‘

OS Assignment 5

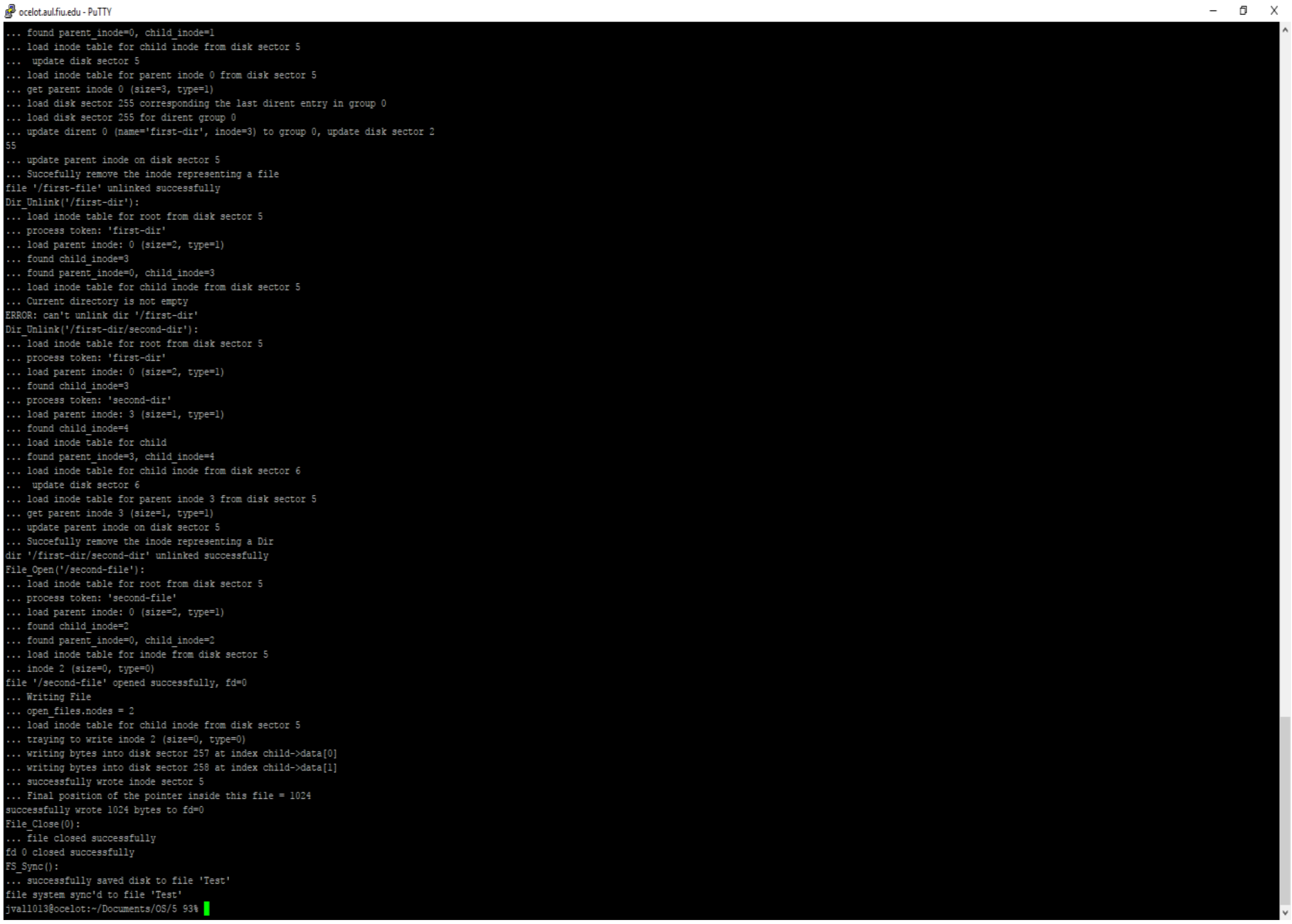
Alejandro Vidal	5913959
Juan C Valladares	2676611

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.

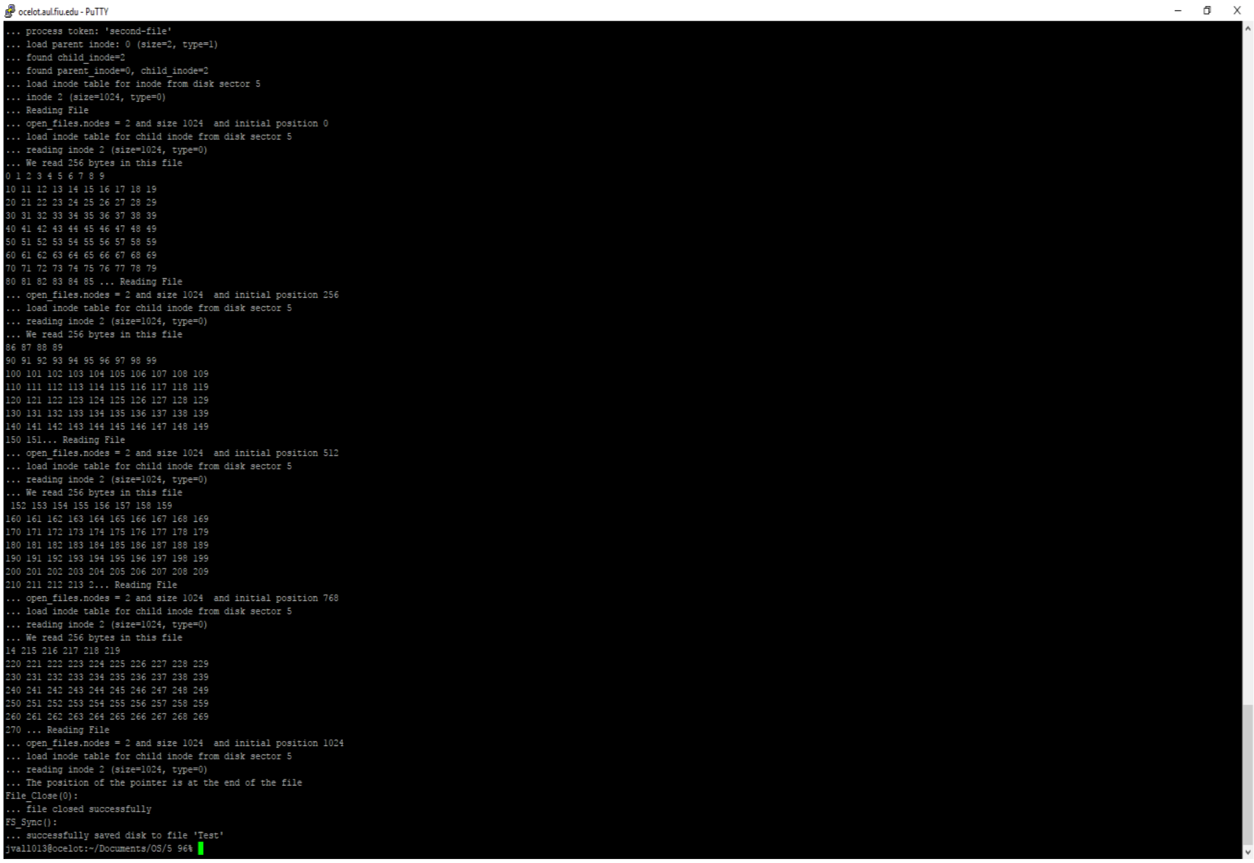
Our system has an FS_Boot functions that is called once. It initialize the file system by creating a file that is the disk image. We then have FS_Sync that ensures that the structure of the file system is in place.

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:



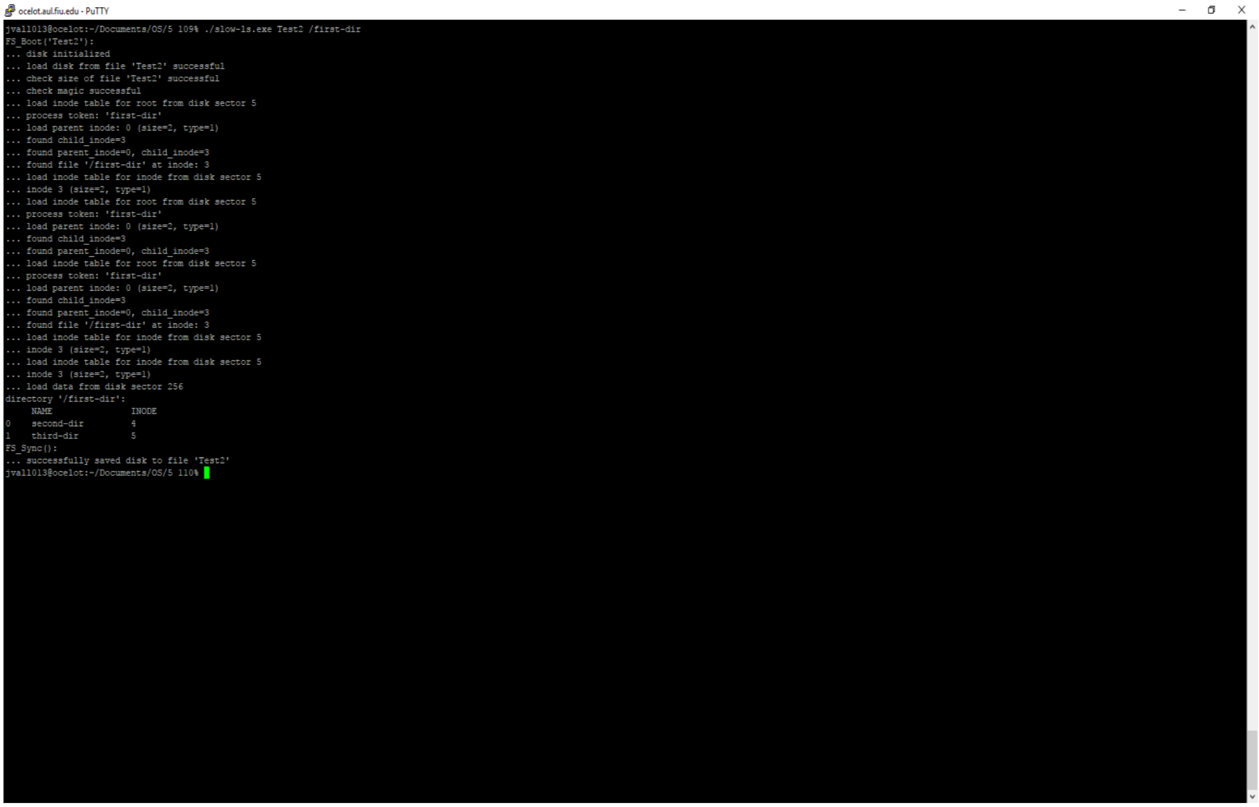
```
... found parent_inode=0, child_inode=1
... load inode table for child inode from disk sector 5
...  update disk sector 5
... load inode table for parent inode 0 from disk sector 5
... get parent inode 0 (size=3, type=1)
... load disk sector 255 corresponding the last dirent entry in group 0
... load disk sector 255 for dirent group 0
... update dirent 0 (name='first-dir', inode=3) to group 0, update disk sector 2
55
... update parent inode on disk sector 5
... Succefully remove the inode representing a file
file '/first-file' unlinked successfully
Dir_Unlink('/first-dir'):
... load inode table for root from disk sector 5
... process token: 'first-dir'
... load parent inode: 0 (size=2, type=1)
... found child_inode=3
... found parent_inode=0, child_inode=3
... load inode table for child inode from disk sector 5
... Current directory is not empty
ERROR: can't unlink dir '/first-dir'
Dir_Unlink('/first-dir/second-dir'):
... load inode table for root from disk sector 5
... process token: 'first-dir'
... load parent inode: 0 (size=2, type=1)
... found child_inode=3
... process token: 'second-dir'
... load parent inode: 3 (size=1, type=1)
... found child_inode=4
... load inode table for child
... found parent_inode=3, child_inode=4
... load inode table for child inode from disk sector 6
...  update disk sector 6
... load inode table for parent inode 3 from disk sector 5
... get parent inode 3 (size=1, type=1)
... update parent inode on disk sector 5
... Succefully remove the inode representing a Dir
dir '/first-dir/second-dir' unlinked successfully
File_Open('/second-file'):
... load inode table for root from disk sector 5
... process token: 'second-file'
... load parent inode: 0 (size=2, type=1)
... found child_inode=2
... found parent_inode=0, child_inode=2
... load inode table for inode from disk sector 5
... inode 2 (size=0, type=0)
file '/second-file' opened successfully, fd=0
... Writing File
... open_files.nodes = 2
... load inode table for child inode from disk sector 5
... traying to write inode 2 (size=0, type=0)
... writing bytes into disk sector 257 at index child->data[0]
... writing bytes into disk sector 258 at index child->data[1]
... successfully wrote inode sector 5
... Final position of the pointer inside this file = 1024
successfully wrote 1024 bytes to fd=0
File_Close(0):
... file closed successfully
fd 0 closed successfully
FS_Sync():
... successfully saved disk to file 'Test'
file system sync'd to file 'Test'
jvall013@ocelot:~/Documents/OS/5 93%
```

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:



```
... process token: 'second-file'
... load parent inode: 0 (size=2, type=1)
... found child_inode=2
... found parent_inode=0, child_inode=2
... load inode table for inode from disk sector 5
... inode 2 (size=1024, type=0)
... Reading File
... open_files.nodes = 2 and size 1024  and initial position 0
... load inode table for child inode from disk sector 5
... reading inode 2 (size=1024, type=0)
... We read 256 bytes in this file
0 1 2 3 4 5 6 7 8 9
10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49
50 51 52 53 54 55 56 57 58 59
60 61 62 63 64 65 66 67 68 69
70 71 72 73 74 75 76 77 78 79
80 81 82 83 84 85 ... Reading File
... open_files.nodes = 2 and size 1024  and initial position 256
... load inode table for child inode from disk sector 5
... reading inode 2 (size=1024, type=0)
... We read 256 bytes in this file
86 87 88 89
90 91 92 93 94 95 96 97 98 99
100 101 102 103 104 105 106 107 108 109
110 111 112 113 114 115 116 117 118 119
120 121 122 123 124 125 126 127 128 129
130 131 132 133 134 135 136 137 138 139
140 141 142 143 144 145 146 147 148 149
150 151... Reading File
... open_files.nodes = 2 and size 1024  and initial position 512
... load inode table for child inode from disk sector 5
... reading inode 2 (size=1024, type=0)
... We read 256 bytes in this file
 152 153 154 155 156 157 158 159
160 161 162 163 164 165 166 167 168 169
170 171 172 173 174 175 176 177 178 179
180 181 182 183 184 185 186 187 188 189
190 191 192 193 194 195 196 197 198 199
200 201 202 203 204 205 206 207 208 209
210 211 212 213 2... Reading File
... open_files.nodes = 2 and size 1024  and initial position 768
... load inode table for child inode from disk sector 5
... reading inode 2 (size=1024, type=0)
... We read 256 bytes in this file
14 215 216 217 218 219
220 221 222 223 224 225 226 227 228 229
230 231 232 233 234 235 236 237 238 239
240 241 242 243 244 245 246 247 248 249
250 251 252 253 254 255 256 257 258 259
260 261 262 263 264 265 266 267 268 269
270 ... Reading File
... open_files.nodes = 2 and size 1024  and initial position 1024
... load inode table for child inode from disk sector 5
... reading inode 2 (size=1024, type=0)
... The position of the pointer is at the end of the file
File_Close(0):
... file closed successfully
FS_Sync():
... successfully saved disk to file 'Test'
jvall013@ocelot:~/Documents/OS/5 96%
```

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:



```
jvall013@ocelot:~/Documents/OS/5 109% ./slow-ls.exe Test2 /first-dir
FS_Boot('Test2'):
... disk initialized
... load disk from file 'Test2' successful
... check size of file 'Test2' successful
... check magic successful
... load inode table for root from disk sector 5
... process token: 'first-dir'
... load parent inode: 0 (size=2, type=1)
... found child_inode=3
... found parent_inode=0, child_inode=3
... found file '/first-dir' at inode: 3
... load inode table for inode from disk sector 5
... inode 3 (size=2, type=1)
... load inode table for root from disk sector 5
... process token: 'first-dir'
... load parent inode: 0 (size=2, type=1)
... found child_inode=3
... found parent_inode=0, child_inode=3
... load inode table for root from disk sector 5
... process token: 'first-dir'
... load parent inode: 0 (size=2, type=1)
... found child_inode=3
... found parent_inode=0, child_inode=3
... found file '/first-dir' at inode: 3
... load inode table for inode from disk sector 5
... inode 3 (size=2, type=1)
... load inode table for inode from disk sector 5
... inode 3 (size=2, type=1)
... load data from disk sector 256
directory '/first-dir':
     NAME              INODE
0    second-dir        4
1    third-dir         5
FS_Sync():
... successfully saved disk to file 'Test2'
jvall013@ocelot:~/Documents/OS/5 110% 
```