Files and Directories

2023 Fall COMP3230A

Contents

- Overview of storage disks
- Overview of file systems
- What is a file?
- What is a directory?

Related Learning Outcome

- ILO 2d - describe the principles and techniques used by OS to support persistent data storage

Readings & References

- Required Reading
  - Chapter 39 – **Interlude: Files and Directories**
    - http://pages.cs.wisc.edu/~remzi/OSTEP/file-intro.pdf

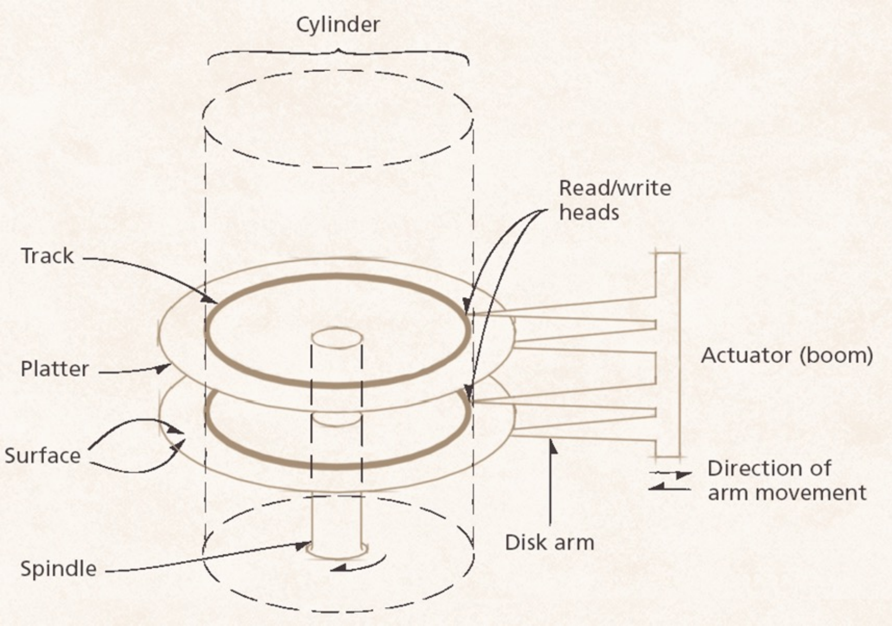
- References
  - Chapter 36 – **I/O Devices**
    - http://pages.cs.wisc.edu/~remzi/OSTEP/file-devices.pdf
  - Chapter 37 – **Hard Disk Drives**
    - http://pages.cs.wisc.edu/~remzi/OSTEP/file-disks.pdf
  - Chapter 44 – **Flash-based SSDs**
    - http://pages.cs.wisc.edu/~remzi/OSTEP/file-ssd.pdf
  - The SSD Anthology: Understanding SSDs and New Drives from OCZ
    - http://www.anandtech.com/show/2738
  - How do SSDs work?
    - http://www.extremetech.com/extreme/210492-extremetech-explains-how-do-ssds-work

Secondary Storage

- Most secondary storage devices involve magnetic disks, which are random-access storage
  - Data can be accessed by read-write head in any order
- Disk drives (magnetic or solid-state) are part of a class of storage called block devices.
  - These devices treat the storage space as a large 1-dimensional arrays of logical disk blocks, in which, the logical block is the smallest unit of data transfer
  - Commonly-used disk block size is 4 KiB
  - The logical blocks are addressed from 0 to N-1, where the disk has N logical disk blocks

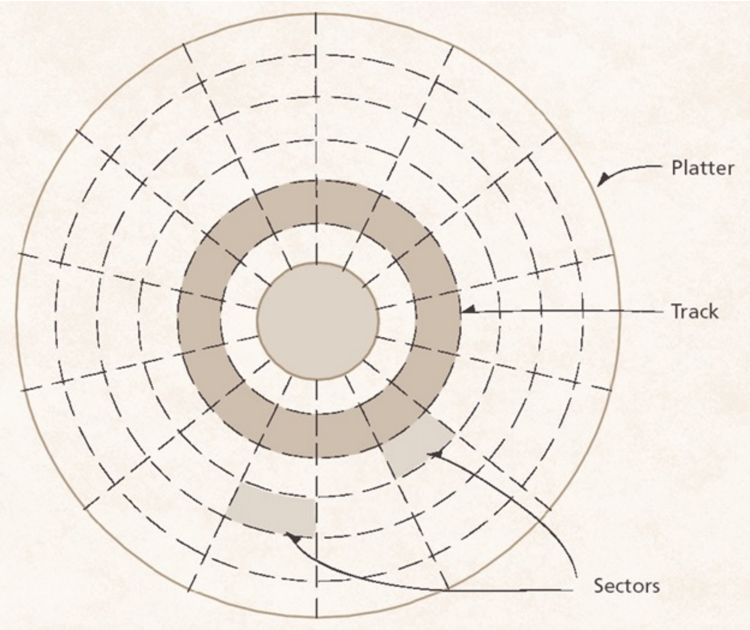
Physical layout of HDD

- A disk consists of a number of magnetic platters with recording surfaces on both sides
- Rotate on spindle
- Each surface is divided into a number of concentric tracks
- Each track is divided in to a number of sectors
- Vertical sets of tracks form cylinders
- Each surface has a disk head for reading and writing

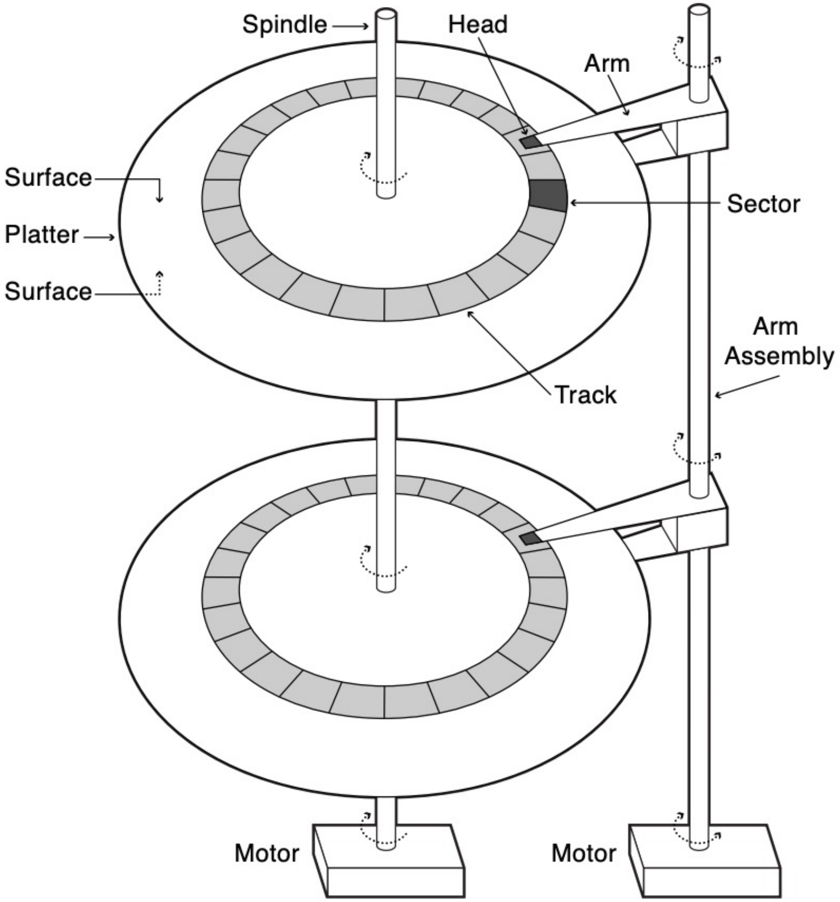


Physical layout of HDD

- Each sector typically contains 512 bytes
- The unit of I/O operation is a **logical block**, typically of 4 KiB, which maps onto the sector(s)
  - Convert a logical block number into a cylinder #, a head #, and a sector #



HDD


Spindle
Head
Arm
Surface
Platter
Surface
Sector
Arm Assembly
Track
Motor
Motor

Performance Characteristics of HDD

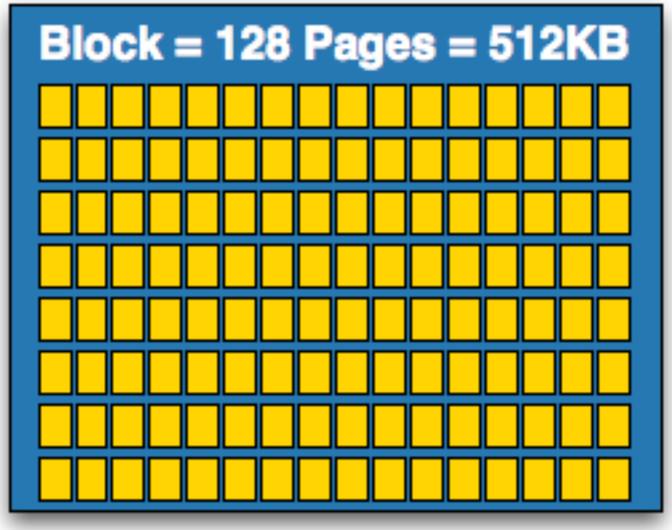
- The data in a particular disk sector can be read/written
- To access a data block
  - Disk arm must move to the target track; then rotate the disk to put target sector under the read-write head; then record is read-from/write-to the disk
- Performance characteristics
  - Seek time
    - Time for read-write head to move to target track from current location
    - average seek times is around 0.5 to 2 milliseconds
  - Rotational latency
    - Time for rotate the platter until the target sector is underneath read-write head
    - depends on the spinning rate; roughly around 2 ms
  - Transfer time
    - Time for further rotate the head to read/write the entire sector and transfer the data

Performance Considerations of HDD

- Ways to improve disk I/O performance
  - Disk scheduling
    - In multiprogramming environment, multiple processes can generate I/O requests at the same time, there may have several pending requests queued up at the disk queue
    - Which request should the system do first?
      - Because of the high cost of I/O, the OS historically played a role in deciding the order of I/Os issued to the disk
      - To optimize the data transfer with the minimum mechanical motion - seek time and rotational time
  - Caching
    - A disk cache buffer (in main memory) is used to temporarily hold disk data
  - Defragmentation
    - Place related data in contiguous sectors
    - Decreases number of seek operations required
  - Multiple disks
    - Disk I/O performance may be increased by spreading the operation over multiple disks

Physical layout of SSD

- SSD storage medium is called NAND flash memory
- Flash memory is non-volatile memory, which is organized as a grid of storage cells
  - Depends on the technologies, each cell can store 1 bit or 2 bits or 3 bits (or even more)
- A group of cells is organized into a "page", which is the smallest structure that's readable/ "writable" in a SSD
  - Today 4KiB (or 8KiB) pages are common on SSDs
- Pages are grouped together into blocks
  - It's common to have 128 pages in a block (512KiB in a block)



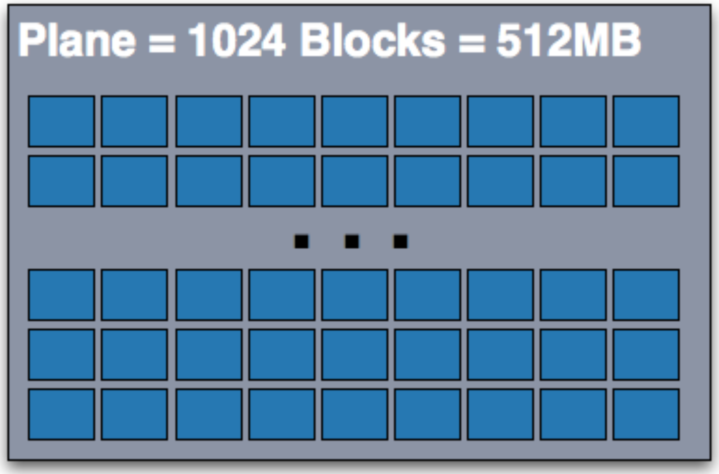
(Source: The SSD Anthology)

Physical layout of SSD

- Blocks are then grouped into planes, and you'll find multiple planes on a single NAND-flash chip

- A block is the smallest structure that can be erased in a NAND-flash device
  - You can read from and "write" to a page
  - But you cannot rewrite to a page unless erasing the whole block (128 pages at a time!!)
  - This is where many of the SSD's problems stem from

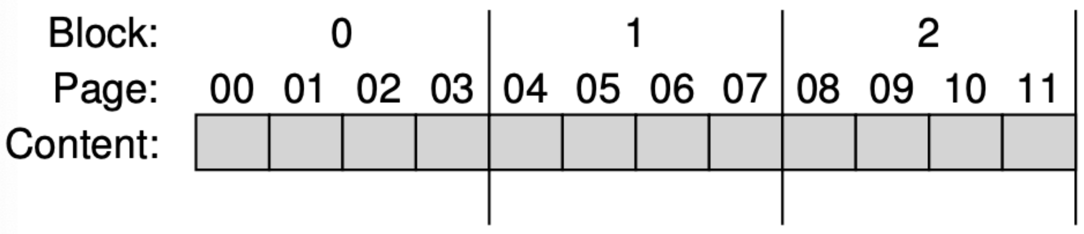
- Another issue is that frequently erase and write to a page/block will cause it to wear out (around 100,000 times)



(Source: The SSD Anthology)

Performance Characteristics of SSD

- Flash-based SSD provides the standard block interface
- To access a logical data block (e.g. 4 KiB)
  - The built-in control logic turns the requests into low-level read, erase, and write commands on the underlying physical blocks and physical pages
- Performance characteristics
  - Read a physical page
    - Able to access any location with the same performance - random access device
    - Typically quite fast, around 10s of microseconds
  - Erase the whole block
    - It is quite expensive as it takes a few milliseconds
    - In addition, to preserve some data in the block, they must be copied to somewhere before the erase
  - Write (program) to a page
    - Usually takes around 100s of microseconds



SSD Program/Erase: An Example

Page 0	Page 1	Page 2	Page 3
00011000	11001110	00000001	00111111
VALID	VALID	VALID	VALID

Page 0	Page 1	Page 2	Page 3
11111111	11111111	11111111	11111111
ERASED	ERASED	ERASED	ERASED

Page 0	Page 1	Page 2	Page 3
00000011	11111111	11111111	11111111
VALID	ERASED	ERASED	ERASED

HDD vs. SSD: Why do we still have HDD?

- Random I/Os: SSD >> HDD
- Sequential I/Os: SSD > HDD
- SSD: Random Reads < Random Writes?

	Random		Sequential	
Device	**Reads (MB/s)**	**Writes (MB/s)**	**Reads (MB/s)**	**Writes (MB/s)**
Samsung 840 Pro SSD	103	287	421	384
Seagate 600 SSD	84	252	424	374
Intel SSD 335 SSD	39	222	344	354
Seagate Savvio 15K.3 HDD	2	2	223	223

Abstraction of Persistent Storages

- Two key abstractions – Files and Directories
- What is a file?
  - From a user's perspective, it is a collection of related information that is recorded on persistent storage with a human-readable name given to it
  - From the system's perspective, it is a linear array of bytes, grouped in (logical) blocks, stored in somewhere, and has some kind of low-level id given to the file
    - In Unix systems, we call this low-level id – inode number
    - In Windows systems, it is called file reference number
  - This low-level id leads us to **a data structure**, where the attributes of the files are kept, e.g., locations of data of the file, ownership, etc.
- What is a directory?
  - Actually, it is a file, but its file content is a mapping table that maps filenames (in that directory) to their low-level ids
    - one entry for each file in that directory; can be a regular file or a directory file

File Systems

- Files (include directories) are managed by OS, and the part of OS dealing with files is known as the file system
  - File Management
    - Providing services to users and applications in the use of files & directories
    - Users should be able to refer to their files by symbolic names rather than having to use physical device names and physical location
  - Storage management
    - Allocating space for files on storage devices
  - File integrity
    - To guarantee, to the extent possible, that the data in the file are valid
  - Security
    - Data stored in file systems should be subject to strict access controls

File Abstraction

- From a user's standpoint, how to
  - locate the file, name the file, access the file, protect the file

- File system represents file as an abstract data type,
  - ***which consists of a set of operations:***
    - Open – associate the target file to the process, allowing the process to perform specific functions on the file (returns a file descriptor)
    - Close – process no longer perform functions on the file until it is reopened
    - Create – a new file is defined (space is allocated) and a new entry must be made in the directory
    - Delete – release all file space and erase the directory entry
    - Write – make updates to the file according to the current file pointer
    - Read – copy data (starting from location points by the current file pointer) from a file to the memory
- Open file table
  - Each process maintains an array of file descriptors, each of which refers to an entry in the system-wide open file table.
  - Each entry tracks which underlying file the descriptor refers to, the current offset, and other relevant details such as whether the file is readable or writable.

File Abstraction

- ***which consists of a set of attributes (metadata) associate to a file:***
  - Name – human-readable name
  - Low-level id – unique tag **identifies** a file within file system
  - Location – pointer to **storage locations of the data** of the file on device
  - Size – current file size
  - Accessibility – restrictions placed on access to file data
    - controls who can do Read, Write, Execute
  - Time, date, and user identification – data for protection, security, and usage monitoring

:
:

- Where to store the metadata that associated to each file?
  - Partly in the directory
  - Mostly in the **file control block** (FCB) of that file
    - In Unix, this is the **inode**; in Windows, this is the **file record**

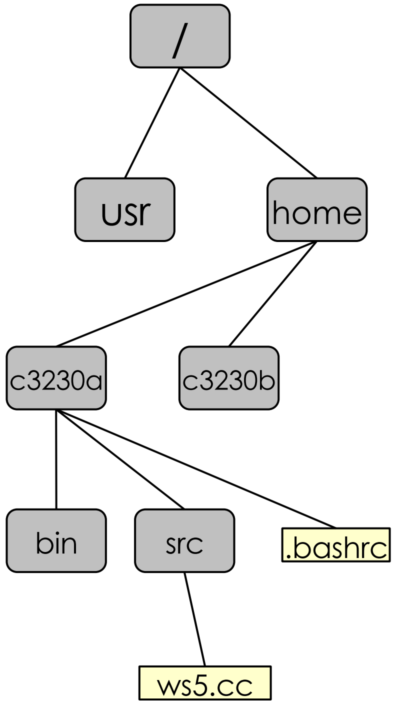
Directories

- As said, a directory is also a file
- We can view the content of a directory as a symbol table that maps file/directory names (in that directory) to the directory entries of that directory, which contain
  - the low-level id and a few other details of that file/directory
- Operations on directory
  - Search for a file
  - Create a file
  - Delete a file
  - List a directory
  - Rename a file
  - Traverse the file system

Name	Low-level ID
.	34
..	56
c0230a	123
c0234a	125

Directory Structure

- Hierarchically Structured File System
  - By placing directories within other directories, we have a directory tree, where all files and directories are stored
- A file system starts at a root directory "/"
  - The root directory contains various directories in the directory hierarchy
- The full name of a file is usually formed as the pathname from the root directory to the file – absolute path name
  - e.g., /home/c3230a/src/ws5.cc
  - Pros
    - File names need to be unique only within a given directory
      - Give more flexibility to users to name and group files
    - Efficient searching – by simply traverse the path to locate the files



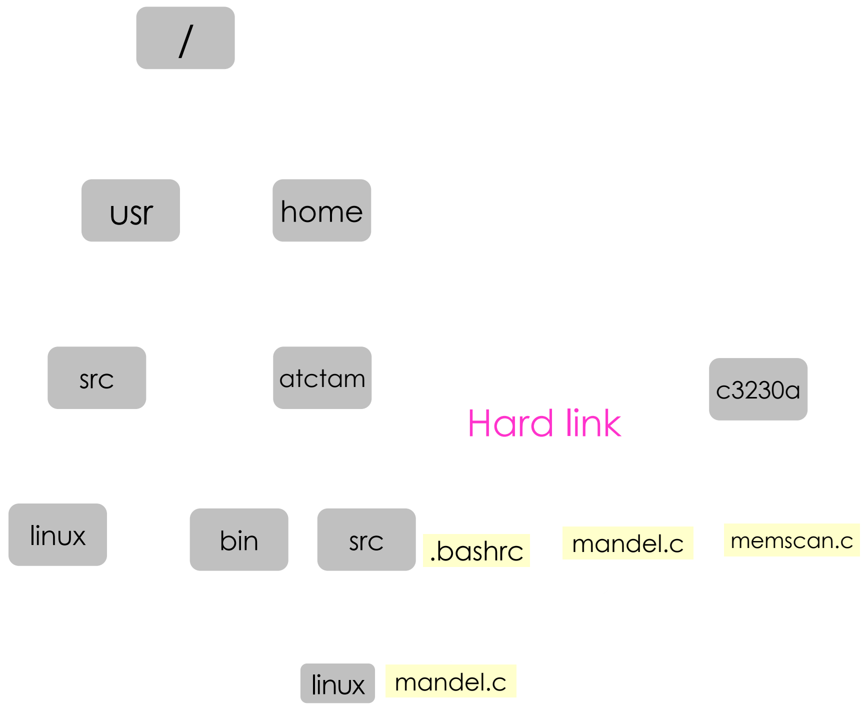
Directory Structure

- To simplify the navigation by using absolute path name, the concept of "Working directory" (Current directory) is used
  - Enables users to specify a pathname that does not begin at the root directory – relative path name
  - Absolute path (i.e., the path beginning at the root) = working directory + relative path name
- Link: a mechanism to create another directory entry that refers to an existing file/directory **in another location**
  - Adv: Facilitates data sharing and can make it easier for users to access files located throughout a file system's directory structure

Directory Structure

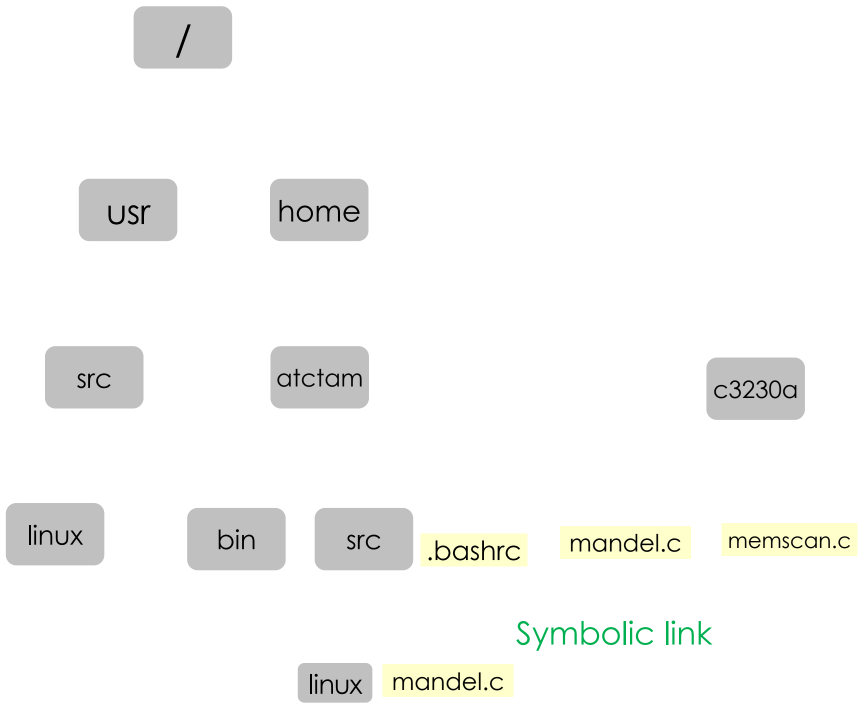
- Hard link: create another directory entry that **maps to the same low-level id** of the original file
  - Unix – **ln** target new
  - Windows – CreateHardLink
- The file is not copied at all; the system just creates two directory entries at different locations but refer to the same inode (file control block)

- Remove one directory entry will not cause the file to be deleted
  - The system keeps track on how many different directory entries have been linked to the same low-level id – reference count
  - The system deletes the file, only when reference count reaches zero



Directory Structure

- Limitation of hard link
  - Can't create a hard link to directory
  - Can't create hard link to files in other disk partitions (i.e. another file system)
- Symbolic link: create another file that contains the pathname of original file as its data
  - Unix – **ln -s** target new
  - Windows – mklink, Shortcut
  - Symbolic link is a special file type
  - Remove the original file causes the soft link to be invalid – dangling reference




```
atctam@atctam-LinuxPC:~/src> ls -l linux
lrwxrwxrwx 1 atctam users 14 Nov 30 16:48 linux -> /usr/src/linux
atctam@atctam-LinuxPC:~/src> stat linux
  File: `linux' -> `/usr/src/linux'
  Size: 14          Blocks: 0          IO Block: 4096   symbolic link
```

Summary

- Describe what is the basic structure of a storage disk
- Discuss a few factors that affect the performance of the storage systems
- Understand the key concept of the file system – the FILE
  - What a file is consisted of? How a file provide persistent storage to the user?
- Describe what is the purpose of using DIRECTORY
  - What a directory is? How it is being structured? How it is related to the files?

Operating Systems

- **Virtualization**
  - CPU Virtualization
    - Process Abstract
      - Address space
      - Process states
      - Process control block
      - Process operations API
      - Signals
    - Limited Direct Execution
      - System calls
      - Context switch
      - Interrupts
    - Scheduling
      - Scheduling metrics
      - FIFO, SJF, HRRN, STCF, RR, MLFQ
      - Multi-core scheduling, Linux CFS
  - Memory Virtualization
    - Address space
    - Address translation: dynamic relocation
    - Segmentation
    - Paging
    - TLB
    - Multi-level paging
    - Inverted page table
    - Swap space
    - Page replacement policy: FIFO, LFR, LRU, Clock
    - Thrashing
- **Concurrency**
  - Thread
    - POSIX threads (pthreads)
    - Race conditions, critical sections, mutual exclusion, atomic operations, synchronization
  - Locks
    - Atomic instructions: test-and-set, compare-and-swap
    - Mutex locks
  - Condition Variables
    - Pthread CVs
    - Producer-Consumer problem
  - Semaphores
    - Binary Semaphores
    - Counting Semaphores
    - Ordering
    - Readers-Writers problem
  - Deadlock
    - Dining philosophers' problem
    - Four necessary conditions
    - Deadlock prevention, avoidance, detection&recovery
- **Persistence**
  - I/O devices (HDD, SSD)
  - Files and Directories
    - Inode
    - File descriptor
    - Hard/Symbolic links
  - File System Implementation
    - On-disk data structure
      - Superblock, Bitmap, Inodes, Data blocks
    - Free space management
      - Bitmap, linked-list, block-list
    - Caching and buffering
    - Access control and protection
    - Journaling file system
      - Data journaling
      - Metadata journaling
- ***Advanced Topics***