File Systems

- Abstraction
  - Directories and Files instead of disks
- Protection

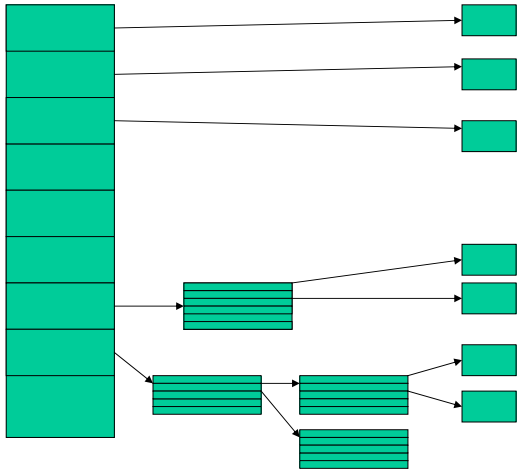
- Project: Simple UNIX-like File system

Inodes

- Which disk blocks go with which file.
- **Inode:** Data structure for bookkeeping
  - List of Blocks
  - File or Directory
  - Link Count
  - Other information…owner/permissions

Inode Structure

- Direct and Indirect Blocks



Inodes

- Advantages:
  - Fast access for small files (majority)
  - Supports large files
  - Supports sparse files

Directories

- Like a file: List of files and directories
  - name
  - inode number
- Can read it like a file
- Always has at least 2 entries:
  - "." current directory
  - ".." parent directory

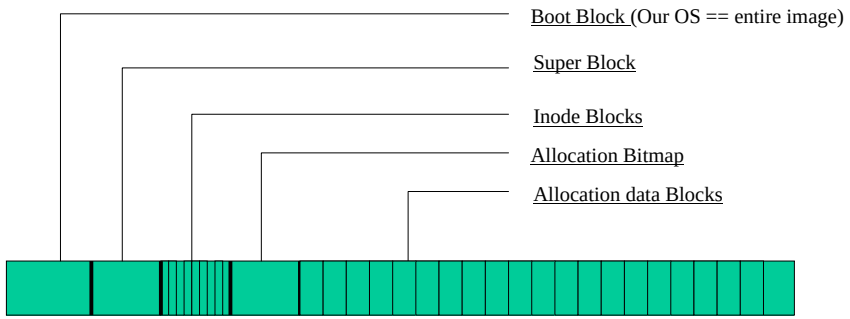
Super Block

- Contains the layout of the Disk
  - Size of Disk
  - Number of Inodes
  - Number of Data Blocks
  - Where inodes start, where data blocks start, etc….

Super blocks (cont.)

```
typedef struct {
    char signature[SIGN_SIZE]; /* Signature */
    int size;              /* Size of file system in blocks */
    int root_inode;        /* Inode no. of root directory */
    int inode_start;       /* First block for inodes */
    int inode_blocks;      /* Number of inode blocks */
    int bitmap_start;      /* First block for bitmap */
    int bitmap_blocks;     /* Number of blocks used to store the bitmap */
    int alloc_start;       /* First block managed by the allocater */
    int num_blocks;        /* Number of blocks for allocation */
    int free_blocks;       /* Number of free blocks: Note: IGNORE this since we do not want to update superblock frequently. */
} superblock;
```

Disk Layout



Project

- System calls to access file system
  - mkfs: Formatting
  - link, unlink:
  - open: file creation
  - close, read, write, lseek: file access
  - mkdir, chdir, rmdir: directory stuff
  - stat: information about a file or directory

Formatting(mkfs)

- Make a file system:
  - Write superblock
  - Mark inodes and data blocks to be free
  - Create root directory
  - initialize user file descriptor table
- fsck: Check integrity of file system
  - provided

File Creation / Deletion

- link: Hard link to a file
  - create a link to an existing file
  - hard vs soft link
- unlink: Delete a file if link count == 0
  - delete directory entry

File Access

- open: create file if it does not exist
- read:
- write:
- lseek: position in file
- close: free file descriptor

Directories

- Mkdir: make a directory
  - create an entry in parent directory
  - create two directories: “.”, “..”
- rmdir: remove directory if empty
- chdir: change the current directory
  - For relative path names

Example: mkdir()

```
int fs_mkdir(char *file_name) {
   if (file_name exists)  return ERROR;
   /* allocate data block */
   /* allocate inode */
   /* set directory entries for ‘.’, ‘..’ */
   /* set inode entries appropriately *
   /* update parent */
   return SUCCESS
}
```

Doing the Assignment

- Most under Linux environment
  - Use a file to simulate a disk (make lnxsh)
  - code is provided (*Fake files)
- Should be able to move right over to our OS.
- Shell supports
  - System calls for File System
  - Commands like “ls”, “cat”, “create” (create foo 200)