File Project – Answer Document

Basic Submission (earns a maximum of 88 points)

bio.c has several test cases. You will build and run TFS to understand the file system and how it is tested. You will use the hexdump tool on tinyfs to further your understanding.

1. Build and run tiny, using all three of the arguments - create, write, read. You can read above and study main in bio.c for the current content of these three arguments. Copy/paste your run log here.
2. Update main in bio.c to write data to a file you create. Also update main to read data from your file. For both of these updates, I suggest you retain small sizes for writes and reads for you initial update. Copy/paste your run log here.
3. Use hexdump to dump the file tinyfs, which is the TFS TDD to verify that your files are on the TDD. You must locate several aspects of them. Their names, inode numbers, and data block(s) in the root directory. Copy/paste your use of hexdump here. Annotate your copy/paste to demonstrate understanding.
4. Create an annotated diagram of TFS. Show TFS structures and how they interconnect. Include the superblock, the inode bitmap, the data bitmap, the inodes, and two files - one of which is what you added to main in step 2. Also show file descriptors, struct proc, struct file, struct inode and how a file descriptors are converted to inodes. When creating this diagram, use the hex dump from step 3 to demonstrate your understanding. Place your annotated diagram here.
5. Create a function call trace starting with tfs\_read. Create this trace by reading the code. For each function, describe the parameters and a detailed description of what the function does. For example,
   1. tfs\_read’s (in file tfsfile.c) parameters are a file descriptor, a buffer, and a size to read. tfs\_read calls fd\_to\_file to convert the file descriptor to a struct file. Then tfs\_read calls fileread to continue reading the file.
   2. fd\_to\_file (in file tfsfile.c) parameters are an fd and a struct file \*\*. fd\_to\_file ensures the fd is within range and returns the pointer to the struct file in the proc’s ofile[].
   3. fileread (in file.c) parameters are … continue

Place your function call trace here.

1. Update hexdump to include a new flag. You can select the flag. For example, you could add a -i flag that dumps inodes, or you can add a -e flag that has the end block. Copy/paste a run log of hexdump applied to tinyfs with your new flag.
2. Submit your updated bio.c and hexdump.c files.

Advanced Submission (earn 2 points per test)

Update main in bio.c to test additional tfs\_ functions. You will first have to investigate the Linux equivalent to discover what it is supposed to do. Then you can try the tfs\_ version to see if it works. When something does not work, you will be forced into deep study of the underlying code base, which will sharpen your understanding of file system implementation. You can update bio.c with explicit test calls similar to the provided tests. A beautiful update of bio.c would include a shell-like interface that allows you to open, close, link, mkdir, etc. Select from (or do all) the following tests.

1. Create a tinyfs file with over 512 bytes of data. This will cause a second block to be allocated. See that the inode indicates two data blocks. Copy/paste a hexdump of tinyfs showing your large file.
2. Repeat step 6a, but this time write to the file in steps. First write data that fits on a single data block. Then create another file with data. Then write more data to the first file to cause it to grow to two data blocks. This will show a file with two non consecutive data blocks. Copy/paste a hexdump of tinyfs showing your large file with data blocks separated.
3. tfs\_fstat - equivalent to fstat. Copy/paste a log showing your tfs\_fstat test.
4. tfs\_link - equivalent to link. Copy/paste a log showing your tfs\_link test.
5. tfs\_unlink - equivalent to unlink. Copy/paste a log showing your tfs\_unlink test.
6. tfs\_mkdir - equivalent to mkdir. Copy/paste a log showing your tfs\_mkdir test.
7. tfs\_chdir - equivalent to chdir.  Copy/paste a log showing your tfs\_chdir test.
8. tfs\_lseek - equivalent to lseek.  Copy/paste a log showing your tfs\_lseek test.
9. Submit your updated bio.c file that includes your test cases.