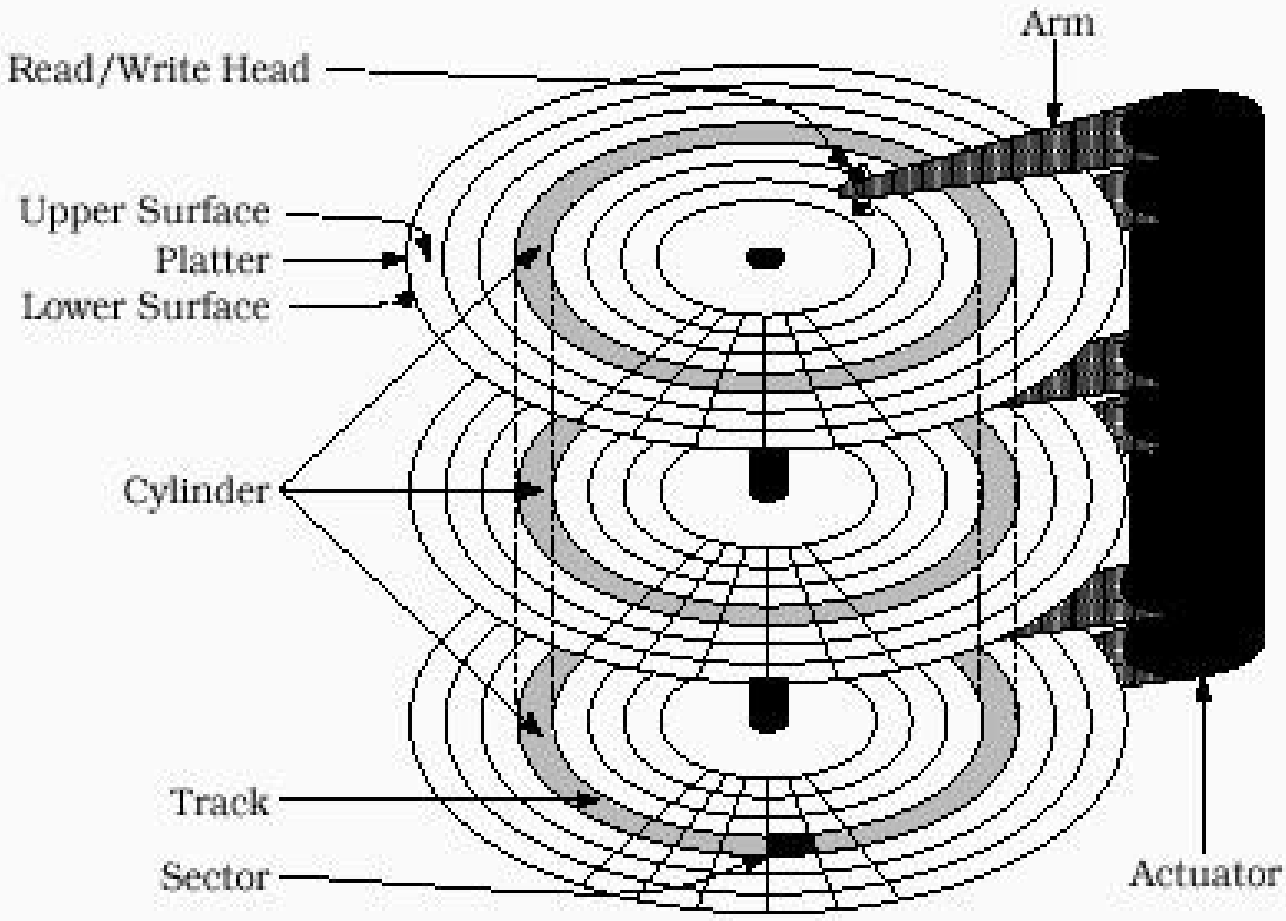


Figure 2.2: Important parts of a disk mechanism.

Application Performance in the QLinux Multimedia Operating System

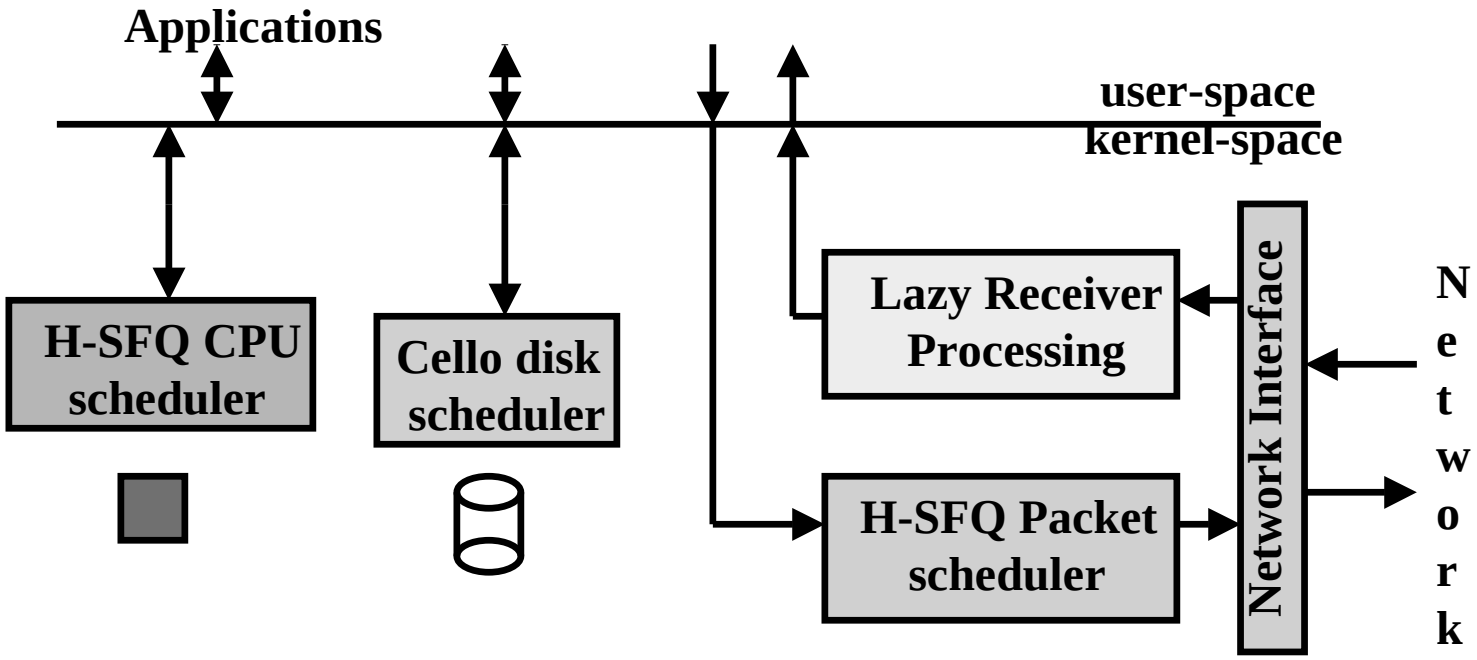
Vijay Sundaram, Abhishek Chandra
Pawan Goyal, Prashant Shenoy
Jasleen Sahni and Harrick Vin

http://lass.cs.umass.edu

UMASS Computer Science

QLinux Architecture

- QoS support using four components



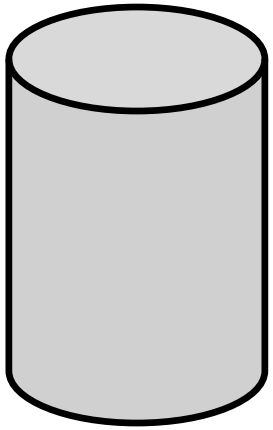
- Key contribution :
  - Integrate resource management mechanisms
  - Study impact on application performance

Lab4: File System

- Purpose of the Assignment
  - How the **Abstract notion of file system** is mapped to a **raw disk**
  - Disk is simulated using a **randomAccess** file class in **Java**
  - Allows us to open the disk in read/write mode and arbitrarily position the r/w head

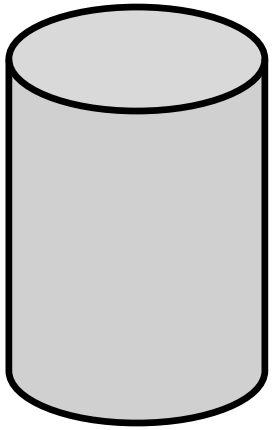
Disk

- Partitioned into 1024 bytes blocks
- Our disk has 256 blocks
- Blocks are used in two ways:



Disk

- Partitioned into 1024 bytes blocks
- Our disk has 256 blocks
- Blocks are used in two ways:
  - Data blocks
  - Directory blocks
- Free blocks: not really in use

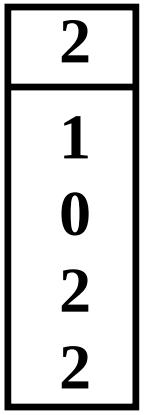


Disk.java

- Number of blocks
- Block size
- RandomAccessFile fp

Blocks

- 1024 byte blocks
- 2 bytes (type short) reserved for nextBlkPtr
  - Leaves only 1022 bytes for storing data
- Linked list structure for files
- "**nextBlkPtr**" used to store the block number of the next Block

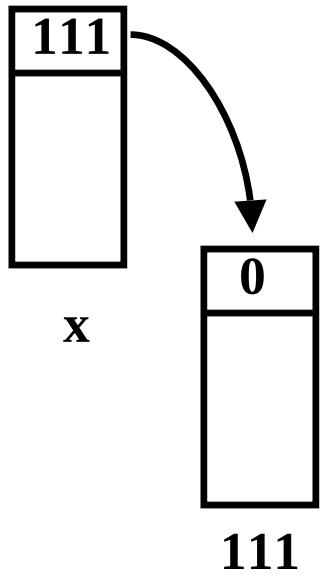


Block.java

- Some constants
- nameSize: limit on directory and file name
- maxFiles: set to 36.. why?? later.
- block_num to remember the block number
- Reference to the disk
- Abstract read and write methods

Data Block

- nextBlkPtr used to point to next block
- Defines a buffer
- Buffer size of 1022
- write/read
  - seek (seek to a byte offset)
  - writeShort/readShort
  - write

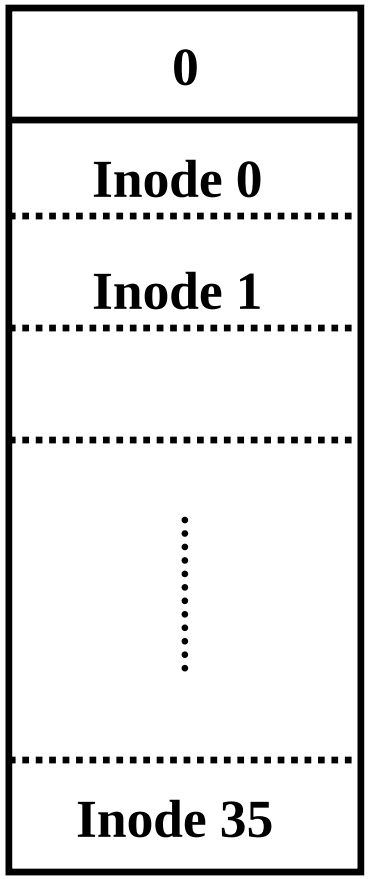


Directory Block

- Defines Inodes
- Creates individual Inodes

- Provides
  - list: list contents of directory
  - read/write: uses the inode methods
  - nextBlkPtr is not used

- Block 0 is the root directory
  - (see Disk.java for initialization)



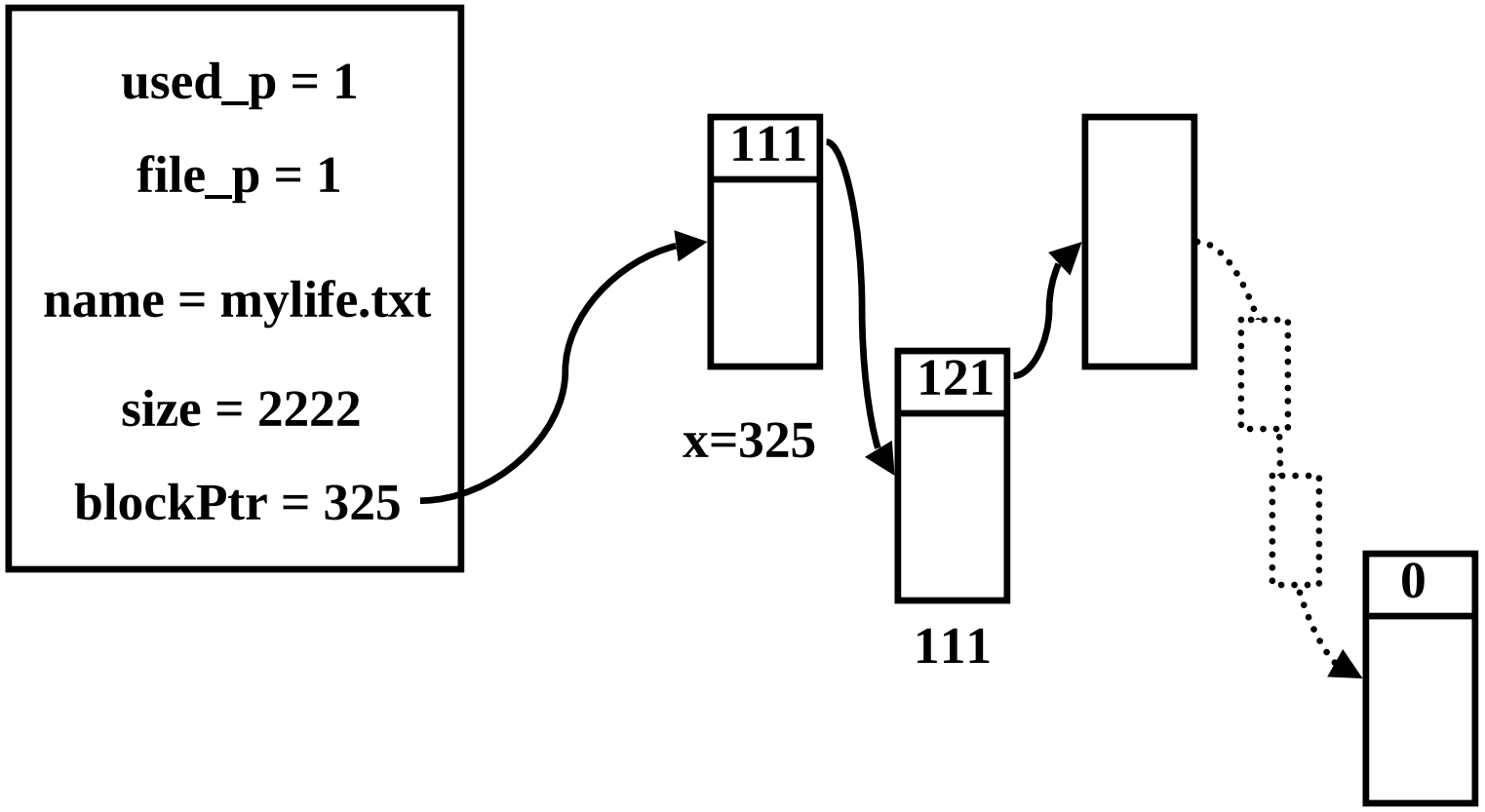
Inodes

- boolean used_p:
- boolean file_p:
- byte name[]:
- int size
- short blockPtr

1+1+20+4+2=28

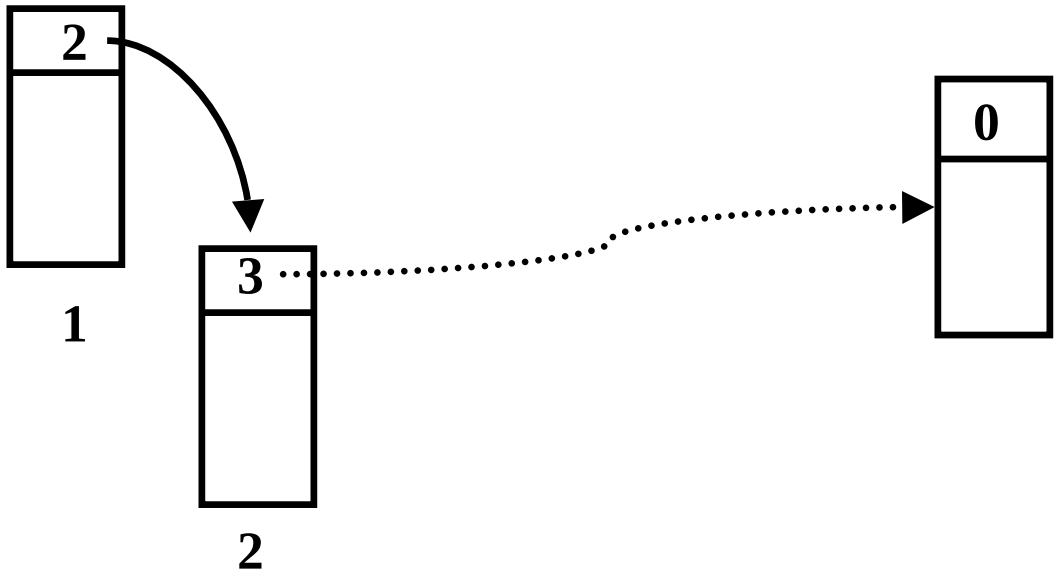
1022/28 = 36

Inodes (contd.)



FreeListBlk

- Define Block 1 as a Data Block, FreeListBlk
- Done in Disk.java to initialize the free list
- You manipulate it in AllocBlocks and DeallocBlocks



Some other files

- FileSystemAbstract.java
- FileSystemException.java
- UserInterface.java

You Implement

- DirectoryBlock.java
- FileSystem.java