Programming Assignment 4

CS450 Fall, 2019

1. This assignment is a pair programming effort. It is due on 11/29/2019
2. **Purpose:** We described the EXTEND method to improve the accessing time for large files. But what about small files? It was observed that more than 20% of the files of a typical user are less than 64 bytes long. The purpose of this programming assignment is to implement a method that will improve the overhead in handling small files in xv6.
3. **Requirements:**
    1) For files smaller than certain size (what size?), we ask you to modify the xv6 file system to store the data directly in the inode itself. No need for another data block!
    2) We ask you to create a directory creation command **mkSFdir**. Files created under small files directories are expected to be small and fits into an inode.
    3) The **open(), read()** and **write()** system calls will behave differently when small files are involved. What else? Implement these system calls. They should continue to work with other normal files.
    4) The base requirement of this assignment will not allow a small file to be linked to a directory not created by **mkSFdir**. But we ask you to provide a design to allow small files to be linked to other directories.
4. **Simplifications and hints:**
    1) You will create a new types of files and directories (see **stat.**h) but will maintain the old pointer-based code for backward compatability and simplicity.
    2) The system call **write()** may overflow a small file. There are two ways to handle it: (a) do not allow it; or (b) change the file into a normal file. (a) is the base requirement. (b) is for extra credit from design to actual implementation.
    3) Keep the inode structure exactly as it is except for those data block points. You will find the structure of the inode in **fs.h**.
    4) You will need to modify the data block allocation and deallocation function **bmap()** in the file system.
    5) Most of the code that you will be dealing with are in **fs.h** and **fs.c**.
5. **Deliverables:**

1) A document that describes your design. The various existing functions and header files that you changed and why. Describe your write overflow and link to normal directories methods.

2) The test data that you use and explanation on why the test data is of good quality. If you use the equivalence partitioning method to generate your test data, describe your equivalence partitions.

3) Describe each file that you changed (Notice using:$ diff -uw “$orginal” “$modified$ > ./diff/“$orginal.txt”).

4) Source(with complete xv6 source code) and executable objects with a README on how to build and execute them. If you use an version of xv6 other than the one in http://github.com/mit-pdos/xv6-public you need to provide the details of how to make it work and how to compile it.

5) Upload all files as a **zip** archive as FirstName_LastName_CWID_PA3.zip. Documents and readme only supports: txt, doc, docx and pdf format.

6) Self evaluation is due 24 hours after submission.