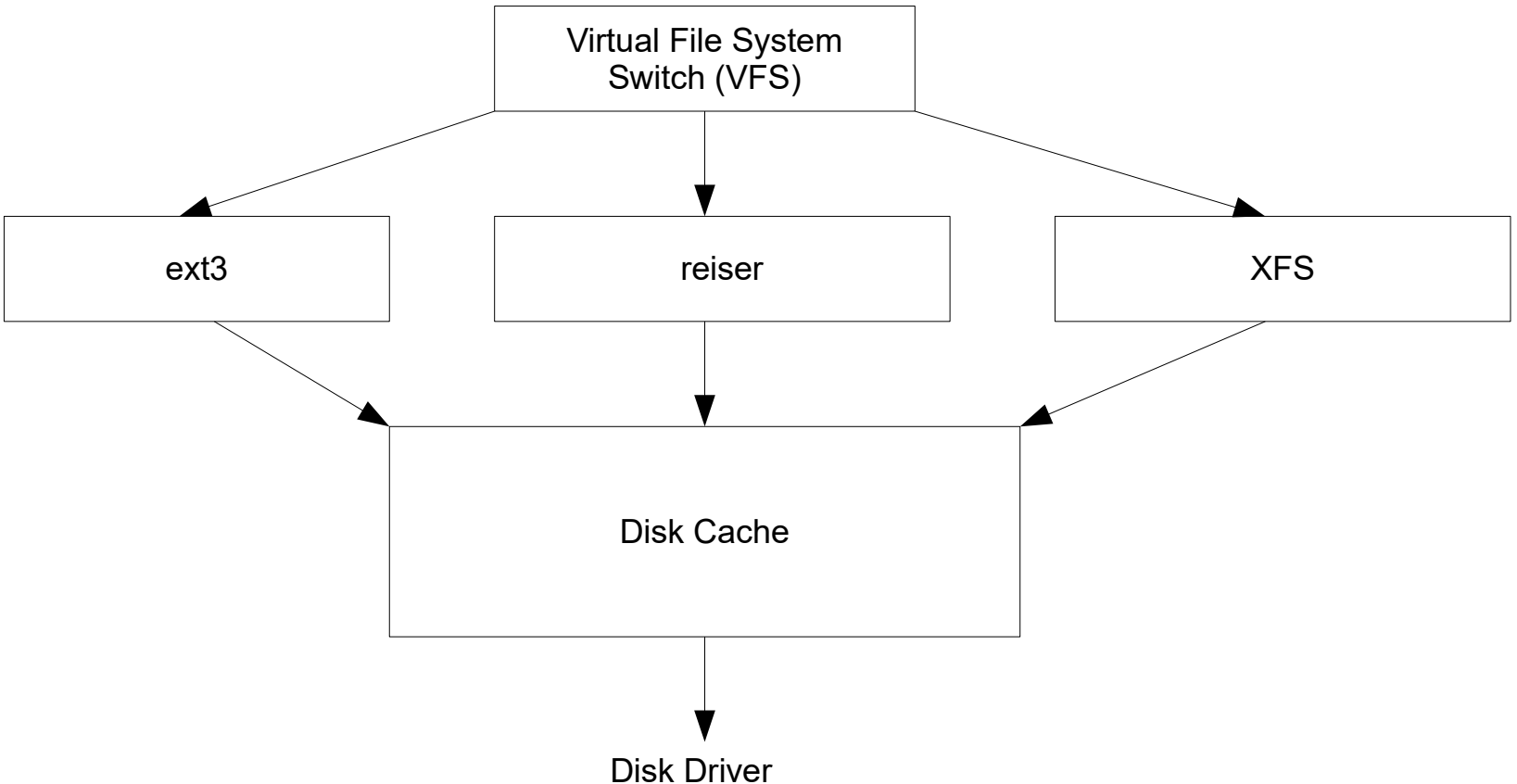
File System Features

- Many features are commonly supported
  - Journals
  - Access control lists
  - Encryption
  - Data compression
  - Extended attributes
  - Versioning
  - *And more!*
- We will talk about some of these features

Example

- Application says "open `/home/pchapin/afile.txt`"
  - File system driver must...
    - Locate root directory on disk. Read it
    - Interpret contents of root directory. Look for "home"
    - Locate `/home` on disk. Read it
    - Interpret contents of `/home`. Look for "pchapin"
    - Locate `/home/pchapin` on disk. Read it
    - Interpret contents of `/home/pchapin`. Look for `afile.txt`
    - If `afile.txt` exists and has appropiate access settings... SUCCESS!
  - File system driver knows where file system structures are and what they look like.

Multiple File System Types



VFS

- In Linux the VFS dispatches requests to an appropriate file system driver
  - Depending on which file is being used...
    - VFS computes which partition the file is on using the system mount points that are active
    - Knows which file system type is on each partition
    - Calls into appropriate file system driver
- The cache and driver know nothing about this
  - They deal with raw disk blocks
  - Don't care about file system layout issues

Basic POSIX Layout Concepts

- Superblock
  - Special block that contains information about the file system layout
- i-nodes ("index nodes")
  - Small data structure containing file metadata
- Free map(s)
  - Tracks which blocks or inodes are available
- Files
- Directories

Superblock

- Contains information about the file system as a whole
  - File system type
  - Size and/or location of other data structures
- Allows the file system driver to manipulate different sized instances of a file system
  - e.g., ext3 on a 100 MiB partition vs a 10 GiB partition
- Sometimes duplicated on the disk for backup purposes
  - If the superblock becomes unreadible, the entire file system is destroyed

I-Nodes

- One i-node for every file and directory
  - Contains metadata (except for the file name)
    - Exact size in bytes
      - Needed since the last block of the file is partially filled
    - Permissions
    - Owner and group association
    - Number of links
      - A file can appear in multiple directories
    - Information for finding the file contents
- Number of i-nodes fixed when disk formatted?
  - Not necessarily! Advanced systems allow dynamic allocation

Free Maps

- Where is the free space?
  - Must track which blocks are used
    - Typically a "free map" uses a single bit to represent a block
      - If the bit is clear the block is free. If the bit is set the block is used
  - Must track which inodes are used
    - Another free map used to track i-nodes the same way

Files

- The i-node contains information about file
  - Block #s of the first part of the file stored directly in the i-node
    - BUT... the i-node is small so not many block #s will fit
  - Block # of a block full of block numbers
    - Called the *first indirection pointer*
  - Block # of a block full of first indirection pointers
    - Called the *second indirection pointer*
  - Block # of a block full of second indirection pointers
    - Called the *third indirection pointer*
- See GenericFS documentation for details!

Directories

- Directories are like files
  - Just a large array of bytes
  - Managed like files internally
- Contain a list of (name, i-node) associations
  - For each file named, the i-node controlling that file is specified
  - When a file is opened the i-node is brought into kernel memory
  - The name of the file is not important after that

Tools (`ls`)

- To view the i-node numbers use the `-i` option with `ls`

```
pchapin@lemuria:~/Projects/GenericFS/doc$ ls -i
655666 Compiling-Linux.tex          655715 doc-Implementation.tex
655668 DevBox-HackBox.tex           655934 doc-Preliminaries.tex
655696 doc-GenericFS-Structure.tex  655669 Figures
```

This is a directory.
Directories are a special kind of file and also have i-node numbers

Tools (`stat`)

- The `stat` program dumps the information in the i-node except for the information about how to locate the file on disk.

```
pchapin@lemuria:~/cis-4020$ stat readme.txt
  File: readme.txt
  Size: 2322            Blocks: 8          IO Block: 4096   regular file
Device: fd00h/64768d    Inode: 28573718    Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/ pchapin)   Gid: ( 1000/ pchapin)
Access: 2024-10-22 11:11:17.209971399 -0400
Modify: 2024-10-22 11:11:16.881954633 -0400
Change: 2024-10-22 15:03:56.687710599 -0400
 Birth: 2024-10-21 11:49:53.333608251 -0400
```