CS 342 Operating Systems

Project 4
A Simple File System

Musa Ege Ünalan 21803617

Mustafa Göktan Güdükbay 21801740

1. Implementation Details

The disk is divided into blocks of size 4096. The first block is the superblock. The superblock contains information about how many blocks are in the disk and how many are currently being used. The subsequent four blocks are the bitmap. Each bit in a bitmap block tells if the respective block is free or not.

The next four blocks are for *directory* entries. Each directory entry is 128 bytes. There can be at most 128 directories in the disk. A directory entry holds information about the filename (at most 110 characters); if any directory uses the entry, the file control block index.

The next four blocks are the *file control block* entries. Each file control block is 128 bytes. There can be at most 128 file control blocks in the disk. A file control block holds information about the size of the file, the location of the index block, if the file control block is used, the permission (read or append), the read pointer, and the write pointer.

When a file is created, an *index* block is assigned. A block to hold the file data is also assigned, and its location is written into the index block.

We made several tests to check if the library can correctly create, read, append, and delete files.

2. Experiment

We performed experiments to measure the computation times of file create, read and write operations. First, we measured the time to create 128 different files with sizes 4096 bytes. We then measured the total time to create 8, 16, 32, 64, and 128 files of 4096 bytes. Please note that these created files are empty. Then we measured the time to append 65,536 bytes using buffer sizes of 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, and 8192 bytes. We then measured the time to read 65,536 bytes using buffer sizes of 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, and 8192 bytes.

Table 1 provides the computation times of creating each file of size 4096 bytes (in μs). Table 2 gives the total computation time to create 8, 16, 32, 64, and 128 files of 4096 bytes (in μs). Table 3 gives the time to append 65,536 bytes using buffer sizes 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, and 8192 bytes (in μs). Table 4 gives the time to read 65,536 bytes using buffer sizes 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, and 8192 bytes (in μs).

Figure 1 provides the graph of the computation times of creating each file of size 4096 bytes (in μs). Figure 2 gives the graph of the total computation time to create 8, 16, 32, 64, and 128 files of 4096 bytes (in μs). Figure 3 gives the graph of the time to append 65,536 bytes using buffer sizes of 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, and 8192 bytes (in μs). Figure 4 gives the graph of the time to read 65,536 bytes using buffer sizes of 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, and 8192 bytes (in μs).

Table 1.The computation times of creating each file of size 4096 bytes (in μs).

File Number	Time (μs)		File Number	Time (μs)		File Number	Time (μs)		File Number	Time (μs)
1	336		33	20		65	22		97	25
2	19		34	19		66	23		98	26
3	17		35	20		67	21		99	24
4	16		36	21		68	23		100	24
5	18		37	20		69	22		101	24
6	16		38	19		70	21		102	23
7	17		39	21		71	21		103	23
8	17		40	19		72	21		104	25
9	17		41	19		73	21		105	23
10	17		42	19		74	23		106	24
11	16		43	20		75	23		107	24
12	18		44	20		76	22		108	24
13	16		45	19		77	23		109	23
14	16		46	19		78	24		110	34
15	17		47	19		79	21		111	24
16	17		48	20		80	22		112	25
17	17		49	20		81	22		113	28
18	17		50	19		82	21		114	25
19	17		51	18		83	22		115	24
20	17		52	19		84	22		116	25
21	18		53	27		85	22		117	24
22	16		54	19		86	23		118	24
23	17		55	20		87	22		119	27
24	17		56	22		88	21		120	24
25	16		57	20		89	22		121	23
26	16		58	20		90	22		122	24
27	16		59	20		91	22		123	25
28	18		60	21		92	22		124	24
29	17		61	20		93	23		125	23
30	17		62	22		94	22		126	25
31	17		63	20		95	22		127	23
32	17		64	20		96	22		128	25

Table 2. The total computation time to create 8, 16, 32, 64, and 128 files of 4096 bytes (in μs).

Number of Files Created	Total Time (us)
8	136
16	278
32	544
64	1,172
128	2,628

Table 3. The time to append 65,536 bytes using buffer sizes 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, and 8192 bytes (in µs)

Buffer Size	Time (us)
1	532,443
2	265,340
4	132,133
8	66,783
16	33,636
32	16,745
64	8,327
128	4,278
256	2,209
512	1,175
1,024	682
2,048	392
4,096	243
8,192	154

Table 4.The time to read 65,536 bytes using buffer sizes 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, and 8192 bytes (in µs).

Buffer Size	Time (us)
1	364,051
2	188,779
4	93,791
8	48,135
16	23,990
32	11,834
64	5,696
128	2,851
256	1,368
512	705
1,024	354
2,048	188
4,096	115
8,192	73

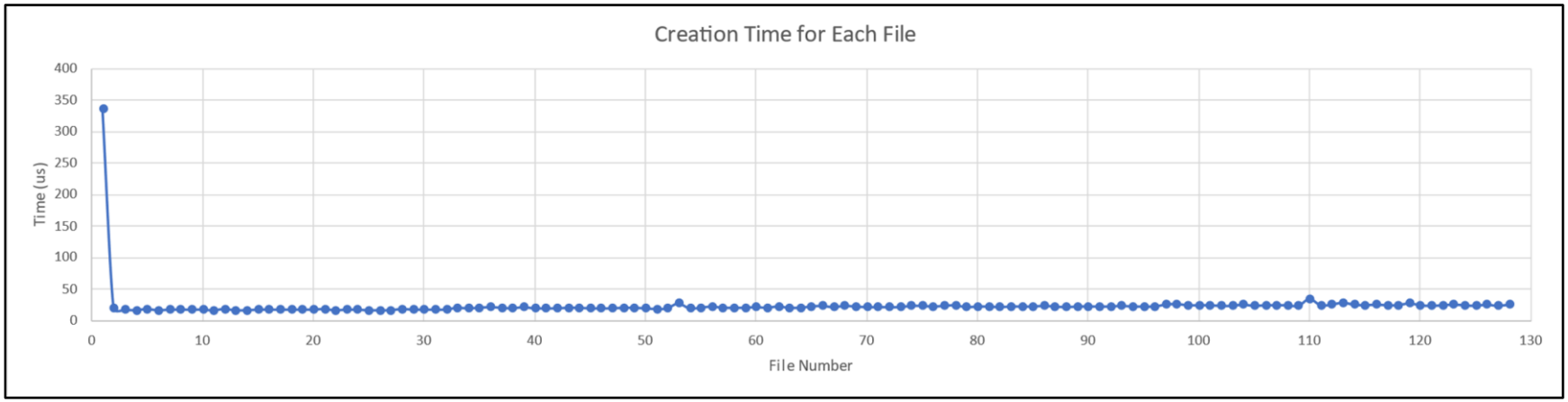

Figure 1. The graph of the computation times of creating each file of size 4096 bytes

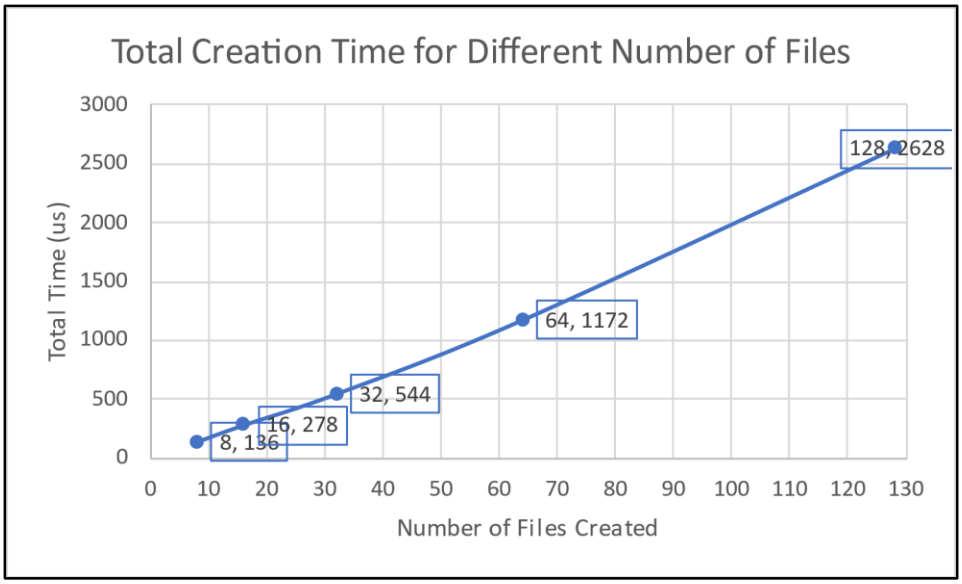

Figure 2. The graph of the total computation times to create 8, 16, 32, 64, and 128 files of 4096 bytes.

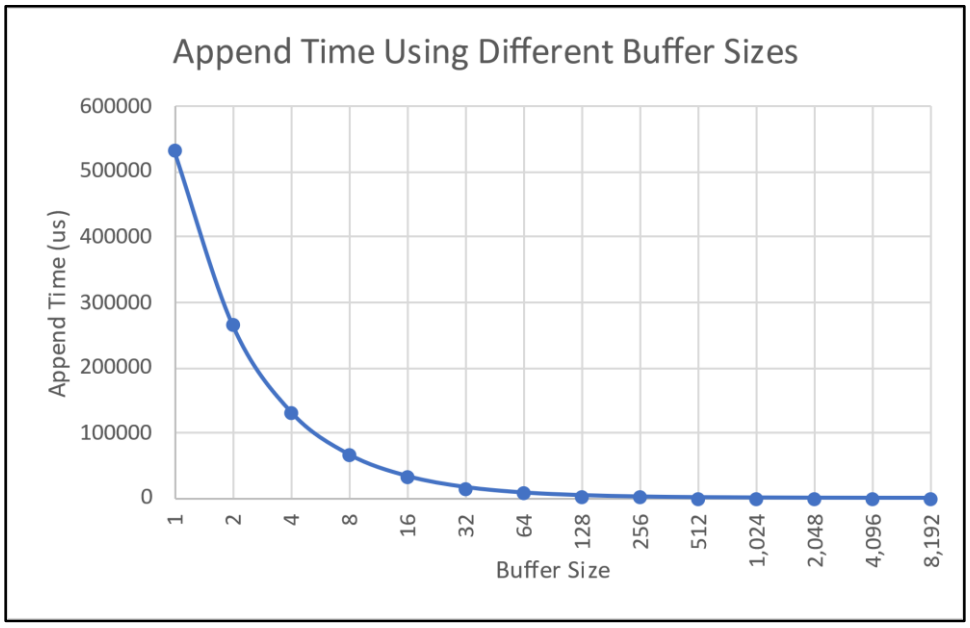

Figure 3. The graph of the times to append 65,536 bytes using buffer sizes 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, and 8192 bytes.

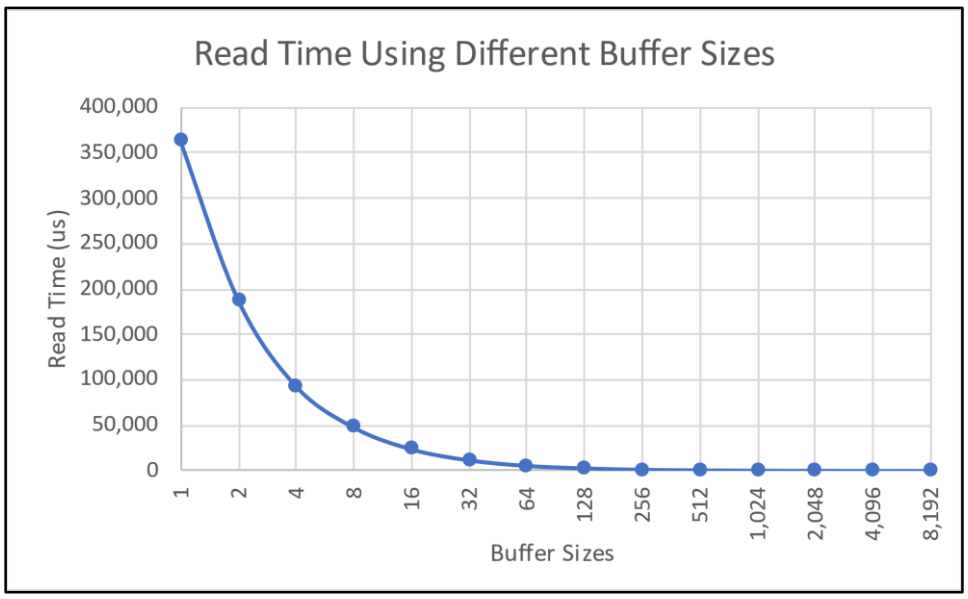

Figure 4.The graph of the times to append 65,536 bytes using buffer sizes 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, and 8192 bytes.

3. Analysis

The creation time for each file is similar, although they are stored in different blocks. There are slight differences. We think that these differences arise from the internal paging of the system and some initializations. The time to create the first file is significantly higher compared to the times to create the subsequent files due to the initialization cost. Besides, the times of creating the subsequent files increase slightly with the increasing file number because they are created in the later disk blocks and reaching these blocks takes slightly more time.

The total time to create a number of files increases linearly with the number of files. This is meaningful because the operations performed for each file are the same. The cost of these operations is multiplied by a constant factor for each file.

The time to append decreases as the buffer size increases because the append operation requires fewer write block operations with the increasing number of buffer sizes.

The time to read also decreases as the buffer size increases because the read operation requires fewer read block operations with the increasing number of buffer sizes.

Appendix: Experiment Code

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include "simplefs.h"
#include <sys/time.h>

int main(int argc, char **argv)
{
    int ret;
    int fd;
    char filename[12];
    char vdiskname [12];
    strcpy (vdiskname, "disk1");
    struct timeval startTime;
    struct timeval finishTime;

    sfs_mount(vdiskname);

    //measuring ith file creation
    for(int i = 0; i < 128; i++){
      gettimeofday(&startTime, NULL);
      snprintf(filename, 12, "file%d.bin", i);
      sfs_create(filename);
      gettimeofday(&finishTime, NULL);
      printf("Time to create file %d: %li microseconds\n", i ,
           1000000*(finishTime.tv_sec - startTime.tv_sec) +
           finishTime.tv_usec - startTime.tv_usec);
      //printf("%li\n", 1000000*(finishTime.tv_sec - startTime.tv_sec)
           + finishTime.tv_usec - startTime.tv_usec);
    }
```

```
//deleting files
for(int i = 0; i < 128; i++){
  snprintf(filename, 12, "file%d.bin", i);
  sfs_delete(filename);
}

//measuring 8, 16, 32, 64, 128 file creation Times
for(int i = 0; i < 5; i++){
  gettimeofday(&startTime, NULL);
  for(int j = 0; j < (8 << i); j++){
    snprintf(filename, 12, "file%d.bin", j);
    sfs_create(filename);
  }
  gettimeofday(&finishTime, NULL);
  printf("Time to create %d files: %li microseconds\n", (8<<i) ,
       1000000*(finishTime.tv_sec - startTime.tv_sec) +
       finishTime.tv_usec - startTime.tv_usec);

  for(int j = 0; j < (8 << i); j++){
    snprintf(filename, 12, "file%d.bin", j);
    sfs_delete(filename);
  }
}

//append times
for(int i = 0; i < 14; i++){
  int bufferSize = (1 << i);
  char* buffer = malloc(bufferSize);
  sfs_create("FILETOAPPEND.bin");
  int fd = sfs_open("FILETOAPPEND.bin", 1);
  for(int k = 0; k < bufferSize; k++)
    buffer[k] = 'a';

  gettimeofday(&startTime, NULL);
```

```
  for(int j = 0; j < (65536/ bufferSize); j++){
     sfs_append(fd, (void*)buffer, bufferSize);
  }
  gettimeofday(&finishTime, NULL);
  printf("File Size: %d, Buffer Size %d, Append Time: %li\n",
       sfs_getsize(fd), bufferSize, 1000000*(finishTime.tv_sec -
       startTime.tv_sec) + finishTime.tv_usec - startTime.tv_usec);

  sfs_close(fd);
  sfs_delete("FILETOAPPEND.bin");
  free(buffer);
}


//read times
//create a file of size 65536 bytes

sfs_create("FILETOREAD.bin");
fd  = sfs_open("FILETOREAD.bin", 1);
char* buffer = malloc(65536);
for(int i = 0; i < 65536; i++)
  buffer[i] = 'a';

sfs_append(fd, buffer, 65536);
sfs_close(fd);

free(buffer);

for(int i = 0; i < 14; i++){
  int bufferSize = (1 << i);
  char* buffer = malloc(bufferSize);
  int fd = sfs_open("FILETOREAD.bin", 0);

  gettimeofday(&startTime, NULL);
```

```
      for(int j = 0; j < (65536/ bufferSize); j++){
         sfs_read(fd, (void*)buffer, bufferSize);
      }
      gettimeofday(&finishTime, NULL);
      printf("File Size: %d, Buffer Size %d, Read Time: %li\n",
           sfs_getsize(fd), bufferSize, 1000000*(finishTime.tv_sec -
           startTime.tv_sec) + finishTime.tv_usec - startTime.tv_usec);

      sfs_close(fd);
      free(buffer);
    }
    sfs_delete("FILETOREAD.bin");

}
```