IInd HALF
Homework 3

```
./cs5600-hw3 --cmdline /dev/null
```
gives file system explorer.

```
./hw3 <img file> <directory>
```
to mount itself on this.

```
./hw3 disk1.img <directory>
```
To see whether code can read the directories.
- Make copies of this image.

VFS – Virtual File System
Earlier 1st gen of OS there is no multiple FS.
Unix – basic structure of FS.

Goals
- Support any file system
- Varying levels of implementation
- Block management and other complex integration
- Simple syscalls like read()
- generic caching of objects.

Objects that Unix considers in FS

- i-nodes
  - its in every file/dir...
  - length of the file
  - permissions, pointers, etc.

Its identified by

filesystem: node #

- dir entry: entries in dir
  - this is cached
directory traversal:
lookup: to traverse directory paths
When OS uses cache entry, it revalidates with FS to check the value.
read dir

VFS has separate caches for nodes, attr-entry. The cache entry has pointer to access methods.

Berkeley FFS / Linux ext2/3

You have to guess the i-nodes to be kept aside for files.
- Can't write more files than 90%-95% of space.
This for performance, as free-space depletes searching a file takes more time.

If computer crashes FS becomes inconsistent while doing these operations.
Robustness of FS
- Sequentially doing the operations
Here FS structure won't get into bad state

- File system check when system is booting.

Writing meta-data synchronously makes it slow.
Since the file sizes FS↑↑, it takes lot of time to check FSck

Solution: Journaling

\[
\begin{align*}
\text{alloc } B_1 & \quad \rightarrow \text{log} \\
\text{list of actions} & \quad \rightarrow \{ \text{alloc } I_2 \\
\text{disk} & \quad \rightarrow \text{add entry } \{ B_1, I_2 \}
\end{align*}
\]

This is updated into the log.
Then you perform the actions. If system crashes you come back and check these operations.
Used to schedule operations, depending on how faster it is to access it on the disk.

LBA

Elevator algorithm. Higher logical addresses and than lower logical addresses.
This is command queuing. (high perf)

Tagged operation handling - SCSI (ex)
(This enhances ordering, for correctness)
Stream of requests.

Alternative for journaling
Log - Structure FS

Here you modify all the entries till root.
It's efficient for writing.
log structure

Here you make entire FS structure as journal

Costs

- Read time it can be inefficient
- Fragmented reads
- Write-once medium first
- Garbage collection is a problem

Advantages

- Flash file system
- Snapshots
  If you keep pointers of old roof. It keeps snapshots
- WAFL
  write-anywhere