‘

OS Assignment 5

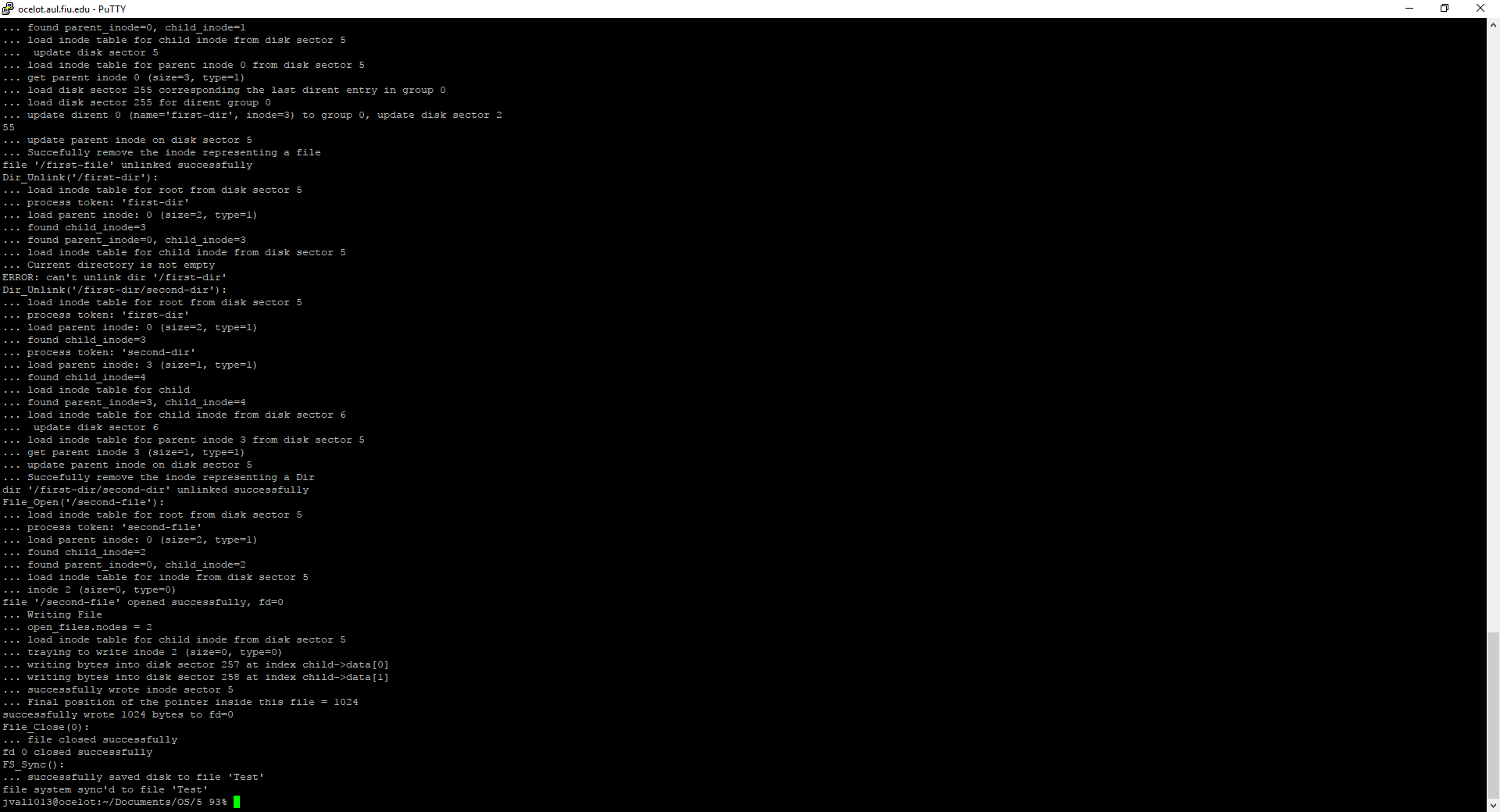
Alejandro Vidal 5913959

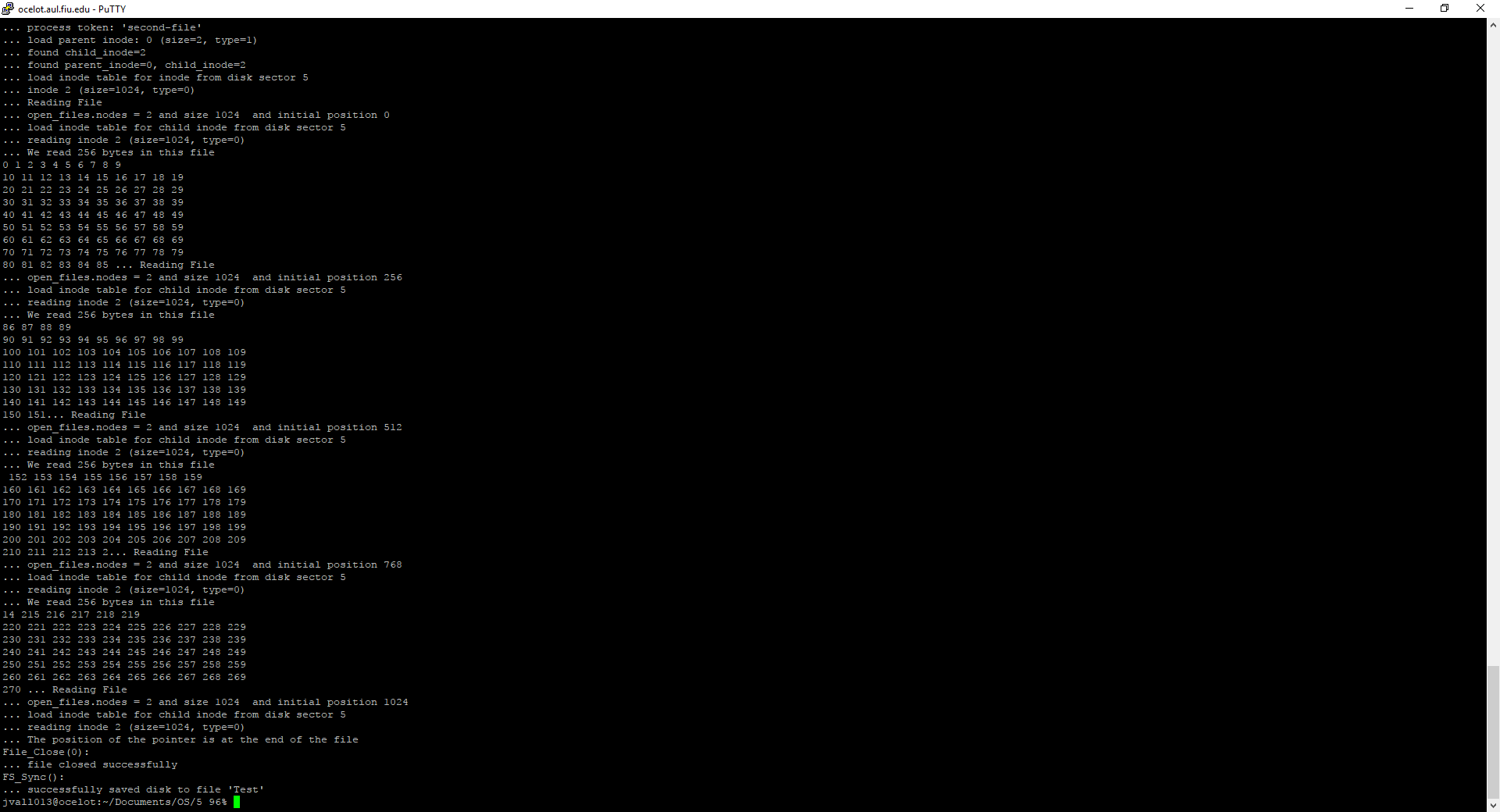
Juan C Valladares 2676611

12/7/2017

The design of our file system was based on the implementation that professor Liu provided. File System would work with a super block, two bitmaps, one to map the inode table, and another to map the sectors of the whole disk. Then there would be the inode table which has the information of where and how much information the files or directories have, finally after the inode table, the data would be stored in the sectors after.

Our system has an FS\_Boot functions that is called once. It initialize the file system by creating a file that is the disk image. We then have FS\_Sync that ensures that the structure of the file system is in place.

We then have a function int File\_Create and dir\_Create which are similar. In professor Liu’s implementation this calls create\_file\_or\_directory. Which then calls the function add\_inode. Add inode turns the first zero in the bitmap of inodes to 1, then creates the inode in the inode table. Below is an example of File\_Create and Dir\_create succesfully working. File\_write writes a buffer passed into the function into the data blocks pointed by the inodes. Here is an example, which also includes File\_Unlink and Dir\_Unlink, who do the opposite of File\_Create. They use the remove\_inode function, which essentially sets the bitmap reference to the inode to 0, and then removes the inode from the inode table. So next time there is a write, that space will be available as nothing will point to those sectors in the disk. Finally we have File\_Close, which removes a file’s entry from the open\_file\_table, so that it can’t be read or opened unless it’s there. Here is an example of all those functions working together:

File\_Open opens up a file. If the file doesn’t exist there is an error. Otherwise the file’s inode information is placed in the Open\_File\_Table. We then have File\_Read(), which only reads from files that are open the amount that was asked for and reads it into a buffer that was passed in through a pointer. If the file isn’t open, there is an error. Then we have file\_write, which receives a buffer, the function gets the inode and find the sector where to write the buffer into. Here is all three working together:

We also have Read\_Dir, which goes through the Dirent in the dir\_Inode and gets back that information and puts it in a buffer. Below is an example:

