Kevin Jiang
9/23/2020
ECE357: Operating Systems
Problem Set 2, Problems 1 and 2

1. Problem 1
    a. 10001
    b. 10015
    c. 2 extent descriptors, because there are 17 data blocks, and one extent descriptor covers 15 data blocks at most. The extent descriptors are stored inside the block map portion of the inode (60 bytes).
    d. Only one more data block is allocated. This is an example of sparse allocation, where the data between 65600 and 409600 is undefined. All the data blocks between 10016 and 10100 would have block numbers of 0, meaning they are unallocated.
2. Problem 2
    a. Any empty directory will have a link count of 2: The "." Entry and the entry that exists in the directory's parent directory. Assuming no hard links are directories (since this is considered bad practice and can cause a lot of confusion within the file system), a link count of 4 means there are 2 more paths that point to this directory. These 2 paths are likely the ".." entries of 2 sub-directories within this directory.
    b. When you first open the file, the inode date of that file needs to be found by traversing through the file system and reading the inode that "/home/pcooper/stuff/12345.txt" points to. After this is done once, all the data for that particular inode is stored in a data structure called an in-core inode, which is stored in kernel memory. The in-core inode also stores other data that the kernel needs for accessing the same inode again. So after a few seconds when you try to open to file again, all the data needed for accessing that file already exists in kernel memory, which is much faster to access than persistent storage.
    c. Part C
        i. In transaction #1234, a new file is being created. The inode for that file has number 9999, and the inode for the directory that the file is being created in has number 100. After creating a new file, mtime, which is the timestamp for the last write operation performed to a file or directory, is updated. In transaction #1235, data is being written to the file we just created. The amount of data being written is less than or equal to 4K, since only one new block is allocated, and size is shown to be 4096. The new block that is allocated is block 5555.
        ii. Inode #9999 should be of type S_IFREG, and it is a non-free inode. Journal mode is ordered, so these journal entries were only written after the actual transaction is committed.

iii. There will be no perceptible corruption because both transactions have been committed and the journal is in ordered mode. The same data will just be written to those blocks again.
iv. It is possible to have perceptible corruption. In writeback mode, the metadata in the journal may have been written before the actual data was written to the file system. This means for example, block 5555 may just have garbage in it, and replaying the writing to that block will result in the same garbage.

d. The inode number will not be 10. It will be 2, because now /mnt/usbdrives/volume01 is obscured by the root of /dev/sdb1. This is fine because the kernel keeps tracks of which inodes are being used as mount points and can distinguish if a directory is in a mounted volume.
e. This may be because /A/B and /A/C are in the same volume, while /A/Z is in a different volume. Moving files within the same volume just involves the rename() system call, which basically creates a new hard link (/A/C/F2) and removes the old hard link (/A/B/F1). This should be very fast since you are only creating a new pointer to the same file inode. However, moving to a different volume cannot be done with a new hard link, because hard links cannot be created between volumes. An actual copy of F1 needed to be made on the second volume, which involves allocating new data and other operations on the second volume. This would be much slower.