Bilkent University

Department Of Computer Engineering

`

Operating Systems Project

Project 4 Report

Arda Göktoğan 21702666

Mehmet Bora Kurucu 21703404

Instructor: Prof. Dr. İbrahim Körpeoğlu

Introduction

In this project, we implemented a very simple file system, which consists of a single (root) directory and its files. Architecture is designed based on the instructions given in the assignment and details are discussed in the next section. Then we conducted experiments to analyse the performance of the file system based on various read and append operations.

File System Architecture

File system architecture consists of blocks and entries. First block is **superblock**, which contains the following metadata about the file system:

1- Number of blocks (1x4 byte = 4 byte)
2- Integer array to keep valid/invalid entries in the root directory. (112 x 4byte = 448 byte)

The rest of the superblock is idle.

Each root directory block contains 16 root directory entries. Th structure of the **root directory entry** is as follows:

1 - File Name (64 x 1 byte = 64 byte)
2 - File Size (1 x 4 byte = 4 byte)
3 - Number of First content block (1 x 4 byte = 4 byte)

The rest of the root directory entry is idle.

Root directory blocks are followed by **FAT blocks**. There are 256 FAT blocks and each contains 512 **FAT entries**. We decided to keep occupied blocks information in the FAT entries because it is possible to have 128K blocks and it is not possible to keep that array in the superblock. Therefore FAT entries consist of the following members:

1 - Next Block (1 x 4 byte = 4 byte)
2 - Flag for current block is occupied (1 x 4 byte = 4 byte)

Therefore we achieved 8 byte FAT entry size as desired.

Experiments

Below the experiments. Timer for append to file 1000 bytes per step.

File Size(Append)	Time to Insert
0	real 0m8,979s user 0m0,000s sys 0m0,001s
1000	real 0m8,,212s user 0m0,001s sys 0m0,000s
2000	real 0m8,350s user 0m0,001s sys 0m0,000s
3000	real 0m9,353s user 0m0,001s sys 0m0,000s
4000	real 0m10,247s user 0m0,001s sys 0m0,001s

File Size(Read)	Time to Insert
0	real 0m0,018s user 0m0,004s sys 0m0,000s
1000	real 0m0,017s user 0m0,009s sys 0m0,004s
2000	real 0m0,019s user 0m0,005s sys 0m0,002s
3000	real 0m0,021s user 0m0,003s sys 0m0,002s
4000	real 0m0,021s user 0m0,004s sys 0m0,006s

Generally, writing to the file takes more time than reading, because reading continues from where it is left in our implementation. Real time increases as the size of the file increases for both read and write operations.