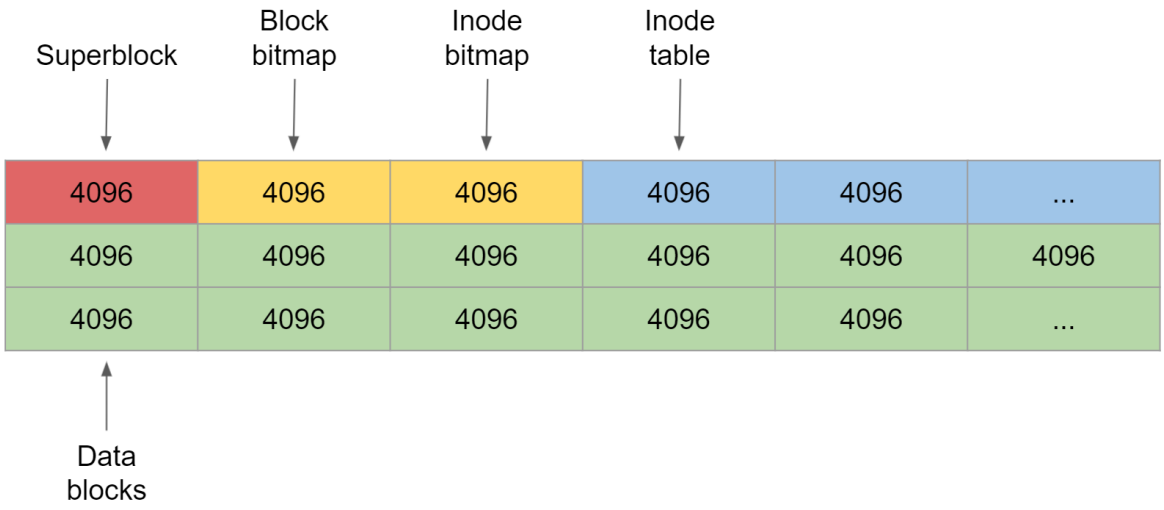
CSC369 Assignment 1a: File System Proposal

Xin Yao Yu
Yawen Xiao

➢ **A simple diagram of the layout of your disk image.**

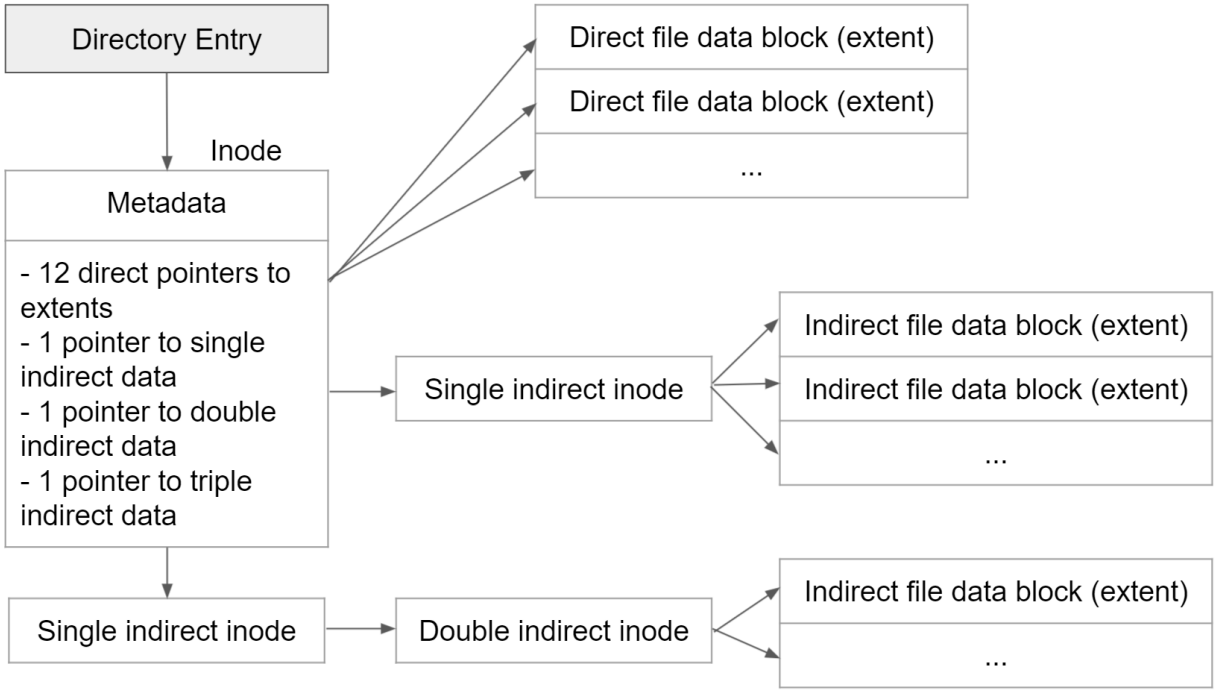


➢ **A description of how you are partitioning space.**

We will use the block bitmap and inode bitmap to keep track of the free data blocks and free inodes. The number of inodes, number of data blocks, address of the inode bitmap, block bitmap, inode table, and data blocks are stored inside the superblock. In order to determine which blocks are used for inodes, we will store the address of the data block used for each inode inside the inode. For instance, each inode will store a list of extents, and each extent will store the starting position of the data block of that extent used for the inode and the number of blocks in that extent. Using the information stored in each extent we will be able to know which blocks are used for inodes.

➢ **A description of how you will store the information about the extents that belong to a file.**

The extents will be stored inside the corresponding inode and the details of the extent (starting block of the extent and the number of blocks in the extent) will be stored in each of the extents. As shown in the diagram below, there is an array that contains 15 pointers inside the inode structure. We store each pointer to the extent that describes the location of the file data block.



➢ **Describe how to allocate disk blocks to a file when it is extended.**
We will use the block bitmap to see which block is free. And count the number of continuous free blocks to know the length of the extent that is free. For instance, if part of the block bitmap looks like 11000100, we know that there are two extents that are free, (2, 3) and (6, 2). One of length 3 and one of length 2. If we find the right place to allocate disk blocks to that file (depending on the size of the file), we will add the extent to the inode.

➢ **Explain how your allocation algorithm will help keep fragmentation low.**
There are three principles from the video to keep file fragmentation low:

Principle 1: Extend an extent if possible
Principle 2: Allocate single extent, if possible
Principle 3: Prefer to not break up large extents

Therefore, in our algorithm, if we want to write a new file, by principle 3 and principle 2, we want to find the smallest extent possible that this new file can fit in. If we cannot find a single extent for this file, we need to separate the file and allocate multiple extents. if we want to add to an existing file, by principle 1, we will first see if we can extend an extent without having to allocate a new extent. If

this is impossible, see if we can extend part of the data and the rest of the data can be treated as a new file and be stored to one or more new extents. If we don't have any space to extend the extent, then our last choice is to treat this as a new file and find the smallest extent possible that this file can fit in or separate the file and allocate multiple extents.

➢ **Describe how to free disk blocks when a file is truncated (reduced in size) or deleted.**

We will change the corresponding bit position in the block bitmap to 0. Since a truncate operation might not remove the entire extent (in some cases we have to remove the entire extent but not all of the extents stored inside the inode), we will also have to make revisions inside the inode. For instance, if the block bitmap looks like 10111010 with two extents (2, 3) and (7, 1) and after the truncate operation the block bitmap looks like 10100010, then we also have to change the extents stored inside the inode to from (2, 3) to (2, 1).

➢ **Describe the algorithm to seek to a specific byte in a file.**

Loop through the array of pointers in the inode and find out which extent stores the data block that contains the specific byte we want. Since the extent is described in the form of (a, b), where a is the starter block and b is the size. Given the byte, we can know the starter block of the file that contains the byte. By counting the size of each block, we seek the specific byte in the data block within the extent.

➢ **Describe how to allocate and free inodes**.

To allocate an inode, we need to fill in the metadata inside the inode and also update the inode table (change the corresponding bit that represents that inode from 0 to 1). To free inodes, we don't need to remove the metadata stored in the inode, we only need to update the inode table.

➢ **Describe how to allocate and free directory entries within the data block(s) that represent the directory.**

To free a directory entirely within the data block, I will only remove the name and pointer to the inode stored inside the directory entry that we want to remove. We don't need to remove the data inside the inode that the directory entry stores because removing the pointer inside the directory entry is enough to not be able to access the information inside the inode. If removing a directory entry empties a block, we need to free the block and update the information stored in the inode. To allocate a new directory entry, we need to check that the block has enough space to allocate a new directory entry, if yes, we can just append the new

directory entry into the list of directory entries stored inside the block. If not, we need to extend the file and then append the new directory entry into the new block.

- **Describe how to lookup a file given a full path to it, starting from the root directory.**

  Let's say the path is /rootdir/dir1/dir2, then we will first find the inode that stores the root directory (stored in the super block), then, we will look through the extents stored inside the inode. We go into each of the blocks in the extents and find the directory entry with the name "dir1", we get the inode number stored inside that directory entry and go into that inode. Again, we look through the extents stored inside the inode. We go into each of the blocks in the extents and find the directory entry with the name "dir2", then we get the inode number stored inside that directory entry and go into that inode to find the files stored inside that directory.